Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

TIPS & TRICKS
WAWANCARA KERJA MENJADI PROGRAMMING

YAYASAN PRIMA AGUS TEKNIK

TIPS & TRICKS
WAWANCARA KERJA MENJADI PROGRAMMING

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang dan dari Universitas Kristen Satya Wacana Salatiga (UKSW) Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-5734-00-2
9 786235 734002

TIPS & TRICKS
WAWANCARA KERJA
MENJADI PROGRAMMING

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

TIPS &TRICKS, WAWANCARA KERJA MENJADI PROGRAMMING

Penulis :
Dr. Ir. Agus Wibowo, M.Kom., M.Si., MM.

ISBN : 9 786235 734002

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

Hak cipta dilindungi undang-undang
Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa ijin dari penulis

KATA PENGANTAR

Programming adalah suatu proses atau kegiatan menulis dan menguji (pemrograman) agar program dapat dibuat, dan hasilnya sesuai apa yang diinginkan. Apa yang ditulis dalam proses tersebut? Tentunya, bukan tulisan biasa. Tidak seperti saat menulis buku novel ataupun puisi karena ini berhubungan dengan program komputer. Sehingga bahasa tulisan pun harus dapat dimengerti oleh komputer, yakni berupa kode-kode program yang sudah ada sebelumnya.

Saat ini banyak bermunculan bahasa pemrograman yang baru. Hal ini terjadi karena seseorang berupaya menyederhanakan dan memudahkan bahasa pemrograman yang sudah ada. Tujuannya untuk pembuatan program jauh lebih cepat, efektif, dan sesederhana mungkin. Anda akan mengenal bahasa pemrograman yang biasa digunakan untuk programming. Perkembangan teknologi yang begitu cepat tentu membawa dampak bagi bahasa pemrograman. Salah satunya adalah mulai bermunculan bahasa pemrograman yang baru.

Buku ini mempelajari bagaimana tips untuk proses mendapat pekerjaan dengan memahami tahap-tahap melakukan prosedur wawancara dengan baik dan benar yang harus diperhatikan. Tujuan wawancara adalah agar pewawancara mempekerjakan seseorang yang akan bekerja dengan baik dalam pekerjaan tertentu, sebagian besar orang yang melakukan wawancara tidak menerima pelatihan, kurang pengalaman wawancara dan bahkan sering tidak bersusah payah mempersiapkan wawancara.

Wawancara juga tentang membangun hubungan dan kepercayaan, dan ada hal-hal yang dapat Anda lakukan (hal-hal yang tidak boleh kamu lakukan) yang akan sangat membantu meningkatkan keterampilan kamu dalam semua area penting wawancara. Idealnya, kamu harus berlatih pada wawancara nyata. Semakin banyak wawancara yang Anda hadiri, semakin baik — bahkan jika Anda harus menghadiri wawancara untuk pekerjaan yang sebenarnya tidak kamu minati.

Langkah untuk kesuksesan wawancara memberikan kerangka kerja yang mudah digunakan yang dengannya Anda dapat menangkap semua informasi relevan yang Anda butuhkan untuk menyusun jawaban wawancara. Selain menangkap apa yang Anda lakukan dan bagaimana Anda melakukannya, itu juga memaksa Anda untuk berpikir tentang konteks dan hasil. Ini cocok untuk menjawab pertanyaan perilaku dan dapat digunakan dengan cara yang fleksibel.

Dalam pekerjaan pemrograman seperti yang Anda ketahui, sebagian besar posisi pengembang Java yang menguntungkan menuntut keterampilan dan pengalaman multi-threading Java inti yang sangat baik dalam mengembangkan, men-debug, dan menyetel aplikasi Java bersamaan dengan latensi rendah yang berkinerja tinggi dan Multithreading atau Concurrency adalah salah satu topik populer dalam pertanyaan wawancara java.

Pada Buku Tips and Trick Progamming menjelaskan dengan jelas pertanyaan dan jawaban mengenai bab Java Multi Threading dan kurensi, Exception Java, Java Miscellaneous, Core Java, Java Collection, Java 8, dan Standar Pengkodean Java IX. Semoga buku ini bermanfaat bagi para Pembaca.

Ungaran, Oktober 2021
Penulis

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

DAFTAR ISI

BAB 1
MITOS WAWANCARA

Salah satu alasan penting orang gagal dalam wawancara karena beberapa kesalahpahaman atau mitos tentang apa yang sebenarnya terjadi selama wawancara. Kita semua tahu bahwa tujuan wawancara adalah agar pewawancara mempekerjakan seseorang yang akan bekerja dengan baik dalam pekerjaan tertentu, tetapi di luar itu hanya sedikit orang yang sepenuhnya memahami bagaimana wawancara benar-benar bekerja dan apa yang membuat satu kandidat lebih menonjol daripada yang lain.

Kurangnya pemahaman ini merupakan kendala utama untuk memaksimalkan kinerja saat duduk di hadapan pewawancara dan mencoba memberikan jawaban terbaik Anda.

Wawancara tidak berbeda dengan upaya lain dalam hidup: semakin baik kamu memahami cara kerjanya (atau tidak berhasil), semakin tinggi kemungkinan berhasil menanganinya. Pemahaman tentang dinamika mendasar yang melekat dalam sebagian besar wawancara adalah awal yang penting untuk meningkatkan kinerja wawancaramu.

Mitos nomor 1: Orang terbaik untuk pekerjaan itu mendapatkannya -

Kadang-kadang ini benar - terutama dalam situasi di mana semua orang mengenal orang lain, seperti ketika perusahaan merekrut secara internal. Namun, seringkali tidak demikian. Agar orang terbaik untuk pekerjaan tersebut memenangkannya, sejumlah hal yang sangat penting harus ada (dan bahkan kemudian, tidak ada jaminan).

Ini termasuk :

Pewawancara tahu pertanyaan apa yang harus diajukan dan bagaimana mencari kebenaran dalam jawaban. Kedua hal ini mungkin terdengar cukup sederhana, tetapi saya dapat meyakinkan dirimu bahwa sebagian besar orang yang melakukan wawancara tidak menerima pelatihan, kurang pengalaman wawancara dan bahkan sering tidak bersusah payah mempersiapkan wawancara.

Pewawancara tidak terpengaruh oleh pesona, ketampanan, humor yang bagus atau aspek lain dari orang yang diwawancarai. Ini bisa menjadi kendala yang sulit, bahkan bagi pewawancara berpengalaman.

Orang yang diwawancarai telah belajar bagaimana mengartikulasikan keterampilan mereka dengan jelas, pencapaian utama dan bagaimana mereka dapat menambah nilai bagi organisasi.

Tidak ada benturan kepribadian antara pewawancara dan orang yang diwawancarai. Tidak ada pihak yang mengalami hari yang buruk.

Beberapa bos biasanya adalah orang-orang yang pernah mengalami luka parah karena mempekerjakan orang yang salah di masa lalu — berusaha keras untuk mengatur prosedur perekrutan profesional yang dirancang untuk meminimalkan kesalahan perekrutan. Meskipun beberapa dari prosedur ini efektif dalam

meningkatkan pemilihan kandidat, namun tidak menjamin bahwa orang terbaik untuk pekerjaan itu benar-benar akan memenangkannya.

Dalam analisis terakhir, memilih seseorang untuk suatu pekerjaan melibatkan setidaknya satu manusia membuat keputusan tentang orang lain, dan apa pun yang kita lakukan untuk menghilangkan subjektivitas, sebagai manusia tidak mungkin mengesampingkan kecenderungan, kecenderungan, dan preferensi pribadi kita — tidak peduli seberapa banyak kita berusaha.

Dalam dunia yang ideal, orang terbaik untuk pekerjaan itu akan selalu memenangkannya. Namun, kenyataannya sering kali orang yang melakukan wawancara terbaiklah yang memenangkan hadiah. Pelajaran penting di sini adalah:

Jangan berhenti melamar pekerjaan secara otomatis jika kamu tahu seseorang yang lebih cocok untuk pekerjaan itu juga melamar pekerjaan itu.

Jika kamu mengalami kesulitan dalam mempersiapkan wawancara dengan benar, ada kemungkinan kamu akan terlihat sebagai kandidat yang disukai terutama jika orang lain menerima begitu saja wawancara dan gagal mempersiapkannya. Jika kamu kebetulan tahu bahwa dirimu orang terbaik untuk pekerjaan itu, hindari menerima wawancara begitu saja. Berperilaku seolah kamu sedang bersaing dengan rival yang tangguh. Luangkan waktu untuk mempersiapkan diri dengan baik. Hanya karena kamu memiliki banyak pengalaman tidak berarti kamu tahu bagaimana menyampaikan pesan ini dalam sebuah wawancara.

Mitos tidak. 2: Wawancara itu seperti ujian sekolah- Semakin banyak Anda berkata, semakin baik hasil Anda -

Ya, wawancara itu seperti ujian sejauh kamu ditanyai sejumlah pertanyaan yang perlu kamu tanggapi dengan cerdas, tetapi di sanalah kesamaannya berakhir. Tidak seperti ujian, di mana banyak detail yang akurat itu penting, wawancara lebih banyak tentang berinteraksi dan membangun hubungan sambil mengartikulasikan jawaban cerdas secara bersamaan. Dan jawaban cerdas seringkali bukan yang paling detail. Faktanya, jawaban yang panjang dan terlalu mendetail dapat membuat pewawancara terganggu, terlepas dari akurasi teknisnya. Mengetahui kapan harus berhenti berbicara adalah keterampilan yang dimiliki semua orang yang diwawancarai yang sukses.

Juga tidak seperti banyak ujian, seringkali tidak ada jawaban benar atau salah dalam wawancara. Kami semua berbeda dan mengikuti wawancara dari berbagai latar belakang dan situasi bisnis. Yang penting dalam wawancara adalah membenarkan tindakanmu dan membicarakan pencapaian dirimu dengan cara yang percaya diri.

Mitos nomor 3: Pewawancara tahu apa yang mereka lakukan -

Beberapa pewawancara sangat pandai dalam apa yang mereka lakukan, terutama profesional penuh waktu (asalkan mereka tidak menderita kelelahan wawancara). Namun, banyak manajer dan pemilik usaha kecil sering gagal karena wawancara bukanlah sesuatu yang mereka lakukan secara teratur.

Beberapa tanda pasti dari pewawancara yang buruk adalah:

- Mereka yang paling banyak bicara.
- Mereka terdengar seolah-olah telah mengambil keputusan tentang kamu dalam lima menit pertama.
- Mereka tampaknya memetik pertanyaan mereka secara acak.
- Telepon mereka terus berdering dan mereka menjawabnya.
- Mereka terdengar seperti tenaga penjualan yang sangat tajam dan kurang jujur dalam hal menjual pekerjaan.

Beberapa tanda pasti dari pewawancara yang baik adalah:

- Mereka telah mempersiapkan pertanyaan mereka dengan hati-hati sebelumnya.
- Mereka ingin tahu apa yang telah kamu lakukan dan bagaimana kamu melakukannya, termasuk contoh spesifiknya.
- Mereka membiarkan kamu melakukan sebagian besar pembicaraan.
- Mereka mungkin ingin mewawancarai kamu lebih dari sekali.
- Mereka akan mencoba membuat kamu merasa nyaman.
- Mereka benar-benar tertarik dengan pencapaian, keterampilan, dan tipe orang kamu.

Pewawancara yang tidak berpengalaman biasanya tidak mengajukan pertanyaan yang tepat dan dapat dengan mudah dipengaruhi oleh faktor-faktor yang tidak ada hubungannya dengan kemampuan dirimu, untuk melakukan pekerjaan tersebut. Jadi jika kamu sedang diwawancarai oleh pewawancara yang tidak berpengalaman, jangan menunggu untuk ditanyai pertanyaan yang bagus yang akan memungkinkan kamu untuk berbicara tentang semua keterampilan dan kualitasmu yang luar biasa.

Sebaliknya, ambil inisiatif dengan cara sesederhana mungkin dan bicarakan hal-hal yang menurut kamu benar-benar ingin diketahui pewawancara. Sayangnya, hal ini tidak selalu memungkinkan terutama jika kamu diwawancarai oleh orang yang kuat yang menyukai suaranya sendiri.

Jika kamu pernah berada dalam situasi seperti itu, jangan panik. Ingatkan dirimu bahwa wawancara sama pentingnya dengan membangun hubungan seperti halnya menjawab pertanyaan. Jadi anggukkan kepalamu, tersenyumlah dan buat semua suara yang tepat. Pewawancara yang cerewet menyukai orang-orang yang setuju dengan mereka.

Mitos nomor 4: Jangan pernah mengatakan 'Saya tidak tahu' Wawancara adalah tentang membuat kesan positif dengan menjawab pertanyaan secara cerdas dan membangun hubungan baik dengan pewawancara. Untuk tujuan ini, banyak narasumber merasa bahwa mereka harus memberikan jawaban yang sempurna untuk setiap pertanyaan yang diajukan kepada mereka, terlepas dari apakah mereka benar-benar mengetahui jawabannya atau tidak.

Jelas, wawancara yang bagus adalah wawancara di mana kamu bisa menjawab semua pertanyaan (dan kamu harus bisa melakukannya jika kamu meluangkan waktu untuk mempersiapkan dengan benar). Namun, jika Anda tidak tahu jawaban untuk sesuatu, lebih baik mengakuinya daripada berpura-pura tahu dan mulai bimbang.

Sebagian besar pewawancara dapat memilih omong kosong satu mil jauhnya dan mereka tidak menyukainya karena beberapa alasan yang sangat penting: pertama, itu mungkin membuat kamu terdengar tidak jujur; dan kedua, itu akan membuat kamu terdengar kurang cerdas. Kamu mungkin juga tidak menghadiri wawancara jika kamu memberi kesan bahwa dirimu tidak jujur atau cerdas.

Mencoba menjawab pertanyaan yang tidak terlalu kamu ketahui dapat merusak wawancara yang bagus. Ini tidak berarti bahwa kamu tidak dapat mencoba jawaban yang kamu tidak yakin. Tidak ada yang salah dengan mencobanya, selama kamu menjelaskan ketidakpastianmu kepada pewawancara sejak awal. Jawabannya mungkin seperti ini:

Saya harus jujur dan mengatakan bahwa ini bukan bidang yang saya kenal, meskipun saya sangat tertarik. Jika kamu mau, saya akan senang untuk mencoba mengatasi masalah ini, selama kamu tidak mengharapkan jawaban yang tepat.

Atau: Saya ingin menjawab pertanyaan itu, tetapi saya harus jujur di awal dan mengatakan bahwa ini bukan area yang terlalu saya kenal, meskipun saya sangat tertarik untuk meningkatkan pengetahuan saya tentangnya.

Mitos nomor 5: Orang yang tampan mendapatkan pekerjaan-

Saya kira jika pekerjaan itu untuk tipe femme fatale yang sangat cantik dalam sebuah film, maka ketampanan pasti akan membantu, tetapi untuk sebagian besar pekerjaan lain, penampilan Anda tidak sebesar yang dilihat banyak orang.

Seperti yang telah kita bahas, akan selalu ada pemberi kerja yang tidak berpengalaman yang akan mempekerjakan berdasarkan faktor dangkal, tetapi kebanyakan pemberi kerja lebih pintar dari itu.

Klaim bahwa orang yang tampan mendapatkan pekerjaan daripada orang yang berpenampilan biasa membuat asumsi yang sangat salah bahwa pemberi kerja membuat kebiasaan mengutamakan ketampanan seseorang sebelum kepentingan mata pencaharian mereka. Semua pengalaman saya telah mengajari saya sebaliknya. Sebagian besar bisnis mendapati diri mereka berada dalam lingkungan yang sangat kompetitif dan pemberi kerja sangat menyadari bahwa keputusan perekrutan yang buruk terbukti sangat mahal.

Ini tidak berarti bahwa penampilan dan kepribadian yang cerah bukanlah faktor penting dalam sebuah wawancara. Sangat penting bagi kamu untuk berpakaian dengan pantas dan mencoba yang terbaik untuk menunjukkan semua sifat ramahmu. Ketampanan memang dilebih-lebihkan dalam wawancara, tetapi penampilan yang pantas dan kepribadian yang ramah tidak.

Mitos nomor 6:Jika Anda menjawab pertanyaan dengan lebih baik daripada yang lain, Anda akan mendapatkan pekerjaan itu –

Mampu mengartikulasikan jawaban yang baik dalam sebuah wawancara sangat penting, dan kegagalan untuk melakukannya hampir pasti berarti kamu tidak mendapatkan pekerjaan itu. Namun, wawancara — seperti yang telah kita lihat — lebih dari sekadar memberikan jawaban yang bagus.

Mereka juga tentang meyakinkan pewawancara bahwa kamu akan menjadi orang yang baik untuk diajak bekerja sama. Dengan kata lain, tidak peduli seberapa bagus jawabanmu secara teknis, jika pewawancara tidak menyukai dirimu, tidak banyak kemungkinan kamu akan mendapatkan pekerjaan itu (kecuali jika bakat pada dirimu unik, sangat sulit ditemukan atau pewawancara. putus asa).

Jadi hindari memikirkan wawancara hanya dalam hal menjawab pertanyaan dengan benar. Wawancara juga tentang membangun hubungan dan kepercayaan, dan meskipun tidak ada metode yang aman dalam melakukan ini, ada hal-hal yang dapat Anda lakukan (dan hal-hal yang tidak boleh kamu lakukan) yang akan sangat membantu meningkatkan keterampilan kamu dalam semua ini- area penting wawancara.

Mitos tidak. 7: Anda harus mencoba memberikan jawaban yang sempurna –
Saya telah mendengar terlalu banyak orang tersandung kata-kata mereka, mengulangi diri mereka sendiri dan berbicara dalam lingkaran karena mereka mencoba untuk mengartikulasikan jawaban yang sempurna — atau apa yang menurut mereka merupakan jawaban yang sempurna.

Beberapa orang begitu terobsesi untuk memberikan jawaban yang sempurna sehingga mereka tidak berhenti sampai mereka menghasilkan apa yang menurut mereka merupakan tanggapan yang sempurna. Karena kita tidak pernah bisa sepenuhnya yakin dengan apa yang pewawancara ingin dengar, beberapa dari kita akan terus berbicara dengan harapan kita akan membahas semua dasar. Masalah dengan pendekatan ini adalah bahwa kita akhirnya berbicara terlalu banyak, menyebabkan pewawancara kehilangan konsentrasi yang, tentu saja, merupakan hal terakhir yang Anda butuhkan dalam sebuah wawancara.

Kenyataannya adalah bahwa dalam banyak kasus tidak ada yang namanya jawaban yang sempurna. Pelajarannya di sini adalah: sangat masuk akal untuk menerima jawaban bagus yang langsung ke intinya daripada berkelok-kelok di mana-mana mencari jawaban sempurna yang sulit dipahami.

Mitos nomor 8: Anda harus mengajukan pertanyaan untuk menunjukkan minat dan kecerdasan Anda -

Banyak orang yang diwawancarai memiliki keyakinan yang salah bahwa mereka harus mengajukan pertanyaan di akhir wawancara. Tampaknya ada kepercayaan umum di antara banyak orang yang diwawancarai bahwa hal ini membuat mereka terdengar lebih cerdas dan juga lebih tertarik pada pekerjaan tersebut.

Ini tidak benar. Mengajukan pertanyaan hanya untuk kepentingan melakukannya tidak akan meningkatkan peluangmu untuk mendapatkan pekerjaan.

Itu bahkan bisa membuat kamu terdengar sedikit membosankan, terutama jika kamu mengajukan pertanyaan tentang hal-hal yang sudah dibahas selama wawancara.

Ajukan pertanyaan hanya jika kamu memiliki pertanyaan yang tulus. Pertanyaan yang dapat diterima mencakup pertanyaan yang berkaitan langsung dengan pekerjaan yang kamu lamar, serta kondisi kerja dan kebijakan perusahaan tentang hal-hal seperti gaji, cuti, dan sebagainya. Pewawancara tidak pernah keberatan menjawab pertanyaan tentang hal-hal seperti itu, tetapi mereka keberatan menjawab pertanyaan yang mereka anggap tidak relevan.

Jika kamu tidak memiliki pertanyaan untuk ditanyakan, cukup katakan sesuatu seperti: ‘Terima kasih, tapi saya tidak punya pertanyaan. Kamu telah sangat teliti selama wawancara dan telah membahas semua hal penting terkait pekerjaan. 'Tidak ada salahnya menyertakan pujian kepada pewawancara tentang ketelitian dan profesionalisme mereka — asalkan tidak berlebihan atau terdengar seperti merendahkan.

Dua poin lebih lanjut perlu dibuat tentang mengajukan pertanyaan. Pertama, hindari menanyakan terlalu banyak pertanyaan. Secara keseluruhan, pewawancara tidak menyukai pembalikan peran. Kedua, jangan pernah mengajukan pertanyaan yang berpotensi memalukan. Ini bisa termasuk:

Pertanyaan yang berkaitan dengan insiden negatif; sesuatu yang tidak seharusnya berada dalam domain publik; Pertanyaan sulit yang mungkin membingungkan pewawancara.

Aturan praktisnya adalah: jika menurutmu suatu pertanyaan dapat menyebabkan rasa malu, berhati-hatilah dan hindari.

Mitos nomor 9: Santai dan jadilah dirimu sendiri-

Meskipun penting untuk santai dan menunjukkan sisi mu yang lebih baik, penting juga untuk memahami bahwa wawancara bukanlah keterlibatan sosial. Sebagian besar wawancara adalah acara yang sangat formal di mana perilaku yang tidak berbahaya dianggap tidak dapat diterima.

Singkatnya, menjadi dirimu sendiri yang biasa bisa berarti bencana (mungkin terdengar kontradiktif). Misalnya, jika menjadi diri sendiri berarti bersandar di kursimu, berpakaian agak lusuh dan membuat lelucon, kamu mungkin mendapati dirimu menghadiri banyak sekali wawancara.

Sementara pewawancara menyukai orang-orang yang santai, mereka juga memiliki ekspektasi yang pasti tentang perilaku apa yang pantas untuk sebuah wawancara dan kamu melanggar ekspektasi ini dengan resiko mu sendiri!

Mitos nomor 10: Pewawancara mencari kekurangan

Bahaya dengan mitos ini adalah bahwa hal itu dapat dengan mudah membuat orang yang diwawancarai mengambil sikap defensif, bahkan mungkin tidak percaya, selama wawancara.

Jika kamu yakin bahwa pewawancara dengan tekun mencari kekurangan didirimu, kemungkinan besar hal itu akan merusak upayamu untuk membangun hubungan dan kepercayaan yang sangat penting itu.

Ini juga dapat menghalangimu untuk terbuka dan memberikan jawaban yang sangat bagus. Yakinlah bahwa sebagian besar pewawancara tidak mempersiapkan pertanyaan wawancara mereka dengan tujuan untuk mengungkap kekurangamu. Pertanyaan sebagian besar disiapkan dengan maksud untuk memberikan pewawancara wawasan keseluruhan atau holistik tentang apa yang kamu tawarkan kepada perusahaan. Pewawancara yang baik memang akan mengungkap area di mana kamu tidak kuat, tetapi itu jauh dari pemikiran bahwa pewawancara sangat ingin mengungkap hanya kekurangamu.

Sangat penting untuk memperlakukan setiap pertanyaan sebagai kesempatan untuk unggul daripada dijaga secara tidak perlu. Hanya dengan menjawab pertanyaanmu dapat menunjukkan seberapa baik dirimu. Memperlakukan pertanyaan sebagai objek kecurigaan sama sekali tidak masuk akal.

Memahami mitos seputar wawancara memberi kamu awal yang baik untuk sukses. Ingat, wawancara tidak berbeda dengan upaya lain dalam hidup: semakin baik kamu memahami sifat dasarnya, semakin tinggi kemungkinan kamu akan berhasil menanganinya. Sebuah wawasan tentang mitos wawancara umum akan membekali dirimu dengan informasi yang kamu butuhkan, untuk mencegah kamu jatuh ke dalam perangkap yang mengecewakan itu.

Sama pentingnya, gambaran yang lebih jelas tentang sifat sebenarnya dari wawancara lebih baik menginformasikan sisa persiapan dirimu dan akan berkontribusi pada kepercayaan diri dan kinerjamu.

Ringkasan point-point penting

Orang terbaik untuk pekerjaan itu belum tentu memenangkannya — seringkali orang yang memberikan wawancara terbaik.

Wawancara lebih dari sekedar memberikan jawaban yang benar secara teknis. Mereka juga sangat banyak tentang membangun hubungan baik.

Tidak semua pewawancara tahu apa yang mereka lakukan; tugasmu adalah mengetahui bagaimana menangani pewawancara yang baik dan yang buruk.

Lebih baik jujur dan mengakui ketidaktahuan daripada mencoba berpura-pura tahu jawaban dan tampil sebagai tidak jujur dan kurang cerdas.

Orang-orang yang tampan memenangkan pekerjaan — mungkin di film-film Hollywood, tetapi secara keseluruhan, para pemberi kerja tertarik untuk mempekerjakan bakat daripada faktor-faktor yang dangkal.

Berusaha keras untuk memberikan jawaban yang sempurna bisa membawa mu ke dalam masalah. Lebih baik memberikan jawaban yang tepat dan langsung daripada mencari kesempurnaan; selain itu, seringkali tidak ada yang namanya jawaban yang sempurna.

Jangan mengajukan pertanyaan demi itu. Ajukan pertanyaan hanya jika kamu memiliki pertanyaan asli yang belum tercakup.

Wawancara adalah acara formal yang membutuhkan perilaku yang relatif formal.

Pewawancara akan mengharapkan ini dan mungkin bereaksi negatif jika mereka tidak melihatnya. Pewawancara tidak menghabiskan waktu mereka untuk mencari kekurangan pada dirimu. Mereka lebih tertarik untuk mendapatkan gambaran menyeluruh tentang siapa kamu. Hindari menjawab pertanyaan secara defensif. Jauh lebih baik melihat setiap pertanyaan sebagai peluang untuk menyoroti poin terbaik mu.

“Ketahuilah, ketika Anda benar-benar menginginkan kesuksesan, Anda tidak akan pernah menyerah. Tidak peduli seberapa buruk situasinya. " – Harry.

BAB 2
MEYAKINKAN MEREKA BAHWA ANDA TEPAT UNTUK PEKERJAAN ITU

Berhasil dalam wawancara tidak sesulit yang dipikirkan banyak orang. Dengan persiapan yang benar dan sedikit latihan, kebanyakan orang yang takut akan wawancara dapat belajar untuk menjadi yang terbaik. Hal penting yang perlu diperhatikan adalah bahwa melakukan wawancara dengan baik adalah proses yang dipelajari. Orang yang diwawancarai sangat efektif tidak dilahirkan dengan keterampilan wawancara; sebaliknya, mereka mengajari diri mereka sendiri apa yang harus dikatakan, bagaimana mengatakannya dan bagaimana berperilaku selama wawancara.

Kesalahan wawancara umum

Kita semua telah membuat kesalahan selama wawancara, dan sebagian besar dari kita telah keluar dari wawancara memikirkan semua hal hebat yang lupa kita sebutkan dan semua hal yang seharusnya tidak kita katakan. Tetapi hal terpenting tentang kesalahan adalah belajar darinya dan tidak mengulanginya. Berikut adalah beberapa kesalahan wawancara yang umum:

Gagal mengekspresikan diri dengan jelas. Seringkali, karena kecemasan dan ingin mengatakan sesuatu dengan sempurna, kita berusaha terlalu keras dan mengubah kalimat yang seharusnya sederhana menjadi kalimat yang tidak masuk akal. Bahasa sederhana selalu yang paling efektif. Hindari mencoba terdengar berpengetahuan dengan menggunakan jargon atau kalimat kompleks.

Tidak menyadari bahasa komponen seseorang. Banyak orang yang diwawancarai berhasil mengasingkan pewawancara karena mereka sedikit atau tidak memperhatikan bahasa komponen mereka. Bahasa komponen adalah komunikator yang sangat kuat, dan gagal menggunakannya secara efektif hampir pasti akan membuat kamu sangat tidak beruntung. Kontak mata, posisi duduk, dan ekspresi wajah adalah aspek yang sangat penting dalam wawancara, dan perlu dipikirkan sebelum wawancara.

Gagal mengendalikan saraf itu. Kadang-kadang orang membiarkan saraf mereka menjadi tidak terkendali sehingga mereka gagal membangun hubungan dan bahkan melupakan jawaban mereka. Merasa cemas sebelum dan selama wawancara adalah hal biasa. Nyatanya, sentuhan saraf bisa menjadi hal yang baik. Tapi tidak perlu menjadi korban saraf yang melemahkan. Saat kamu membaca buku ini, kamu secara bertahap akan belajar bagaimana mengurangi kecemasan mu.

Gagal memberikan contoh yang tepat. Gagal memberi contoh, atau memberi contoh yang tidak tepat, akan menyebabkan bencana. Sebelum wawancara, penting untuk memikirkan contoh yang relevan tentang apa yang telah kamu capai dan bagaimana kamu mewujudkan pencapaian tersebut. Mengatakan bahwa kamu mencapai sesuatu tanpa dapat mendukungnya dengan contoh spesifik hanya akan

membuat mu mendapatkan surat penolakan. Contoh kamu harus mudah dipahami, mengikuti urutan logis, dan relevan dengan kebutuhan pemberi kerja. Semua ini tidak terjadi tanpa persiapan.

Berusaha terlalu keras untuk menyenangkan pewawancara. Sementara membangun hubungan dan kepercayaan selama wawancara sangat penting, beberapa pewawancara menghargai orang yang diwawancarai secara berlebihan dengan perilaku mereka. Perilaku patuh biasanya dilihat sebagai bentuk penipuan dan tidak terlalu berpengaruh — faktanya, perilaku tersebut dapat merusak upayamu untuk menciptakan kepercayaan.

Tidak ada yang salah denganmu. Kamu mungkin telah melakukan setidaknya beberapa kesalahan yang tercantum di atas. Sangat penting untuk menyadari bahwa membuat kesalahan seperti itu adalah hal biasa. Dengan kata lain, tidak ada yang salah denganmu. Dalam sebagian besar kasus, kinerja yang buruk dalam wawancara terjadi karena sifat dasar wawancara — penyebabnya adalah proses wawancara.

Jadi kesadaran akan sifat dasar wawancara adalah langkah pertama dalam proses langkah demi langkah yang dapat digunakan untuk meningkatkan kinerja kamu secara signifikan. Tempat yang bagus untuk memulai adalah dengan bertanya: 'Apa yang diperlukan untuk meyakinkan pewawancara bahwa kamu orang terbaik untuk pekerjaan itu?' Jawaban atas pertanyaan ini dapat diringkas dalam empat bagian:
persiapan yang benar;
mengetahui hal-hal yang penting bagi pewawancara;
melatih jawaban Anda;
ketekunan.

Persiapan yang benar -

Seberapa baik kamu melakukan wawancara akan sangat bergantung pada seberapa baik kamu mempersiapkannya. Kegagalan untuk mempersiapkan dengan benar hampir pasti berarti kamu tidak akan melakukan yang terbaik. Dalam beberapa kasus, itu berarti kinerja yang cukup buruk, yang dapat berkontribusi pada penurunan kepercayaan diri Anda.

Sekalipun kamu cukup beruntung menjadi kandidat yang disukai, dan hampir pasti memenangkan posisi hanya dengan muncul, kamu tetap harus meluangkan waktu untuk bersiap karena semakin baik kinerja mu, semakin besar kemungkinan kamu akan bernegosiasi dengan lebih baik. gaji dan seringkali perbedaan uang bisa sangat besar.

Kita semua pernah mendengar orang menyombongkan diri bahwa mereka tidak pernah siap untuk wawancara dalam hidup mereka dan melakukannya dengan baik. Meskipun sombong ini mungkin bukan omong kosong, pemeriksaan lebih dekat biasanya akan mengungkapkan bahwa orang-orang ini adalah:
Beruntung — yaitu, di tempat yang tepat pada waktu yang tepat;
Terhubung dengan baik;
Bekerja di pasar tenaga kerja yang menguntungkan dimana ada yang besar

Permintaan karyawan ditambah dengan pasokan rendah;
Melamar pekerjaan dengan baik dalam zona nyaman mereka — yaitu, tidak Meregangkan diri untuk meningkatkan posisi mereka; atau
Melamar pekerjaan secara internal dan bersaing terutama dengan kandidat eksternal.

Kasus untuk persiapan

Argumen untuk persiapan wawancara menjadi menarik ketika kamu memikirkan sifat dasar wawancara.

Kamu tidak hanya diharapkan untuk menjual dirimu dalam lingkungan yang kompetitif, tetapi kamu juga diharapkan untuk memampatkan informasi yang besar dan seringkali kompleks menjadi javaban yang rapi dan sangat jelas yang menghindari konotasi negatif dan berisi informasi yang ingin didengar oleh pewawancara.

Tidak heran jika tingkat stres orang meningkat. Tapi itu tidak berakhir di situ. Ada tiga alasan tambahan yang membuat persiapan wawancara menjadi lebih menarik:

Wawancara adalah peristiwa yang jarang terjadi, sehingga membuat mereka asing dan canggung.

Banyak orang merasa sangat sulit untuk menjual diri saat wawancara karena mereka telah dikondisikan oleh keluarga dan masyarakat untuk tidak meniup terompet mereka sendiri. Membuat pernyataan sederhana seperti 'Saya sangat ahli dalam menjual xyz' bisa menjadi kendala yang cukup untuk diatasi.

Dalam sebagian besar wawancara, menjadi yang kedua tidaklah cukup. Ini bukan hanya masalah kinerja yang baik; ini juga masalah mengalahkan orang lain. Tidak terbayangkan bahwa Anda akan gagal mempersiapkan acara yang jarang terjadi, kompetitif dan membutuhkan perilaku yang tidak biasa digunakan. Namun, itulah tepatnya yang dilakukan orang ketika mereka memasuki wawancara tanpa persiapan.

Apa persiapan yang salah? Persiapan yang salah adalah persiapan apa pun yang tidak akan mengoptimalkan performamu saat wawancara. Jawaban umum hafalan yang telah disiapkan orang lain memiliki nilai yang terbatas.

Paling banter, mereka dapat memberimu wawasan tentang apa yang mungkin merupakan jawaban yang baik; paling buruk, mereka hanya menyesatkanmu. Penting untuk dipahami bahwa, dalam sebagian besar kasus, tidak ada yang namanya jawaban tunggal untuk sebuah pertanyaan. Apa yang mungkin merupakan jawaban yang bagus untuk satu pemberi kerja mungkin dianggap cukup biasa oleh pemberi kerja lainnya.

Salah satu hal terburuk yang dapat kamu lakukan adalah mempelajari tanggapan orang lain secara langsung dan mengulanginya dalam sebuah wawancara. Mengulangi apa yang disebut sebagai jawaban bagus orang lain dapat membuatmu terdengar tidak jujur dan membuat kamu terlihat sedikit konyol saat ditanya pertanyaan lanjutan yang menyelidik. Lebih masuk akal jika kamu menyiapkan jawabanmu sendiri.

Keuntungan dari persiapan-

Meluangkan waktu untuk mempersiapkan wawancara dengan benar akan:

- Tingkatkan tingkat kepercayaan diri kamu;
- Membantumu dalam menjawab pertanyaan secara ringkas, bukan mengambil selamanya untuk membuat poin sederhana;
- Membantumu mengetahui apa yang harus dikatakan dan bagaimana mengatakannya; Membantumu dalam menangani pertanyaan sulit;
- Membantumu menghindari mengatakan hal-hal yang akan menimbulkan kesan negatif;
- Tingkatkan keterampilan membangun hubungan Anda.

Mengetahui hal-hal yang penting bagi pewawancara-

Salah satu kunci untuk mengetahui apa yang harus disiapkan terletak pada pemahaman kebutuhan pewawancara. Setelah kamu mengetahui hal-hal yang penting bagi pewawancara, persiapan wawancara tiba-tiba menjadi jauh lebih jelas dan lebih mudah diatur.

Sebagian besar pewawancara — disadari atau tidak — ingin mendengar tiga hal dari kamu. Faktanya, hampir semua pertanyaan wawancara yang baik bermuara pada tiga pertanyaan umum utama ini:

Bisakah kamu melakukan pekerjaan itu? Dengan kata lain, apakah kamu memiliki keterampilan, pengetahuan, pengalaman, atau potensi untuk bekerja dengan baik dalam pekerjaan itu? Sebagian besar pewawancara akan menghabiskan sebagian besar wawancaranya untuk menyelidiki kamu tentang pertanyaan ini. Mereka ingin tahu apa yang telah kamu lakukan, bagaimana kamu melakukannya, dan apa hasilnya. Jika kamu belum melaksanakan tugas tertentu, mereka akan mencoba memastikan potensi kamu untuk melakukan pekerjaan tersebut.

Apakah kamu tipe orang yang bisa mereka ajak kerja sama? Cara lain untuk menyatakan pertanyaan ini adalah: Apakah kamu akan cocok dengan budaya organisasi yang ada? Atau, dalam kasus organisasi kecil: Apakah kamu akan cocok dengan bos? Meskipun pewawancara biasanya menghabiskan lebih sedikit waktu untuk pertanyaan ini, ini adalah pertanyaan yang sangat penting karena tidak ada yang mau bekerja dengan seseorang yang tidak mereka sukai, bahkan jika mereka bisa melakukan pekerjaan itu.

Seberapa termotivasi kamu? Dengan kata lain, tingkat energi dan dorongan apa yang kamu bawa ke posisi itu? kamu bahkan mungkin tidak ditanyai tentang tingkat motivasi mu, tetapi kamu gagal menjawabnya dengan risiko kamu sendiri. Seperti yang kita semua tahu, karyawan yang bermotivasi tinggi sangat dicari oleh pemberi kerja — dengan alasan yang bagus.

Ada dua manfaat penting untuk mengetahui bahwa pewawancara sangat tertarik dengan tiga pertanyaan umum ini, dan bahwa sebagian besar pertanyaan yang dapat mereka ajukan termasuk dalam satu atau lebih kategori ini. Pertama, ini memandu dirimu dalam mempersiapkan jawabanmu (sebagian besar buku ini didasarkan pada menjawab tiga pertanyaan kunci ini). Daripada menghabiskan banyak waktu untuk mengarungi pertanyaan yang dipilih secara acak dengan harapan kamu

akan menyiapkan jawaban yang benar, pemahaman tentang pentingnya tiga pertanyaan umum utama memberikan arahan dan platform untuk persiapanmu. Singkatnya, kamu dapat merencanakan persiapan dirimu seputar masalah-masalah berikut:

Keterampilan, pengetahuan, dan pengalamanmu — dapatkah kamu melakukan pekerjaan itu?

Atribut pribadi mu — apakah kamu tipe orang yang dapat mereka ajak bekerja sama?

Tingkat motivasi mu.

Kedua, ini memberikan cara yang berguna untuk menangani pertanyaan pada wawancara yang sebenarnya. Dengan menyortir pertanyaan wawancara ke dalam satu atau lebih dari tiga kategori pertanyaan umum, jawabanmu akan mendapatkan struktur tambahan dan arah yang lebih jelas hanya karena kamu tahu apa tujuan yang mendasari pertanyaan tersebut.

Dengan mempelajari cara mengenali maksud sebenarnya dari sebuah pertanyaan, kamu meminimalkan kemungkinan kamu memberikan jawaban yang salah dan/atau membingungkan.

Praktek

Aspek ketiga untuk meyakinkan pewawancara bahwa kamu orang terbaik untuk pekerjaan itu adalah latihan. Sayangnya, tidak ada jalan pintas untuk mengembangkan keterampilan wawancara yang hebat.

Setelah kamu menyiapkan jawaban, kamu perlu duduk dan mempraktikkannya sebanyak mungkin. Semakin banyak kamu berlatih, kamu akan menjadi semakin baik. Seperti kata pepatah lama, 'kesuksesan adalah satu bagian dari bakat dan sembilan bagian ketekunan'. Bagaimana caramu berlatih itu terserah kamu. Lakukan di depan cermin, duduk di sofa, mondar-mandir di kamar atau kamu juga dapat mempraktekkannya saat mengendarai kendaraan.

Berlatih di depan atasan Anda!

Mempraktikkan jawabanmu dengan lantang Penting untuk mempraktikkan jawabanmu dengan keras, daripada hanya melatihnya secara mental. Itu karena otak manusia membedakan antara berbicara dan berpikir dan kamu perlu merangsang bagian otak yang berbicara. Memikirkan jawaban kamu saat wawancara tidak akan membawa kamu kemana-mana, kecuali pewawancara adalah pembaca pikiran.

Dapatkan beberapa umpan balik

Idealnya, kamu harus berlatih pada wawancara nyata. Semakin banyak wawancara yang kamu hadiri, semakin baik — bahkan jika kamu harus menghadiri wawancara untuk pekerjaan yang sebenarnya tidak kamu minati. Setelah wawancara — dengan asumsi kamu bukan kandidat yang menang — telepon kembali pewawancara dan minta umpan balik pada kinerjamu.

Beberapa pewawancara dengan senang hati memberikan umpan balik ini; namun, banyak yang memilih untuk tidak melakukannya karena mereka menganggapnya mengancam dan membuang-buang waktu mereka. Orang-orang ini akan menghindari kamu sama sekali atau memberimu umpan balik yang dipermudah sehingga hampir tidak berguna.

Dalam beberapa kasus, kamu mungkin tidak dapat menyelesaikan masalah ini; namun, kamu dapat meningkatkan peluang untuk mendapatkan umpan balik yang jujur dengan membuat pewawancara merasa senyaman mungkin.

Kamu dapat melakukan ini dengan a) meyakinkan mereka bahwa kamu hanya menginginkan lima menit dari waktu mereka; dan b) memberi tahu mereka bahwa satu-satunya alasan kamu mencari umpan balik adalah untuk meningkatkan kinerja wawancara di masa mendatang.

Wawancara tiruan

Jika kamu tidak bisa mengikuti wawancara sebanyak yang kamu inginkan, sebaiknya siapkan wawancara tiruan dengan seseorang yang dapat kamu ajak kerja sama. Semakin dekat kamu dapat mensimulasikan situasi kehidupan nyata, semakin banyak manfaat yang akan kamu peroleh.

Cara efektif untuk melakukan wawancara tiruan adalah dengan berperan dan bertahan selama wawancara. Tidak ada gangguan, tidak ada basa-basi dan terutama tidak memulai lagi. Jika memungkinkan, hindari memberikan pertanyaan kepada pembantumu, biarkan mereka mengajukan pertanyaan mereka sendiri.

Jika pembantumu tidak dalam posisi untuk melakukan ini, beri mereka banyak pertanyaan dan minta mereka untuk memilih yang mereka inginkan. Yang penting bagi kamu adalah membiasakan diri menjawab pertanyaan yang tidak terduga.

Selain itu, jika kamu merasa penolongmu dapat memberikan umpan balik yang jujur tentang kinerjamu, jangan segan untuk bertanya. kamu tidak pernah tahu apa yang mungkin kamu pelajari. Seringkali hal-hal kecil yang membuat perbedaan besar. Tetapi berhati-hatilah untuk mendapatkan umpan balik yang terlalu positif. Kemungkinannya adalah bahwa penolong kamu adalah seorang teman, dan teman-teman kamu terkenal karena menghindari hal-hal negatif.

Ketekunan

Hal terburuk yang dapat kamu lakukan saat bersiap untuk meningkatkan kinerja wawancara kamu adalah menyerah karena semuanya tampak terlalu sulit. Orang yang berhenti merokok selalu tidak bisa kemana-mana. Mereka pasti tidak mendapatkan pekerjaan yang bagus dan membangun karier yang hebat. Di sisi lain, orang yang tekun sangat sering mendapatkan wawasan berharga hanya karena mereka memiliki stamina untuk bertahan.

Orang yang paling kita kagumi sering kali adalah mereka yang menghadapi rintangan yang tampaknya tidak dapat diatasi namun alih-alih berhenti, secara diam-

diam memutuskan untuk mengatasinya. Di sisi lain, orang yang paling tidak kita hormati adalah mereka yang selamanya memulai sesuatu tanpa menyelesaikannya.

Mereka cenderung menjadi orang yang sama yang membuat klaim yang muluk-muluk tetapi akhirnya memberikan sedikit atau tidak sama sekali. Salah satu karakteristik umum yang cenderung dimiliki oleh orang yang berhenti merokok secara kronis adalah harga diri yang rendah sehingga mereka tidak terlalu percaya pada diri mereka sendiri. Dan jika kamu tidak percaya pada diri sendiri, orang lain biasanya juga tidak percaya pada mu — bukan tempat yang tepat ketika kamu mencoba meyakinkan pewawancara untuk percaya pada kemampuan mu.

Ini adalah orang-orang yang sering terdengar mengatakan hal-hal seperti: 'Itu terlalu sulit', 'Saya tidak bisa mempelajarinya', 'Apa yang akan dipikirkan orang lain', dll. Mereka juga cenderung menjadi orang-orang yang selalu mengeluh tentang berbagai hal. tetapi sepertinya tidak pernah mengambil tindakan apa pun untuk memperbaikinya karena selalu ada alasan. Kamu tidak harus menjadi orang yang gampang menyerah atau terbebani dengan harga diri rendah untuk menyerah dalam mengerjakan keterampilan wawancaramu, mungkin ada sejumlah alasan lain .

Namun, jika kamu membaca buku ini, ada kemungkinan besar bahwa meningkatkan keterampilan wawancaramu adalah prioritas penting dalam hidup kamu, dan oleh karena itu tidak boleh dibiarkan begitu saja. Jika kamu merasa kamu salah satu dari orang-orang yang berdiri di tebing untuk berhenti, berikut adalah latihan kecil yang dapat membantu mu mengambil satu atau dua langkah mundur dari tepi.

Aktivitas yang disarankan: Pemrograman neurolinguistik

Berdasarkan pemrograman neurolinguistik (NLP), latihan ini dirancang untuk memengaruhi perasaanmu. Orang sering berhenti karena mereka mengasosiasikan perasaan negatif dengan apa yang mereka lakukan. Orang yang gigih memiliki kekuatan untuk merasa nyaman dengan tindakan mereka tidak peduli seberapa membosankan atau tidak konstruktifnya tindakan tersebut bagi orang lain.

Jika kamu dapat membuat dirimu merasa nyaman dengan proses peningkatan keterampilan wawancaramu, maka kemungkinan besar berhenti akan menjadi hal terakhir yang kamu pikirkan. Lain kali kamu ingin berhenti, kamu mungkin ingin mencari tempat yang tenang dan mengambil langkah-langkah berikut:

Pejamkan matamu dan bayangkan dirimu tampil sangat baik dalam sebuah wawancara. Luangkan waktu mu untuk melihat gambar ini sedetail mungkin. Bayangkan wajah pewawancara yang antusias, perhatikan betapa penuh perhatian mereka dan betapa terkesannya mereka dengan tanggapanmu. Benamkan dirimu dalam pengalaman itu. Perhatikan detailnya, termasuk suara, bau, warna, suhu, dan sebagainya. Yang terpenting, tangkap perasaan sukses. Jangan menahan diri. Semakin baik perasaanmu, semakin kuat latihannya.

Terus ulangi latihan ini sampai kamu mendapatkan perasaan nyaman dan gembira itu. Kamu mungkin bisa membangkitkan kegembiraan yang lebih besar dengan membayangkan dirimu dalam pekerjaan barumu. Bayangkan betapa

menyenangkan rasanya memenangkan pekerjaan yang hebat. Bayangkan mendapatkan semua panggilan telepon penting yang memberi tahu kamu tentang kesuksesanmu. Bayangkan dirimu dalam posisi melakukan semua hal yang kamu impikan. Kunci dari latihan ini adalah untuk membangkitkan perasaan luar biasa yang menyertai kesuksesan dalam sebuah wawancara. Batasan mu hanyalah imajinasi mu.
Setelah kamu mendapatkan perasaan itu, langkah selanjutnya adalah menciptakannya kembali saat kamu membutuhkannya — dengan kata lain, saat kamu ingin berhenti. Cara efektif untuk menciptakan kembali perasaan gembira adalah dengan memasang apa yang disebut NLP sebagai jangkar. Jangkar adalah rangsangan yang memicu perasaan yang diinginkan saat kamu menginginkannya. Penanda bisa menjadi sesuatu yang kamu lakukan, katakan atau bayangkan. Penanda tindakan biasanya bekerja paling baik. Misalnya, kamu mungkin menyilangkan jari atau melompat ke udara atau menarik telinga Anda. Tidak masalah apa itu, selama kamu bisa melakukannya dengan mudah saat kamu mau dan memicu perasaan yang diinginkan. Setiap kali kamu terkena momok berhenti, gunakan jangkarmu dan biarkan kemampuanmu memengaruhi perasaan melakukan sisanya.

Ringkasan Point Utama

Karena sifatnya, wawancara pada dasarnya menantang. Membuat kesalahan saat wawancara adalah sesuatu yang dilakukan setiap orang. Kabar baiknya adalah kita dapat mengatasi kesalahan kita dengan persiapan, latihan, dan ketekunan yang benar.

Waspadai persiapan yang salah. Hindari menghafal jawaban orang lain. Selalu persiapkan sendiri.

Mengetahui apa yang ingin didengar pemberi kerja saat wawancara merupakan awal yang baik untuk mempersiapkan jawabanmu sendiri dan menyederhanakan persiapan wawancara. Apa yang ingin didengar sebagian besar pengusaha dapat diwakili oleh tiga pertanyaan kunci:
Bisakah kamu melakukan pekerjaan itu?
Apakah kamu tipe orang yang bisa mereka ajak kerja sama?
Seberapa termotivasi kamu?
Lakukan sebanyak mungkin latihan dan selalu minta umpan balik yang jujur.
Ketekunan adalah segalanya.
Singkirkan semua pikiran untuk berhenti dengan mengajar diri sendiri untuk mengasosiasikan perasaan senang yang kuat dengan meningkatkan keterampilan wawancara Anda.

“Saya tidak menyesali hal-hal yang telah saya lakukan, saya menyesali hal-hal yang tidak saya lakukan ketika saya memiliki kesempatan.” - Harry.

“Tantangan itulah yang membuat hidup menarik dan mengatasinya itulah yang membuat hidup bermakna.” - Harry.

"Untuk berhasil, keinginan Anda untuk sukses harus lebih besar daripada ketakutan Anda akan kegagalan." – Harry

BAB 3
DAPATKAH ANDA MELAKUKAN PEKERJAAN ITU?

Sebelum pemberi kerja memutuskan untuk memberi seseorang pekerjaan, mereka perlu diyakinkan bahwa orang tersebut dapat melakukan pekerjaan dengan benar atau mempelajarinya dengan cepat. Oleh karena itu, tidak mengherankan mengetahui bahwa pertanyaan 'Bisakah kamu melakukan pekerjaan itu?' Adalah yang paling umum. Merekalah yang paling banyak dipersiapkan oleh orang-orang.

Pertanyaan 'Bisakah kamu melakukan pekerjaan itu?' Adalah pertanyaan yang secara langsung atau tidak langsung berusaha memastikan kemampuanmu untuk melakukan tugas yang melekat dalam pekerjaan. Itu mencakup pertanyaan-pertanyaan yang berusaha untuk memperjelas:

1. *Keterampilan;*
2. *Pengetahuan;*
3. *Pengalaman;*
4. *Prestasi utama;*
5. *Performa potensial.*

Contoh pertanyaan 'Bisakah kamu melakukan pekerjaan itu?' Meliputi:

- Dapatkah kamu memberi kami contoh saat kamu harus mengomunikasikan sesuatu yang kompleks dan kontroversial? Bagaimana kamu melakukannya?
- Ceritakan tentang salah satu pencapaian utamamu?
- Klien yang marah-marah dan memberimu ledakan amarahnya melalui telepon. Bagaimana kamu menanganinya?
- Menurutmu apa yang dapat kamu bawa ke posisi ini?
- Dapatkah kamu memberi kami contoh proyek yang harus kamu rencanakan dan atur?
- Langkah apa yang kamu ambil?
- Bagaimana kamu menggambarkan dirimu?
- (Sekilas, ini mungkin tidak mengejutkan kamu sebagai pertanyaan 'Bisakah kamu melakukan pekerjaan itu?', Tetapi orang yang diwawancarai yang efektif selalu mencari cara untuk menyoroti keterampilan mereka.)
- Menurutmu, apa yang membuat manajer menjadi efektif?
- Mengapa kami harus mempekerjakan Anda? Apa yang kamu anggap sebagai kekuatan terbesar kamu?

Tugas terpenting dalam pekerjaanmu adalah menjaga x, y, dan z. Beritahu kami bagaimana kamu berniat melakukannya.

Tiga jenis pertanyaan 'Bisakah kamu melakukan pekerjaan itu?' -

Kecuali dirimu sedang diwawancarai untuk pekerjaan yang hampir sama dengan yang pernah kamu miliki, kemungkinan kamu akan ditanyai tiga jenis pertanyaan 'Bisakah kamu melakukan pekerjaan itu?'. Ini adalah:

- Pertanyaan tentang tugas yang telah kamu lakukan sebelumnya.
- Pertanyaan tentang tugas yang belum kamu lakukan tetapi keterampilannya sudah kamu kuasai.
- Pertanyaan tentang tugas yang sama sekali baru bagi dirimu.
- *Mencari tahu sebanyak mungkin tentang pekerjaan itu*
- Hal pertama yang perlu kamu lakukan adalah melihat dengan saksama tugas dan persyaratan pekerjaan yang kamu lamar. Tugas dan persyaratan inilah yang akan menjadi dasar jawaban kamu. Ada beberapa cara untuk mengumpulkan informasi semacam ini:
- Meneliti iklan pekerjaan;
- Mengakses pernyataan tugas — jika ada;
- Menghubungi bos atau agen perekrutan untuk mengklarifikasi tanggung jawab utama pekerjaan.

Dalam dunia yang ideal, kamu akan memiliki akses ke iklan pekerjaan yang terperinci, pernyataan tugas terbaru, dan pemberi kerja yang dengan senang hati mendiskusikan tanggung jawab utama pekerjaan tersebut. Sayangnya, seringkali kenyataannya adalah bahwa iklan pekerjaan ditulis dengan kata-kata yang tipis, pernyataan tugas tidak ada dan pemberi kerja tidak punya waktu untuk membalas telepon Anda.

Namun, sangat penting bagi dirimu untuk mencari tahu sebanyak mungkin tentang pekerjaan itu sebelum duduk dan memikirkan jawabanmu. Sumber informasi terbaik adalah pemberi kerja atau agen perekrutan.

Iklan lowongan dan pernyataan tugas berguna (terkadang hanya itu yang akan kamu miliki); namun, pernyataan tugas sering kali kedaluwarsa dan iklan lowongan mungkin kekurangan informasi yang memadai.

Berbicara dengan orang yang tepat dapat memberimu wawasan yang seringkali tidak dapat diambil dari kata-kata tertulis. Kamu mungkin menemukan, misalnya, bahwa posisi yang kamu lamar dikosongkan karena petahanan sebelumnya memiliki keterampilan komunikasi interpersonal yang buruk dan menjadi agresif ketika ada yang menyatakan pendapat berbeda.

Dalam kasus seperti itu, kemungkinan besar pemberi kerja akan mencari pengganti dengan komunikasi interpersonal yang sangat baik dan keterampilan pemain tim. Kamu akan memiliki peluang yang jauh lebih baik untuk memenangkan pekerjaan jika kamu telah mengakses informasi ini sebelum wawancara dan meluangkan waktu untuk menyiapkan jawabanmu.

Berbicara dengan bos untuk mengetahui lebih lanjut-

Jika kamu dapat berbicara dengan pemberi kerja, pastikan kamu telah mempersiapkan pertanyaanmu. Hal terakhir yang ingin kamu lakukan adalah membuang waktu mereka dengan tersandung pada pertanyaan-pertanyaan yang tidak dipikirkan dengan matang.

Jika bos tidak membalas teleponmu, jangan menyerah. Seringkali orang yang menjawab telepon dapat menjadi sumber informasi yang sangat berharga — terutama di perusahaan kecil dan menengah.

Ada kemungkinan besar mereka tahu banyak tentang posisi tersebut, atau mereka mungkin mengenal orang lain yang tahu dan bersedia berbicara dengan dirimu. Berikut adalah beberapa aturan berguna saat berbicara dengan pemberi kerja sebelum wawancara:

Hindari obrolan ringan dan langsung ke intinya. Obrolan ringan akan dipandang sebagai menyedot — yang, tentu saja, memang demikian!

Hindari mengajukan terlalu banyak pertanyaan — cukup tanyakan yang penting, kecuali jika pemberi kerja telah menjelaskan bahwa mereka punya banyak waktu luang dan bersedia untuk berbicara denganmu.

Jangan pernah mengajukan pertanyaan remeh — pertanyaan yang dapat dijawab dari iklan atau pelamar yang baik diharapkan mengetahui jawabannya.

Jika perlu, berikan alasan singkat mengapa kamu mengajukan pertanyaan— pemberi kerja mungkin tidak memahami pentingnya pertanyaan dan dapat menarik kesimpulan yang salah.

Ucapkan terima kasih atas waktunya dan beri tahu mereka bahwa kamu menantikan wawancara.

Sepatah kata singkat tentang pernyataan tugas

Pernyataan tugas hanyalah ringkasan dari tugas utama suatu pekerjaan. Meskipun mereka adalah sumber informasi yang bagus, mereka mungkin sudah ketinggalan zaman. Jadi, jika kamu pernah dikirimi, berusahalah untuk mencari tahu apakah informasinya masih valid.

Memeriksa pernyataan tugas dapat mewakili peluang besar untuk menghubungi pemberi kerja dan mengajukan beberapa pertanyaan. Sayangnya, pernyataan tugas biasanya disimpan oleh organisasi besar. Perusahaan yang lebih kecil umumnya kekurangan sumber daya untuk menuliskannya.

Mengumpulkan informasi dari iklan pekerjaan

Saat kamu mengamati iklan pekerjaan itu, buatlah daftar semua tugas/persyaratan yang terkait dengan posisi tersebut. Idenya adalah mencoba membaca yang tersirat sebanyak mungkin. Semakin banyak tugas dan persyaratan yang kamu buat, semakin menyeluruh persiapan untuk dirimu, yang akan mengurangi kemungkinan kamu tidak siap saat wawancara.

Empat langkah untuk sukses mewawancarai dirancang untuk menangkap semua informasi relevan yang Anda butuhkan untuk menyusun jawaban wawancara dalam kerangka yang mudah dikelola. Metode ini memiliki empat kolom, dengan tajuk yang ditunjukkan di bawah ini pada Tabel 3.1.

Tabel 3.1 Empat langkah untuk sukses mewawancarai

Langkah 1	Langkah 2	Langkah 3	Langkah 4
Tugas/persyaratan posisi yang saya lamar.	Apa yang sudah saya lakukan yang berkaitan langsung dengan tugas yang tercantum di langkah 1, termasuk mengatasi kendala	Konteks saat ini atau masa lalu	Hasil - organisasi dan pribadi.

Dengan mengisi setiap kolom di tabel, kamu secara efektif mengumpulkan semua informasi yang kamu perlukan untuk menjawab berbagai pertanyaan. Yang terpenting, ini adalah informasimu yang relevan, bukan informasi yang dikumpulkan dari jawaban orang lain yang telah kamu baca di tempat lain.

Setelah kamu mendapatkan informasi yang diperlukan, langkah kamu selanjutnya adalah menggabungkannya untuk menjawab berbagai pertanyaan wawancara dan kemudian mempraktikkan jawabanmu.

Pertanyaan perilaku

Salah satu keuntungan utama dari metode empat langkah ini adalah metode ini cocok untuk menangani teknik pertanyaan populer yang biasa disebut sebagai pertanyaan perilaku.

Kamu dapat mengenali salah satu pertanyaan ini setiap kali pewawancara meminta contoh spesifik untuk mendukung klaim yang kamu buat, termasuk langkah-langkah yang kamu ambil dan hambatan yang kamu hadapi.

Pertanyaan perilaku dirancang untuk mengungkap tindakan (perilaku) di balik hasil atau tugas, dan tidak dapat berhasil dijawab tanpa menyiapkan kolom ketiga.

Jika kamu lulusan atau pendatang baru di dunia kerja, masih ada kemungkinan besar kamu akan ditanyai tentang pertanyaan perilaku; namun, cakupannya akan terbatas. Alih-alih menanyakan pengalaman terkait pekerjaan, pewawancara akan meminta insiden terkait studi atau kehidupan.

Misalnya, pewawancara mungkin ingin mengetahui seberapa baik kamu berfungsi dalam sebuah tim, jadi mungkin bertanya kepada dirimu tentang terakhir kali kamu harus menyelesaikan tugas dengan sekelompok siswa. Prinsip yang sama berlaku untuk keterampilan komunikasi, perencanaan dan pengorganisasian, resolusi konflik, kemampuan kamu untuk mengatasi perubahan, dan sebagainya.

Menggunakan empat langkah-

Setelah kamu mendapatkan informasi sebanyak mungkin tentang pekerjaan itu, kamu perlu mulai memikirkan tentang mempersiapkan jawaban dirimu terkait tugas yang telah kamu lakukan sebelumnya. Yang perlu kamu lakukan hanyalah menceritakan tindakan dan pencapaian kamu sebelumnya dan menautkannya ke pekerjaan baru.

Tetapi berhati-hatilah untuk tidak menerima wawancara ini begitu saja. Terlalu mudah untuk jatuh ke dalam perangkap tidak mempersiapkan karena kamu pikir pertanyaannya akan mudah.

Namun, hanya karena kamu telah melakukan tugas yang sama tidak berarti kamu akan dapat mengartikulasikan detail dari apa yang kamu lakukan dan bagaimana kamu melakukannya. Ada perbedaan besar antara melakukan sesuatu dan benar-benar harus membicarakannya dengan cara yang ringkas dan koheren.

Langkah pertamamu adalah memilih semua tugas/persyaratan pekerjaan baru yang telah kamu lakukan sebelumnya dan menceritakan kembali tindakan dan pencapaianmu di masa lalu dengan cara yang akan memudahkan pembuatan jawaban yang efektif.

Gunakan Tabel 3.1 untuk menangkap semua informasi yang kamu perlukan, termasuk apa yang kamu lakukan, bagaimana kamu melakukannya, konteks di mana kamu melakukannya, dan hasilnya. Penjelasan lebih rinci tentang setiap langkah, termasuk apa yang harus disertakan dan tidak disertakan di setiap kolom, berikut.

Langkah 1: Tugas atau persyaratan

Cantumkan tugas dan persyaratan pekerjaan yang kamu lamar di kolom pertama.

Langkah 2: Apa yang kamu lakukan dan bagaimana kamu melakukannya -

Kolom kedua (langkah 2) berisi inti dari jawabanmu, termasuk kendala yang kamu atasi untuk memenuhi tugas atau persyaratan yang tercantum di langkah 1. Saat mengisi kolom ini, hindari menulis jawaban yang berskala luas atau umum, meskipun ini mungkin tidak selalu menjadi mungkin.

Idenya adalah untuk membagi tugas atau persyaratan yang tercantum pada langkah 1 menjadi tugas atau komponen utamanya. Akan membantu jika kamu bertanya pada diri sendiri pertanyaan berikut: Untuk menyelesaikan tugas atau persyaratan pada langkah 1, tindakan individu apa yang saya ambil, termasuk tindakan apa pun yang saya lakukan untuk mengatasi hambatan? Kemudian daftar ini dalam urutan logis.

Hindari terburu-buru melalui langkah ini, terutama jika kamu sudah lama tidak melakukan tugas tertentu. Ide yang bagus adalah menulis semua hal yang dapat kamu pikirkan dan kemudian kurangi daftar tersebut menjadi poin-poin utama. Sertakan contoh spesifik.

Berhati-hatilah agar tidak terlalu merinci saat mengisi kolom kedua. Melakukannya dapat secara tidak sengaja menghasilkan jawaban yang berisi terlalu banyak detail. Mengingat bahwa banyak orang yang diwawancarai merasa mereka harus memamerkan pengetahuan yang diperoleh dengan susah payah, langkah 2 mudah untuk berlebihan.

Namun, dalam sebagian besar kasus, kamu tidak diharuskan untuk mencakup setiap kemungkinan saat menjawab pertanyaan. Usahakan untuk tidak berbicara lebih lama dari yang seharusnya, sehingga membosankan pewawancara. Kebanyakan pewawancara mampu menarik kesimpulan yang masuk akal dari poin-poin utama dalam jawaban Anda.

Jika mereka menginginkan informasi lebih lanjut, mereka akan memintanya. Jika kamu memang memiliki banyak informasi bagus yang benar-benar kamu rasa tidak dapat ditinggalkan, lanjutkan dan daftarkan mereka di kolom kedua, tetapi selektiflah tentang apa yang kamu gunakan saat wawancara.

Pilih hanya poin yang paling relevan. Kamu dapat meninggalkan poin mu yang lain untuk pertanyaan lain atau, jika tidak ada pertanyaan lanjutan, tepuk punggungmu karena telah matang dalam persiapan dirimu.

Tidak memberikan jawaban yang lengkap saat wawancara sangat masuk akal ketika Anda mempertimbangkan pentingnya membangun hubungan selama wawancara.

Ingat: membangun hubungan baik dengan pewawancara adalah hal terpenting yang dapat kamu lakukan saat wawancara dan berbicara terlalu banyak akan merugikan tujuan yang sangat penting itu.

Berapa lama jawaban saya?

Beberapa jawaban bisa sesingkat satu kata; yang lain mungkin mengalami banyak kalimat. Itu semua tergantung pertanyaan dan keadaan. Berikut adalah beberapa pedoman berguna untuk menjaga jawabanmu dalam parameter yang dapat diterima.

Mari membuat beberapa asumsi yang masuk akal. Katakanlah wawancaramu akan berlangsung selama 40 menit. Luangkan waktu lima menit untuk menyelesaikan dan bertukar basa-basi. Itu menyisakan waktu sekitar 35 menit. (Tidak ada salahnya untuk menanyakan berapa lama wawancara akan berlangsung, tetapi tanyakan sebelum wawancara, bukan pada wawancara yang sebenarnya, agar kamu tidak memberi kesan bahwa kamu sedang terburu-buru untuk berada di tempat lain.)

Sekarang, katakanlah pekerjaan tersebut berisi sepuluh tugas dan persyaratan utama dan pewawancara telah menyiapkan dua pertanyaan per tugas/persyaratan utama.

Itu berarti kamu harus menjawab, minimal, dua puluh pertanyaan dalam 35 menit, yang berarti kamu memiliki waktu kurang dari dua menit untuk setiap pertanyaan. Ini tidak berarti bahwa kamu menyetel pengatur waktu pada satu menit lima puluh detik untuk setiap pertanyaan — ini berarti masuk akal untuk

mengasumsikan bahwa pewawancara telah menyisakan waktu kurang dari dua menit untuk menyelesaikan pertanyaan utama mereka.

Namun, masuk akal juga untuk mengasumsikan bahwa pewawancara mungkin ingin menghabiskan lebih banyak waktu untuk pertanyaan tertentu. Jika kamu telah menyelesaikan pekerjaan rumahmu, ada kemungkinan besar kamu akan tahu sebelumnya pertanyaan mana yang ingin diluangkan lebih banyak waktu oleh pewawancara.

Jika tidak, terserah kamu untuk waspada selama wawancara. Perhatikan petunjuk apa pun (seperti bahasa komponen dan nada suara) yang mungkin menunjukkan bahwa pewawancara sangat mementingkan pertanyaan tertentu. Intinya adalah tidak apa-apa untuk meluangkan sedikit waktu ekstra untuk pertanyaan semacam ini.

Hindari interpretasi pertanyaan yang subjektif atau liberal. Dengarkan baik-baik pertanyaannya, dan jawablah. Ini terdengar jelas, tetapi orang memiliki kebiasaan buruk berasumsi bahwa pewawancara ingin mendengar banyak hal lain.

Tetap berpegang pada pertanyaan. Jika pewawancara memiliki pertanyaan lain, ada kemungkinan besar mereka akan menanyakannya.

Langkah 3: Konteks

Setelah kamu membuat daftar apa yang kamu lakukan dan bagaimana kamu melakukannya di bawah langkah 2, penting untuk memikirkan konteks atau situasi di mana kamu melakukannya. Tanpa konteks, jawabanmu akan terdengar kosong atau hanya setengah jadi.

Faktanya, seperti yang akan kita lihat nanti, seringkali ide yang bagus untuk memulai jawaban kamu dengan memberikan pewawancara wawasan tentang konteks di mana kamu melakukan tugas.

Misalnya, lebih baik memulai jawaban dengan mengatakan, 'Saya merencanakan dan mengatur pekerjaan saya dalam lingkungan kewirausahaan yang serba cepat di mana klien menginginkan semuanya dengan terburu-buru', daripada mengatakan, 'Saya merencanakan dan mengatur pekerjaan saya dengan memastikan bahwa jadwal kerja saya memperhitungkan acara yang akan datang '.

Meskipun tidak ada yang salah dengan yang terakhir, yang pertama adalah awal yang lebih baik karena mengatur suasana dan memberikan wawasan yang lebih baik kepada pewawancara tentang lingkungan tempat Anda bekerja.

Dengan berbicara tentang konteks, kamu memberi pewawancara apresiasi yang lebih baik atas pekerjaan yang kamu lakukan, serta relevansinya dengan pekerjaan yang kamu lamar. Tanpa konteks yang diartikulasikan dengan jelas, jawabanmu akan terdiri dari sedikit lebih dari sekumpulan tugas yang kamu selesaikan.

Dan ada kemungkinan besar pewawancara akan mengadopsi salah satu ekspresi acuh tak acuh tersebut yang menunjukkan bahwa, apa pun yang kamu katakan setelah itu, mereka telah memutuskan kamu tidak mendapatkan pekerjaan itu.

Harap dicatat bahwa kamu hanya perlu menetapkan konteks satu kali untuk setiap pekerjaan yang kamu lakukan. Mengulangi konteks untuk pekerjaan yang sama

tidak masuk akal dan kemungkinan akan membuat pewawancara berpikir bahwa kamu menabrak sesuatu yang sulit dalam perjalanan menuju wawancara!

Langkah 4: Hasil

Langkah ini melibatkan penulisan hasil atau hasil utama dari tindakanmu. Salah satu hal yang saya perhatikan selama bertahun-tahun adalah banyak orang merasa sulit untuk mengartikulasikan hal-hal baik yang telah dihasilkan dari pekerjaan mereka.

Ketika saya bertanya kepada mereka mengapa, saya segera menemukan itu karena banyak dari mereka tidak berpikir dalam kaitannya dengan hasil. Sayangnya, pemikiran mereka terutamaterbatas pada apa yang mereka lakukan, dan terkadang bagaimana mereka melakukannya.

Namun, hasil atau pencapaian bisa dibilang aspek terpenting dari pekerjaanmu. Tidak ada gunanya melakukan semua hal yang benar jika kamu tidak mencapai hasil yang positif.

Dari sudut pandang pewawancara, hasil sangat penting. Saat memikirkan tentang hasil, akan berguna untuk memisahkannya ke dalam kategori organisasi dan pribadi.

Hasil organisasi

Hasil organisasi mencakup setiap peningkatan yang diperoleh organisasi sebagai hasil dari pekerjaanmu. Terkadang hal ini mudah diukur, terutama jika kamu pernah terlibat dalam membuat, menjual, memasang, atau mengubah sesuatu. Ketika memikirkan tentang hasil organisasi, banyak orang membatasi diri pada hasil yang nyata — atau hal-hal yang sebenarnya mereka lakukan.

Contoh hasil nyata mencakup hal-hal seperti menerapkan sistem pengarsipan baru, mengubah templat laporan, atau membangun database baru untuk melacak kontak pelanggan. Tak perlu dikatakan, penting untuk menyebutkan hasil ini dalam sebuah wawancara.

Namun, kekurangan dengan hasil yang jelas adalah bahwa mereka gagal mengartikulasikan manfaat utama mereka bagi organisasi. Mengatakan Anda menerapkan sistem pengarsipan baru itu bagus, tetapi jawabanmu akan jauh lebih baik jika kamu juga mengartikulasikan manfaat sistem pengarsipan baru ini kepada perusahaan.

Menyelesaikan jawaban kamu dengan sebuah hasil atau beberapa hasil

Sebisa mungkin, cobalah menyimpulkan dengan hasil yang positif. Meringkas poin-poin di atas, berikut adalah pertanyaan dan jawaban lengkap:

Pertanyaan: Ceritakan kepada kami tentang caramu menangani bekerja di lingkungan wirausaha yang bergerak cepat.

Saat bekerja untuk perusahaan ini, klien penting membutuhkan perubahan yang dilakukan pada salah satu pesanan yang telah dia lakukan dan dia membutuhkan perubahan ini diselesaikan dalam waktu yang sangat singkat. Mengingat bahwa sejumlah klien kami bekerja di lingkungan yang tidak dapat diprediksi, permintaan ini

tidak jarang terjadi. Tugas kami adalah memastikan bahwa kami dapat bertemu dengan mereka, jika tidak, kami akan secara efektif kehilangan pekerjaan.

Ini menetapkan konteks langkah 3. Di antara hal-hal lain, pembukaan ini memberi tahu pewawancara tentang pentingnya pekerjaanmu. Cara saya menangani pekerjaan dalam lingkungan yang menuntut seperti itu adalah dengan memastikan bahwa perencanaan saya memperhitungkan fakta bahwa segala sesuatunya dapat berubah setiap saat. Misalnya, saya menjelaskan kepada klien dan kolega saya bahwa, karena sifat pekerjaan saya, saya mungkin mengubah janji atau mengirim orang lain daripada diri saya sendiri.

Saya juga menghindari membuat komitmen jangka panjang. Mengatasi lingkungan yang begitu sibuk juga berarti saya harus membuat beberapa perubahan mendasar dalam cara berpikir saya tentang pekerjaan.

Saya harus segera membuang gagasan tentang bekerja dengan jam kerja yang dapat diprediksi dan melakukan tugas-tugas yang dapat diramalkan. Saya juga harus menerima gagasan bahwa pekerjaan seringkali tidak dapat diprediksi yang membutuhkan banyak fleksibilitas. Sekarang saya tidak pernah bisa melihat diri saya kembali ke lingkungan kerja yang mapan.

Saya juga harus siap untuk mempelajari hal-hal baru secepat mungkin. Untuk pekerjaan ini, saya harus mempelajari dasar-dasar PowerPoint dan Access dalam beberapa hari dan menerapkannya pada pekerjaan. Pelatihan kembali menjadi cara hidup, seperti halnya belajar untuk bekerja dengan baik dengan orang lain.

Ini mencerminkan langkah 2: apa yang kamu lakukan dan bagaimana kamu melakukannya. Jawabannya dengan jelas dan ringkas menyatakan tindakan apa yang diambil (perencanaan), dan memberikan contoh spesifik tentang bagaimana tindakan tersebut diambil (misalnya mengubah janji temu).

Hasil dari pekerjaan saya sangat memotivasi saya. Kami tidak hanya secara konsisten memenuhi permintaan klien, tetapi kami memiliki catatan yang sangat baik dalam hal tingkat layanan pelanggan kami sebagaimana diukur oleh survei layanan pelanggan dua kali setahun.

Ini adalah ilustrasi dari langkah 4: hasil. Dalam kasus ini, dua hasil organisasi telah dinyatakan: 'secara konsisten memenuhi permintaan klien' dan 'layanan pelanggan yang sangat baik'. Dan ada satu hasil pribadi — 'motivasi tingkat tinggi'. Jawaban ini lengkap, dan kamu mungkin tidak akan menggunakan semuanya untuk menjawab satu pertanyaan.

Namun, persiapan yang matang adalah tindakan pencegahan yang bijaksana. Kamu dapat memilih untuk menggunakan hanya sebagian dari jawaban ini sebagai tanggapan atas pertanyaan pemain tim dan menyimpan sisanya sebagai cadangan untuk pertanyaan pemain tim lain atau pertanyaan yang membutuhkan keterampilan serupa. Jangan ragu untuk 'memotong dan menempel' jawaban kamu saat diperlukan.

Unsur-unsur tanggapan wawancara yang baik yang terkandung dalam jawaban ini antara lain sebagai berikut:

- Memberikan contoh spesifik.
- Menyebutkan mempelajari dasar-dasar PowerPoint dan Access.
- Menyatakan apa yang kamu lakukan dan bagaimana kamu melakukannya — misalnya, mengubah janji; menghindari membuat komitmen jangka panjang; mempelajari hal-hal baru secepat mungkin; pelatihan ulang; dan melepaskan gagasan bahwa pekerjaan dapat diprediksi dan tidak fleksibel.
- Menyatakan hasil dan disebutkan dimotivasi oleh hasil, termasuk secara konsisten memenuhi permintaan klien dan catatan yang sangat baik dalam hal tingkat layanan pelanggan.
- Menghindari berkelok-kelok di semua tempat. Salah satu kekuatan dari empat langkah tersebut adalah kita dapat menjawab berbagai pertanyaan yang berkaitan dengan tugas atau persyaratan di bawah langkah 1.

Di bawah ini adalah tanggapan untuk beberapa pertanyaan lain yang berkaitan dengan bekerja di lingkungan kewirausahaan.

Pertanyaan: Bagian mana dari bekerja dalam lingkungan kewirausahaan yang menurut mu paling menantang?

Mengingat kerangka waktu yang singkat dan tingkat pekerjaan yang dibutuhkan, aspek yang paling menantang bagi saya setidaknya di awal — adalah memenuhi tenggat waktu klien yang ketat. (langkah 3).

Saya menghadapi tantangan ini dengan meningkatkan cara saya merencanakan keadaan darurat, dengan melatih diri saya sendiri dalam beberapa paket perangkat lunak termasuk PowerPoint dan Access, dan dengan menerapkan langkah-langkah yang meningkatkan komunikasi di antara para pemangku kepentingan utama. (Langkah 2). Hasilnya sangat positif. Saya tidak hanya mulai memenuhi tenggat waktu klien, tetapi saya juga menerapkan prosedur komunikasi yang meningkatkan efisiensi organisasi. (langkah 4).

*Pertanyaan:*Apa yang paling kamu nikmati tentang bekerja di lingkungan kewirausahaan?

Bagian yang paling saya nikmati adalah memenuhi tenggat waktu ketat yang ditetapkan oleh klien. Saya selalu merasakan kepuasan yang dalam setiap kali kami berhasil mengatasi tantangan yang sulit (langkah 4). Banyak perencanaan dan pekerjaan yang terorganisir dengan baik harus diselesaikan sebelum tenggat waktu berhasil dipenuhi. (langkah 2 dan 3).

Misalnya, kami perlu memastikan bahwa semua anggota tim terus berkomunikasi satu sama lain dan bahwa setiap orang mendapatkan pelatihan yang diperlukan. Saya menikmati bekerja di lingkungan yang serba cepat dan menantang yang membuat saya sibuk setiap hari.

Pertanyaan: Bagaimana kamu mengelola tekanan bekerja dalam lingkungan kewirausahaan yang bergerak cepat?

Saya mengelolanya dengan cukup baik. Bahkan, saya akan mengatakan lebih jauh bahwa saya menikmati bekerja di lingkungan seperti itu. Strategi yang berhasil untuk saya terdiri dari memastikan bahwa saya mendapatkan semua keterampilan yang tepat untuk melakukan pekerjaan itu, termasuk keterampilan komunikasi yang baik dan kemampuan untuk bekerja dengan baik dengan orang lain. Namun, yang sama pentingnya dengan keterampilan adalah keadaan pikiran yang benar. Saya menikmati bekerja dengan kecepatan tinggi dan dalam lingkungan yang menantang di mana perubahan adalah satu-satunya hal yang konstan. Saya tidak dapat membayangkan diri saya bekerja di lingkungan yang bergerak lambat dan dapat diprediksi.

Ingat, keempat langkah ini hanya menyediakan cara yang dapat digunakan untuk menangkap banyak data relevan dengan cara yang sederhana. Tidak ada alasan mengapa kamu tidak dapat mengubah beberapa aspek model agar sesuai dengan kebutuhanmu. Ini dirancang agar fleksibel. Berikut adalah dua contoh penting tentang bagaimana keempat langkah dapat digunakan secara berbeda.

Pertama, kamu tidak harus mengisi setiap kolom. Misalnya, jika kamu tidak memiliki hasil pribadi yang layak disebutkan, jangan menciptakannya untuk mengisi bagian itu. Hal yang sama berlaku untuk rintangan di bawah langkah 2.

Dalam beberapa kasus, orang menghadapi hambatan yang sangat kecil saat menjalankan tugas tertentu — sangat kecil, pada kenyataannya, sehingga mereka benar-benar tidak layak untuk disebutkan. Selalu tinggalkan hal-hal sepele. Idenya adalah mengisi setiap kolom hanya dengan informasi yang penting untuk pekerjaan itu dan yang menurutmu akan relevan dengan pewawancara.

Kedua, kamu dapat mengubah tajuk di bawah empat langkah agar sesuai dengan pertanyaan yang kamu ajukan. Misalnya, untuk pertanyaan yang berhubungan dengan kualitas atau masalah yang tidak terkait dengan keterampilan dan/atau tidak siap dengan prosedur langkah demi langkah, judul kolom kedua dapat disesuaikan untuk sekadar membaca 'Contoh'. Kualitas tersebut mencakup kesetiaan, kejujuran, integritas, nilai atau kepercayaan yang terkait dengan pekerjaan, dan hobi.

Karena karakteristik yang terkait dengan nilai seperti di atas adalah kualitas yang tidak memerlukan keterampilan atau pengetahuan teknis, dan yang tidak sesuai dengan urutan tindakan, kolom ini hanya akan mencantumkan contoh kapan kamu berperilaku setia atau jujur (bukan bagaimana Anda telah melakukan sesuatu). Berikut beberapa contoh pertanyaan di mana langkah 2 dapat disesuaikan:

- Beri tahu kami tentang beberapa minat kamu di luar pekerjaan.
- Kami setia kepada karyawan kami dan ingin berpikir bahwa mereka setia kepada kami. Bisakah kamu memberi kami contoh tentang kamu berperilaku setia?
- Apakah kamu lebih suka tempat kerja yang tenang atau tempat yang bising?
- Apakah kamu senang mengikuti aturan?
- Apakah kamu lebih suka mengikuti prosedur langkah demi langkah yang telah ditetapkan atau mengarangnya saat kamu pergi?

Aktivitas yang disarankan: Menggunakan empat langkah

Pilih tugas atau persyaratan pekerjaan yang kamu kenal dan, dengan menggunakan empat langkah untuk mewawancarai sukses, tangkap semua informasi relevan yang dapat kamu pikirkan (lihat Tabel 3.1).

Setelah kamu memasukkan semua informasi tentang dirimu, ajukan dua pertanyaan kepada diri kamu sendiri dengan menggunakan teknik pertanyaan perilaku yang dirujuk dalam bab ini.

Latih jawabanmu dengan lantang sampai kamu mencapai tingkat kefasihan yang memuaskan.

Ringkasan poin-poin penting

Pertanyaan 'Bisakah kamu melakukan pekerjaan itu?' Biasanya merupakan pertanyaan paling umum yang ditanyakan dalam wawancara. Mereka peduli dengan memastikan keterampilan, pengetahuan, dan pengalamanmu.

Pertanyaan ‘Bisakah kamu melakukan pekerjaan itu?’ Dapat dibagi menjadi tiga kategori: pertanyaan tentang tugas yang telah kamu lakukan sebelumnya;

Pertanyaan tentang tugas yang belum kamu lakukan tetapi keterampilannya telah kamu kuasai;

Pertanyaan tentang tugas yang sama sekali baru bagi dirimu.

Langkah penting pertamamu untuk mempersiapkan jawaban wawancara kamu adalah mencari tahu sebanyak mungkin tentang pekerjaan itu.

Empat langkah untuk kesuksesan wawancara memberikan kerangka kerja yang mudah digunakan yang dengannya kamu dapat menangkap semua informasi relevan yang kamu butuhkan untuk menyusun jawaban wawancara. Selain menangkap apa yang kamu lakukan dan bagaimana kamu melakukannya, itu juga memaksa kamu untuk berpikir tentang konteks dan hasil. Ini cocok untuk menjawab pertanyaan perilaku dan dapat digunakan dengan cara yang fleksibel.

Waspadai jawaban bertele-tele.

Cara paling efektif untuk mengumpulkan informasi yang kamu tangkap menggunakan empat langkah adalah mengajukan pertanyaan wawancara hipotetis kepada dirimu sendiri dan kemudian menjawabnya dengan lantang sampai kamu menjadi fasih.

Jawaban wawancara yang baik biasanya berisi poin-poin berikut:

Sebuah konteks;

Contoh spesifik;

Apa yang kamu lakukan dan bagaimana kamu melakukannya;

Hasil;

Ini akan langsung ke intinya.

“Saat kamu mengatakan“ Sulit ”, itu sebenarnya berarti“ Saya tidak cukup kuat untuk memperjuangkannya ”. Berhenti berkata keras. Berpikir positif!" - Harry.

BAB 4
POTENSI ANDA UNTUK MENANGANI TUGAS BARU

Terkadang, kamu akan ditanyai pertanyaan yang tidak ada hubungannya dengan tugas dan pencapaianmu sebelumnya. Lebih buruk lagi, keterampilan yang melekat dalam tugas-tugas ini akan sangat berbeda dengan keterampilan yang sudah kamu miliki, sehingga menjadikannya pertanyaan wawancara yang paling menantang.

Biasanya, kamu ditanyai jenis pertanyaan ini ketika kamu memulai karirmu, mengubah karir atau pergi untuk promosi yang memerlukan tugas baru seperti mengelola orang dalam satu atau beberapa tim.

Jelasnya, jika kamu belum pernah melakukan tugas sebelumnya, membuat hubungan langsung dengan tugas atau keterampilan di masa lalu menjadi bermasalah. Namun, tidak ada alasan untuk putus asa. Ada banyak narasumber yang berhasil menjawab pertanyaan semacam ini secara teratur. Seperti yang telah kamu pelajari, kunci sukses adalah persiapan yang benar.

Pecahkan Segala Masalah yang Ada pada Tugas Anda

Langkah pertama melibatkan kamu untuk mengambil setiap tugas baru dan memecahnya menjadi keterampilan dan pengetahuan individu yang mereka buat. Potongan informasi individual yang kamu dapatkan akan menjadi inti dari jawabanmu. Terkait empat langkah kami, informasi ini akan berada di langkah 2.

Mengurai tugas yang belum pernah kamu lakukan sebelumnya terkadang bisa menjadi latihan yang sulit, terutama jika kamu tidak memiliki pengalaman untuk melakukannya. Tetapi jangan menyerah setelah beberapa kali mencoba, itu menjadi mudah. Berikut adalah beberapa pedoman yang menurutmu berguna. Mulailah dengan bertanya pada diri sendiri pertanyaan, 'Untuk melakukan tugas atau persyaratan tertentu, langkah apa yang harus saya ambil?' Lakukan sesi curah pendapat. Jangan mengabaikan detail apapun, betapapun remehnya anda berpikir itu.

Tuliskan segala sesuatu dan apa pun yang muncul di kepalamu. Kamu bisa membuang barang yang tidak penting nanti. Apa yang menurut kamu sepele dan tidak layak disebut sering kali ternyata merupakan keterampilan yang penting. Contoh bagusnya adalah keterampilan mendengarkan.

Kebanyakan orang bahkan tidak berpikir untuk menyebutkan keterampilan ini, namun keterampilan mendengarkan yang baik sangat penting untuk keterampilan interpersonal yang efektif — termasuk menjadi pemain tim, pemecahan masalah, dan resolusi konflik. Keterampilan ini juga sangat sulit untuk dikuasai, terutama saat kamu mendengar sesuatu yang tidak kamu setujui.

Jika kamu mengalami masalah dalam mendapatkan ide, jangan khawatir. Hubungi teman, rekan kerja, mantan manajer, atau siapa pun yang menurutmu dapat menjelaskan masalah ini. Kamu akan segera menemukan bahwa dua atau tiga kepala lebih baik daripada satu. Apa pun yang kamu lakukan, jangan menyerah. Kamu akan melakukannya — ini hanya masalah memahaminya.

Jika temanmu tidak dapat membantu, jangan panik. Saatnya berkonsultasi dengan buku atau pakar di bidangnya. Jika, misalnya, kamu melamar posisi manajer dan kamu belum pernah mengelola orang sebelumnya, masuk akal untuk berbicara dengan seseorang yang tahu apa yang terlibat. Sebisa mungkin cobalah untuk membuat daftar keterampilan komponen dalam urutan yang logis dan berurutan.

Sebelum kamu menyelesaikan daftarmu, kamu harus meneliti jawabanmu. Karena kamu belum pernah melakukan tugas ini sebelumnya, masuk akal bahwa beberapa jawabanmu mungkin agak naif atau salah. Teliti kualitas jawabanmu dengan berbicara kepada seseorang yang tahu dan/atau dengan bertanya pada diri sendiri pertanyaan yang sangat penting, ‘Dapatkah saya mendukung jawaban saya secara kredibel jika ditanyai secara lebih rinci oleh pewawancara? '

Dalam kebanyakan situasi, tidak ada jawaban yang benar atau salah tentang bagaimana tugas dilaksanakan. Pertanyaan yang membutuhkan jawaban yang sangat teknis dan harus sangat spesifik, tentu saja, merupakan pengecualian. Sebagai individu unik yang bekerja di lingkungan yang bervariasi, kita menghadapi tantangan berbeda yang memengaruhi cara kita melakukan sesuatu.

Jadi, cara saya bekerja dalam lingkungan kewirausahaan, atau merencanakan dan mengatur pekerjaan saya, mungkin berbeda dengan metode orang lain — tetapi metode ini juga valid atau efektif. Dalam hal wawancara, yang terpenting adalah memberikan jawaban yang ringkas dan logis yang dapat menahan pengawasan jika pewawancara memutuskan untuk menggali lebih dalam jawaban Anda. Dan pewawancara yang baik selalu menggali lebih dalam.

Kamu harus yakin pada apa yang menurutmu cara yang benar untuk melakukan suatu tugas, tetapi pastikan kamu telah memikirkan jawabanmu dengan matang. Tanyakan pada dirimu sendiri, 'Mengapa saya harus mengambil tindakan tertentu?' Dengan segala cara, berkonsultasilah dengan para ahli dan dengarkan baik-baik apa yang mereka katakan, tetapi pada akhirnya itu harus menjadi jawabanmu.

Mari kita lihat contoh umum dari tugas baru: mengelola orang di tempat kerja. Mengelola staf, seperti yang diketahui oleh siapa pun yang telah didorong ke posisi itu, membutuhkan serangkaian keterampilan baru yang beberapa di antaranya bisa sangat menantang. Mengingat pentingnya sebagian besar organisasi menempatkan manajemen orang yang efektif, bagaimana kamu menanggapi jenis pertanyaan ini dapat dengan mudah membuat atau menghancurkan wawancaramu. Berikut ini beberapa contoh pertanyaan tentang staf:

Bagaimana kamu akan memimpin tim yang terdiri dari para profesional yang sangat terlatih?

Sebagai manajer orang, bagaimana kamu akan memotivasi mereka dan memaksimalkan kinerja mereka?

Jelaskan manajer idealmu. Tugas yang melekat dalam mengelola manusia di tempat kerja mungkin antara lain:

Mendelegasikan pekerjaan dengan tepat, dengan mempertimbangkan kemampuan staf dan pertimbangan multi-keterampilan;

- Memberikan umpan balik yang tepat waktu dan obyektif;
- Konsultasi tentang hal-hal yang mempengaruhi staf;
- Mengakui dan menghargai usaha mereka;
- Memperlakukan semua orang sama.

Dengan menggunakan empat langkah, kamu akan memasukkan tugas-tugas ini — atau tugas lain yang mungkin kamu anggap penting — di bawah langkah 2 (lihat Tabel di seberang).

Ciptakan konteks yang relevan

Mengingat kamu belum pernah melakukan tugas ini, masuk akal jika kamu tidak dapat memberikan konteks kehidupan nyata seperti yang kamu lakukan untuk tugas yang sebenarnya kamu lakukan.

Namun, hal ini tidak mencegah kamu membuat konteks — konteks yang akan sama atau mirip dengan pekerjaan yang kamu lamar. Melakukan ini mendorong kamu untuk melangkah lebih jauh dengan menempatkan jawaban langkah 2 mu dalam situasi 'kehidupan nyata'.

Dengan melakukan ini, kamu akan berada pada posisi yang lebih baik untuk membuat jawabanmu terdengar lebih meyakinkan.

Dalam sebuah wawancara, kamu mungkin akan ditanyai pertanyaan kontekstual — yaitu, pertanyaan yang menanyakan bagaimana kamu akan melakukan tugas dalam situasi atau konteks tertentu. Jadi, alih-alih ditanya 'Bagaimana kamu akan mengelola staf?' (Tanpa konteks), kemungkinan kamu harus menjawab pertanyaan seperti 'Bagaimana kamu akan mengelola tim profesional yang bermotivasi tinggi dalam waktu singkat. lingkungan -pacu? '

Jika kamu kebetulan ditanyai pertanyaan tanpa konteks (pewawancara yang tidak berpengalaman diketahui mengajukan pertanyaan dekontekstual), dapat menempatkannya ke dalam konteks yang relevan dengan pekerjaan yang kamu lamar kemungkinan akan mengesankan pewawancara.

Tanpa konteks, jawaban kamu hanya akan terdengar setengah selesai. Tempatkan konteks khayalanmu di bawah langkah 3 dalam empat langkah untuk mewawancarai tabel sukses. Hasil yang diharapkan Mengingatmu belum pernah melakukan tugas ini sebelumnya, tidak masuk akal untuk membicarakan hasil yang sebenarnya. Namun, ada baiknya untuk berpikir dalam kerangka hasil yang diharapkan — yaitu, apa yang mungkin terjadi jika kamu mengelola orang secara efektif.

Ada dua keuntungan berpikir tentang hasil yang diharapkan. Pertama, karena hasil mirip dengan sasaran, pewawancara akan menunjukkan bahwa kamu berpikir dalam kaitannya dengan tujuan atau hasil akhir, bukan hanya proses. Ingat, pencapaian tujuan yang paling penting bagi pengusaha.

Kedua, banyak pewawancara suka mengajukan pertanyaan yang mengganggu seperti, 'Dan apa yang kamu lihat dari hasil pendekatan manajemen orang Anda?' Dan, 'Apa yang dapat dicapai oleh manajer orang yang baik dengan stafnya?' kolom dengan

hasil yang kamu harapkan akan membantu dirimu merumuskan jawaban yang efektif untuk pertanyaan semacam itu (lihat Tabel 4.1 di bawah).

Ketahuilah bahwa beberapa perusahaan memilih untuk tidak menggunakan istilah 'mengelola' orang. Sebaliknya mereka lebih menyukai istilah 'memimpin' orang. Jika kamu mempersiapkan jawaban untuk serangkaian pertanyaan mengelola/mengarahkan orang, pastikan kamu membiasakan diri dengan bahasa manajemen perusahaan. Terkadang, hanya menggunakan kata yang tepat dapat membuat perbedaan besar.

Tabel 4.1 Langkah-langkah pertanyaan dan jawaban

Langkah 1	**Langkah 2**	**Langkah 3**	**Langkah 4**
Tugas/persyaratan jabatan	Apa yang akan dilakukan untuk memastikan tugas yang tercantum pada langkah 1 dilakukan dengan benar, termasuk mengatasi hambatan. Mendelegasikan pekerjaan dengan tepat, dengan mempertimbangkan kemampuan staf dan pertimbangan multiskilling. Memberikan umpan balik yang tepat waktu dan obyektif. Konsultasi tentang hal-hal yang mempengaruhi staf. Mengakui dan Mengakui upaya mereka	Konteks imajiner	Hasil yang diharapkan organisasi dan organisasi pribadi. Menjaga atau meningkatkan motivasi staf, yang akan berkontribusi pada peningkatan kinerja individu dan tim, termasuk kemungkinan peningkatan tingkat ketidakhadiran dan pergantian Personal. Menunjukkan kemampuan untuk dengan cepat mempelajari produk yang saya sellin
Mengelola orang di tempat kerja		Mengelola tim kecil yang terdiri dari profesional	

		bermotivasi tinggi dalam lingkungan yang bergerak cepat	

Menyatukan semuanya -

Mari kita lihat pertanyaan yang mungkin muncul dan kemungkinan tanggapannya.

Pertanyaan: Bagaimana kamu akan memimpin tim yang terdiri dari para profesional yang sangat terlatih? Saya pasti akan mempertimbangkan fakta bahwa saya berurusan dengan para profesional —yaitu, orang-orang yang sangat terlatih yang harus tahu apa yang mereka lakukan (langkah 3). Dalam mendelegasikan pekerjaan, saya akan mempertimbangkan kemampuan, preferensi, dan beban kerja mereka saat ini. Saya akan memastikan bahwa pekerjaan didistribusikan secara merata, dengan mempertimbangkan kebutuhan organisasi (langkah 2).

Saya sangat percaya dalam memberi masukan kepada orang lain. Tanpa umpan balik, staf sering kali tidak menyadari hal-hal penting yang berkaitan dengan kinerja mereka. Saya akan memastikan bahwa umpan balik saya tepat waktu dan obyektif — yaitu, berdasarkan fakta dan bukan dugaan (langkah 2).

Saya juga percaya dalam berkonsultasi dengan staf dalam hal-hal yang berkaitan dengan pekerjaan mereka. Staf tidak hanya merasa lebih dihargai ketika diajak berkonsultasi, tetapi seringkali manajemen dapat diberi tahu tentang hal-hal penting yang sebelumnya tidak mereka sadari (langkah 2).

Mengakui dan mengakui upaya individu dan tim, menurut saya, juga sangat penting — terutama dalam hal memberi staf perasaan - dihargai. Saya tahu bahwa ketika manajer saya mengakui sesuatu yang istimewa yang saya lakukan, saya selalu merasa senang (langkah 2).

Saya sangat yakin bahwa manajemen orang yang efektif sangat penting untuk kesuksesan tim mana pun. Manajer yang baik mampu memotivasi dan mengeluarkan yang terbaik dari orang-orangnya. Hal ini pada gilirannya memberikan kontribusi yang signifikan terhadap kinerja tim. Ini akan menjadi tujuan saya untuk memaksimalkan kinerja tim saya dengan menerapkan teknik yang telah disebutkan (langkah 4).

Kekuatan jawaban di atas adalah sebagai berikut:

- Mengakui konteks profesional tim yang akan dikelola;
- Mengartikulasikan poin dalam urutan yang jelas dan berurutan;
- Memberi alasan mengapa tindakan tertentu akan diambil;
- Menyelesaikan dengan hasil yang diharapkan.

Aktivitas yang disarankan: Potensi Anda untuk menangani tugas-tugas baru

Untuk membantumu mempersiapkan jawabanmu untuk setidaknya satu tugas yang secara substansial berbeda dengan apa pun yang telah kamu lakukan sebelumnya, ikuti pedoman yang dijelaskan di atas, yaitu:

- Pecahkan tugas menjadi beberapa komponennya masing-masing.
- Gunakan empat langkah dan letakkan masing-masing komponen tersebut pada langkah 2.
- Gunakan konteks imajiner yang mungkin muncul dalam wawancara.
- Sertakan hasil yang diharapkan.

Ringkasan poin-poin penting

Pertanyaan tentang tugas yang secara substansial berbeda dengan apa pun yang telah kamu lakukan sebelumnya biasanya merupakan yang paling menantang dalam sebuah wawancara.

Seringkali tidak ada satu jawaban (atau satu jawaban yang benar) tentang bagaimana tugas dilaksanakan. Kami semua berasal dari latar belakang pekerjaan yang berbeda dan membawa serta berbagai cara untuk melakukan sesuatu.

Nama-nama langkah 2, 3 dan 4 akan sedikit berubah, mencerminkan tantangan berbeda yang ditimbulkan oleh jenis pertanyaan ini. Secara khusus, judul konteks (konteks imajiner) dan hasil (hasil yang diharapkan) berubah untuk memperhitungkan fakta bahwa Anda belum pernah melakukan tugas ini sebelumnya.

BAB 5
PENGUSAHA SUKA KARYAWAN YANG TERMOTIVASI

Pemberi kerja yang berpengalaman tahu bahwa karyawan yang bermotivasi tinggi sangat berharga. Karyawan yang termotivasi cenderung belajar banyak hal dengan cepat, menyelesaikan tugas mereka dengan antusias, peduli dengan bisnis dan sering melampaui panggilan tugas. Bandingkan ini dengan karyawan yang tidak termotivasi. Bahkan orang-orang bertalenta tinggi yang kurang motivasi bisa berbatasan dengan yang tidak efektif. Seperti yang dikatakan seorang majikan yang sukses kepada saya:

“Beri saya motivasi atas bakat setiap hari. Orang yang termotivasi mengembangkan bakat dengan dorongan dan antusiasme mereka. Mereka mengajukan pertanyaan, menjadi sukarelawan untuk pekerjaan dan mengatasi segala kekurangan yang mungkin mereka miliki. Mereka berharga dua kali lebih banyak daripada orang-orang berbakat yang kurang motivasi. Orang berbakat yang tidak termotivasi adalah sebuah oxymoron. "

Mengkomunikasikan tingkat motivasi Anda

Pada wawancara, tingkat motivasi kandidat cenderung disimpulkan oleh pewawancara. Dengan kata lain, pewawancara menangkap sinyal yang diberikan oleh narasumber. Sinyal-sinyal ini dapat dipecah menjadi tiga kelompok:

- *Apa yang dikatakan;*
- *Bagaimana dikatakan;*
- *Bahasa komponen*

Bab ini akan berfokus pada dua kelompok pertama apa yang dikatakan dan bagaimana dikatakan. Bab terakhir akan membahas bahasa komponen. Cukuplah untuk mengatakan bahwa meyakinkan atasan bahwa kamu sangat termotivasi bertumpu pada lebih dari sekedar kata-kata yang keluar dari mulutmu. Bahasa komponenkamu dan cara kamu mengatakan sesuatu sama-sama penting.

Terlepas dari pentingnya motivasi di tempat kerja, pertanyaan tentang motivasi tidak sesering yang seharusnya. Salah satu alasannya adalah bahwa ada banyak pewawancara yang tidak berpengalaman di luar sana yang tidak yakin bagaimana menyusun pertanyaan motivasi.

*Pertan yaan seperti, 'Seberapa termotivasi Anda?'*terdengar sangat amatir dan cenderung menarik jawaban seperti, 'Saya sangat termotivasi. Jika kamu memberi saya pekerjaan ini, saya akan bekerja sangat keras."

Ketika pertanyaan motivasi langsung ditanyakan, biasanya dimulai dengan kata-kata 'mengapa' dan 'apa'. Berikut beberapa contoh klasik:

- *Kenapa kamu ingin bekerja di sini?*
- *Mengapa Anda ingin melakukan pekerjaan ini?*
- *Apa yang menarik minat Anda tentang pekerjaan ini?*
- *Hal apa yang Anda sukai dari bekerja di lingkungan seperti ini?*
- *Apa yang Anda sukai dari pekerjaan ini?*
- *Hal-hal apa saja yang Anda sukai di tempat kerja?*

Cara yang berguna untuk mempersiapkan semua pertanyaan di atas adalah dengan bertanya pada diri sendiri, 'Hal-hal apa yang saya sukai dari pekerjaan ini?' Atau, dengan kata lain, 'Mengapa saya menyukai pekerjaan semacam ini?' apa yang kamu sukai dari pekerjaan tertentu, kamu perlu melihat tugas pekerjaan itu dengan sangat hati-hati.

Langkahmu selanjutnya adalah membuat daftar semua hal yang menarikmu ke pekerjaan itu, sespesifik mungkin. Kamu harus spesifik, jika tidak jawabanmu mungkin terdengar hampa.

Pernyataan yang luas seperti 'Saya suka retail', misalnya, hampir tidak meyakinkan seperti 'Saya suka berinteraksi dengan orang setiap hari' atau 'Saya suka sensasi melakukan penjualan dan melihat pelanggan yang bahagia meninggalkan toko'. Itu karena dua pernyataan terakhir tidak hanya memberi tahu pewawancara bahwa kamu menyukai ritel, tetapi juga menjelaskan alasannya. Berikut beberapa contoh pernyataan motivasi yang menggairahkan pemberi kerja (tapi pastikan kamu memiliki contoh spesifik untuk mendukung pernyataanmu):

- *Saya suka bekerja dengan orang.*
- *Saya sangat menikmati tantangan seperti yang Anda sebutkan.*
- *Saya sangat suka bekerja dengan angka.*
- *Berinteraksi dengan orang-orang itulah yang membuat saya bangun dari*
- *tempat tidur di pagi hari.*
- *Saya sangat menikmati bekerja sendiri.*
- *Saya senang mempelajari hal baru.*
- *Saya suka Pemrograman.*
- *Memecahkan masalah kompleks adalah hal yang paling saya sukai.*
- *Saya merasakan kepuasan yang dalam saat membuat pelanggan senang.*
- *Saya sangat tertarik untuk memecahkan masalah teknis.*
- *Saya suka bekerja dengan komputer.*
- *Saya benar-benar bekerja di lingkungan seperti ini.*
- *Saya tidak pernah merasa cukup dengan pekerjaan seperti ini.*

Jangan sembunyikan antusiasme Anda

Kamu akan mengetahui bahwa semua pernyataan di atas memiliki satu kualitas yang sangat penting: semuanya diungkapkan dengan antusias. Hindari bahasa pemalu atau tidak pasti karena kamu akan terdengar tidak meyakinkan. Tempatkan dirimu pada posisi pemberi kerja dan bandingkan dua jawaban berikut tentang layanan pelanggan. Manakah dari keduanya yang lebih suka kamu dengar saat wawancara?

Jawaban 1: Secara keseluruhan saya suka berurusan dengan pelanggan meskipun mereka bisa sangat menjengkelkan dan mengajukan pertanyaan bodoh. Tetapi saya menyadari bahwa tanpa pelanggan saya akan kehilangan pekerjaan, jadi saya berusaha keras untuk memuaskan mereka.

Jawaban 2: Saya suka berurusan dengan pelanggan. Saya sangat menikmati interaksi dengan orang-orang, termasuk menjawab semua pertanyaan mereka — tidak peduli betapa sepele mereka kelihatannya. Saya merasakan kepuasan yang dalam ketika saya dapat memecahkan masalah untuk pelanggan atau membantu mereka dengan cara tertentu.

Jelas, jawaban kedua adalah yang lebih baik. Ini dimulai dengan pernyataan yang sangat antusias dan memperkuatnya dengan beberapa penegasan lagi. Ini penuh dengan energi positif dan memberikan kesan yang jelas bahwa orang tersebut sangat termotivasi dalam memberikan standar layanan pelanggan yang tinggi.

Perhatikan juga bahwa jawaban ini membuat pernyataan nilai — yaitu, 'Saya merasakan kepuasan yang dalam'. Dengan melakukan itu, ini memberi kita wawasan tentang keyakinan atau nilai-nilai pembicara dan karenanya sebagian membahas 'Apakah kamu tipe orang yang dapat kami ajak bekerja sama? pertanyaan.

Di sisi lain, jawaban pertama terdengar seolah-olah orang tersebut memberikan layanan pelanggan yang baik karena terpaksa. Kita semua tahu bahwa pelanggan terkadang mengajukan pertanyaan bodoh, tetapi wawancara bukanlah tempat untuk mengartikulasikan pandangan seperti itu.

Informasi yang kamu kumpulkan menggunakan empat langkah juga dapat digunakan untuk menjawab pertanyaan motivasi. Informasi di bawah langkah 2 dapat menjadi sumber informasi spesifik yang kaya ketika membahas pertanyaan motivasi.

Katakanlah, misalnya, kamu melamar pekerjaan di mana kamu harus memimpin tim dan kamu ditanyai salah satu pertanyaan motivasi klasik. Seperti inilah suara pertukaran itu:

Pertanyaan: Apa yang menarik minat Anda tentang pekerjaan ini?

Ada banyak hal yang sangat menarik minat saya tentang pekerjaan ini. Salah satunya adalah kesempatan untuk memimpin tim yang terdiri dari orang-orang pekerja keras. Saya suka mengeluarkan yang terbaik dari orang-orang dan melihat mereka mendapatkan hasil maksimal dari pekerjaan mereka. Saya dapat melakukan ini dengan menerapkan prinsip-prinsip kepemimpinan tim yang baik. Misalnya, saat mendelegasikan pekerjaan, saya memperhitungkan kemampuan orang serta beban kerja. Saya memberikan umpan balik yang tepat waktu dan konsisten yang dirancang untuk meningkatkan kinerja orang. Saya berkonsultasi dengan orang-orang, mengakui

pekerjaan yang baik dan memperlakukan semua orang dengan setara. Mendapatkan rasa hormat dari tim Anda adalah pengalaman yang sangat memotivasi.

Bagian tebal dari jawaban di atas diambil langsung dari kolom kedua Tabel Dengan menyatakan secara spesifik apa yang Anda lakukan untuk memimpin tim dengan sukses, kamu memberikan kredibilitas pada klaimmu tentang menikmati 'mengeluarkan yang terbaik dari orang-orang'.

Hal yang menarik tentang jawaban ini adalah bahwa ini berfungsi pada beberapa level:

- *Ini menjawab pertanyaan secara langsung.*
- *Ini memberi tahu pewawancara bahwa Anda mungkin pemimpin tim yang hebat.*
- *Ini memberi tahu pewawancara bahwa mengeluarkan yang terbaik dari orang-orang adalah sesuatu yang sangat memotivasi Anda.*
- *Ia melakukan semua hal di atas tanpa bimbang.*
- *Sebuah kata peringatan tentang motivator*

Saat menyusun jawabanmu tentang hal-hal yang kamu sukai dari pekerjaan itu, ada beberapa hal yang perlu kamu perhatikan. Ini termasuk:

- *Uang;*
- *Kedekatan dengan tempat tinggal Anda;*
- *Jam nyaman;*
- *Teman yang bekerja di sana.*

Semua ini dapat menjadi motivator penting bagi banyak orang — dan tentu saja dapat disebutkan selama wawancara. Namun, mereka tidak boleh disebut sebagai motivator utama karena tidak satupun dari mereka ada hubungannya dengan kinerjamu yang baik dalam pekerjaan itu.

Motivator utama harus dikaitkan dengan sifat pekerjaan itu sendiri, dan harus menunjukkan kemampuan untuk bekerja dengan baik di bidang utama pekerjaan itu. Jauh lebih efektif untuk mengatakan bahwa kamu suka bekerja dengan orang daripada menyukai uang atau waktu perjalananmu akan berkurang setengahnya!

Aktivitas yang disarankan: Motivasi

Buat daftar semua hal yang menarik dirimu ke pekerjaan pilihanmu. Jika kamu mengalami masalah dalam mencari jawaban, perhatikan dengan saksama tugas utama dan tanyakan pada dirimu sendiri, 'Apa yang saya sukai dari tugas ini?' Ingatlah untuk menghindari pernyataan yang luas. Mohon jelaskan dengan spesifik. Setelah kamu menyusun daftar, jawablah pertanyaan-pertanyaan berikut. Terus latih jawabanmu sampai kamu puas dengan kefasihan mu.

Pertanyaan 1: Mengapa Anda melamar pekerjaan ini?

Pertanyaan 2: Hal-hal apa saja yang memotivasi Anda?

Ringkasan poin-poin penting

Pewawancara yang meyakinkan bahwa Anda sangat termotivasi membutuhkan lebih dari sekadar mengatakan hal yang benar. Bahasa komponen dan cara Anda mengatakan sesuatu sama pentingnya.

Saat mempersiapkan jawaban Anda atas pertanyaan motivasi, salah satu pertanyaan bermanfaat yang dapat Anda tanyakan pada diri sendiri adalah 'Mengapa saya menyukai pekerjaan semacam ini?' Tanggapan spesifik Anda untuk pertanyaan ini akan menjadi inti dari jawaban motivasi Anda.

Ekspresikan diri Anda dengan antusias. Pewawancara berharap untuk melihat ketajaman pada kandidat yang termotivasi.

Langkah 2 dari empat langkah seringkali merupakan sumber informasi yang baik untuk pertanyaan motivasi.

Hindari menyebutkan motivator seperti uang dan waktu perjalanan — mereka tidak berkontribusi pada kemampuan Anda untuk bekerja dengan baik.

“Jangan khawatir tentang kegagalan, khawatir tentang peluang yang Anda lewatkan ketika Anda bahkan tidak mencoba.” - Harry.

“Rasa sakit yang Anda rasakan hari ini adalah kekuatan yang Anda rasakan besok. Untuk setiap tantangan yang dihadapi, ada peluang untuk berkembang. ” - Harry.

BAB 6
PERTANYAAN ‘LIMA BESAR’

Ada lima pertanyaan umum yang sangat umum yang muncul di hampir setiap wawancara. Mereka berhubungan dengan:

- *Menjadi pemain tim yang baik;*
- *Merencanakan dan mengatur pekerjaan Anda secara efektif;*
- *Komunikasi interpersonal yang baik;*
- *Mengatasi perubahan di tempat kerja;*
- *Memberikan layanan pelanggan yang efektif (termasuk pelanggan internal).*
- Dengan menggunakan empat langkah, bab ini mengajukan pertanyaan tentang masalah ini dan menyarankan kemungkinan tanggapannya.

Pentingnya pertanyaan 'lima besar'

Keterampilan yang tercantum di atas sangat penting untuk sebagian besar pekerjaan. Sulit untuk memikirkan pekerjaan di mana kelimanya tidak berperan pada satu tahap atau lainnya, dan tidak mungkin untuk memikirkan pekerjaan di mana setidaknya salah satu dari mereka tidak relevan. Untuk alasan ini, 'lima besar' sebenarnya merupakan ratusan pertanyaan wawancara.

Setelah kamu mempelajari cara menjawab pertanyaan 'lima besar', kamu akan dapat menjawab banyak pertanyaan lain karena ada banyak tumpang tindih di antara mereka. Misalnya, jika kamu dapat menjawab pertanyaan dasar, 'Apa yang membuat kamu menjadi pemain tim yang baik?', kamu juga harus dapat menjawab berbagai pertanyaan pemain tim yang serupa, termasuk:

- *Bagaimana Anda suka bekerja dalam tim?*
- *Apakah Anda menganggap diri Anda sebagai pemain tim yang baik?*
- *Jelaskan tim ideal Anda.*
- *Apa yang diperlukan untuk menjadi pemain tim yang efektif?*

Namun, ketahuilah bahwa, meskipun mempelajari cara menanggapi satu pertanyaan umum memungkinkan kamu menjawab banyak pertanyaan serupa, bukan berarti kamu akan dapat menjawab setiap pertanyaan yang mungkin diajukan. Terserah kamuuntuk rajin dan mencari pertanyaan dalam genre yang mungkin sedikit berbeda atau tidak terduga.

Mengingat sifat umum dari keterampilan di atas, keterampilan tersebut akan diperlakukan seolah-olah telah dilakukan sebelumnya.

Menjawab pertanyaan 'pemain tim'

Kebanyakan orang bekerja dalam tim. Bahkan orang yang tampaknya bekerja sendiri sering harus berinteraksi dengan orang lain dalam organisasi, sehingga menciptakan satu atau lebih tim yang terbentuk secara longgar. Beberapa tim perlu bekerja sama secara erat, yang lainnya kurang begitu; beberapa tim bekerja bersama sepanjang waktu, sedangkan yang lain hanya bertemu secara berkala. Poin pentingnya adalah bahwa pemberi kerja sangat bergantung pada kelancaran fungsi tim mereka dan ingin merekrut pemain tim yang efektif. Berikut beberapa contoh pertanyaan pemain tim:

- Apa yang membuat Anda menjadi pemain tim yang baik?
- Bagaimana Anda dapat bekerja dalam tim?
- Apakah Anda lebih suka bekerja sendiri atau dalam tim? Mengapa?
- Apa yang Anda tidak suka ketika bekerja dalam tim?
- Apa yang akan Anda lakukan jika salah satu kolega Anda tidak menarik beban mereka?
- Jelaskan tim ideal Anda.
- Dapatkah Anda memberi kami contoh tentang apa yang telah Anda lakukan untuk memastikan bahwa peran Anda dalam tim adalah peran yang positif?
- Bagaimana Anda akan menangani anggota tim yang keras dan agresif pada pertemuan tim dan mendominasi proses dengan mengintimidasi orang lain?

Sekarang, mari gunakan empat langkah untuk menyiapkan informasi yang diperlukan untuk menanggapi pertanyaan 'pemain tim': Berikut adalah contoh pertanyaan wawancara dan kemungkinan tanggapan.

Pertanyaan: Apakah Anda pemain tim yang baik? Dapatkah Anda memberi kami contoh tentang Anda yang mendemonstrasikan kemampuan pemain tim?

Ya, saya menganggap diri saya sebagai pemain tim yang efektif. Dalam pekerjaan saya sebelumnya, saya adalah bagian dari tim yang terdiri dari empat orang yang bertanggung jawab untuk membayar gaji, termasuk lembur dan bonus, sekitar 2000 karyawan. Ketika saya pertama kali mulai bekerja di tim, ada masalah komunikasi antara beberapa anggota tim. Selain memengaruhi kinerja kami, masalah ini juga merusak hubungan antara anggota tim tertentu. Setelah beberapa minggu, saya berpikir bahwa jika kami mengadakan pertemuan yang lebih teratur dan kursi bergilir, komunikasi akan meningkat.

Ketika saya memberikan saran ini, anggota tim menyetujuinya dan, singkatnya, format rapat yang baru ternyata berhasil. Baik komunikasi dan kinerja meningkat. Saya juga menunjukkan kemampuan pemain tim saya dengan mengakui pendapat dan kontribusi kolega saya, serta membantu anggota tim ketika mereka mengalami masalah.

Saya pikir ketika Anda bersedia membantu orang lain, mereka akan membantu Anda ketika Anda membutuhkannya sebagai imbalan dan itu hanya akan baik untuk tim. Saya juga berusaha membagikan semua informasi yang menurut saya perlu diketahui oleh rekan-rekan saya. Saya akan menyebutkan bahkan informasi yang

tampaknya tidak penting seperti individu yang mengeluh tentang gaji mereka dan kesalahan kecil dengan perangkat lunak karena seringkali hal-hal kecil dapat menyebabkan masalah besar di kemudian hari.

Menurut kolega saya, kehadiran saya di tim menyebabkan komunikasi yang lebih baik di antara anggota tim, serta dengan klien kami, yang memberikan kontribusi signifikan terhadap kinerja kami secara keseluruhan. Secara khusus, tingkat kesalahan kami berkurang setengahnya dalam dua bulan.

Ingatlah bahwa, kecuali pewawancara secara khusus memberi tahu Anda bahwa perusahaan sangat menekankan pada mempekerjakan seseorang dengan keterampilan pemain tim yang efektif, kemungkinan besar Anda tidak akan menggunakan setiap aspek dari jawaban di atas untuk menanggapi satu pertanyaan.

Anda dapat memutuskan untuk menggunakan sebagian dan menyimpan sisanya sebagai cadangan untuk pertanyaan lanjutan atau pertanyaan yang mencari informasi tentang keterampilan serupa. Adalah bijaksana untuk mempersiapkan diri secara berlebihan dan bahkan lebih bijaksana untuk mengetahui kapan harus berhenti. Prinsip yang sama berlaku untuk pertanyaan 'lima besar' lainnya.

Menjawab pertanyaan perencanaan dan pengorganisasian -

Sulit untuk memikirkan pekerjaan yang tidak melibatkan perencanaan dan pengorganisasian. Jika kita menerima bahwa teknologi sebagian besar telah mengambil alih banyak tugas berulang yang dilakukan oleh orang-orang di masa lalu, sebagian besar pekerjaan saat ini melibatkan semacam perencanaan dan pengorganisasian. Oleh karena itu, pertanyaan perencanaan dan pengorganisasian cenderung menjadi agenda utama banyak pewawancara. Berikut adalah beberapa pertanyaan umum tentang perencanaan dan pengorganisasian.

- *Beri tahu kami bagaimana Anda merencanakan dan mengatur jadwal kerja Anda.*
- *Dapatkah Anda memberi kami contoh kapan Anda harus merencanakan dan mengatur acara penting atau aktivitas terkait pekerjaan? Langkah apa yang Anda ambil?*
- *Apakah Anda menganggap diri Anda perencana dan penyelenggara yang baik? Mengapa?*
- *Apa yang Anda lakukan jika manajer meminta Anda untuk menyelesaikan tugas, tetapi Anda sudah memiliki agenda yang sangat lengkap?*
- *Bagaimana Anda memprioritaskan pekerjaan Anda?*
- *Jelaskan pendekatan Anda untuk merencanakan dan mengatur pekerjaan Anda.*

Sekarang mari kita lihat contoh pertanyaan dan tanggapan wawancara.

Pertanyaan: Dapatkah Anda memberi kami contoh kapan Anda harus merencanakan dan mengatur acara penting atau aktivitas terkait pekerjaan? Langkah apa yang Anda ambil?

Ketika saya bekerja di unit pendukung administrasi untuk Michael Angelo Enterprises, saya bertanggung jawab untuk merencanakan berbagai kegiatan mulai dari pemesanan persediaan cat yang tepat waktu hingga keamanan, pemeliharaan gedung dan membantu departemen dan manajer dengan kebutuhan infrastruktur dasar. Menyulap semua aktivitas ini secara bersamaan berarti saya harus merencanakan pekerjaan saya dengan sangat detail dan juga terorganisir dengan sangat baik.

Ada suatu waktu ketika kami harus memasang sistem keamanan baru dan perangkat lunak grafik komputer baru, serta menjawab berbagai permintaan yang dibuat oleh klien kami. Untuk menangani semua ini, saya perlu membagi pekerjaan saya setiap hari, setiap minggu, dan setiap bulan dan memastikan bahwa saya terus mengikuti perkembangan terbaru tentang apa yang dilakukan orang lain.

Saya memastikan saya menghadiri sebanyak mungkin pertemuan dan tetap memperhatikan. Mengingat banyaknya tugas yang harus saya selesaikan, saya merasa penting untuk memprioritaskan pekerjaan saya sesuai dengan kebutuhan organisasi, dibandingkan dengan kebutuhan beberapa individu. Menyiapkan sistem keamanan baru harus dilakukan sebelum beberapa permintaan dibuat oleh manajer.

Dan, akhirnya, penting untuk mempelajari cara mengatakan 'tidak' untuk beberapa permintaan. Dalam pandangan saya, seorang perencana yang baik tahu seberapa banyak untuk membuatnya cukup. Mengambil lebih banyak pekerjaan daripada yang dapat ditangani hanya akan mengarah pada layanan berkualitas buruk atau bahkan kegagalan untuk melakukan pekerjaan itu.

Selain mempelajari banyak hal tentang apa yang diperlukan untuk mempertahankan organisasi dalam hal dukungan infrastruktur, salah satu hasil terbaik dari tindakan saya adalah bahwa klien saya menilai layanan saya sebagai 'sangat tinggi' selama tiga tahun berturut-turut, yang memberi saya kepuasan yang luar biasa.

Menjawab pertanyaan komunikasi interpersonal Keterampilan komunikasi interpersonal bukan hanya tentang komunikasi yang jelas. Mereka juga tentang cara kita berinteraksi dengan orang lain. Orang dengan keterampilan komunikasi interpersonal yang efektif lebih mungkin bergaul dengan orang lain di tempat kerja (dan dengan demikian maju) karena mereka menunjukkan berbagai perilaku yang menghasilkan yang terbaik dalam diri orang yang berinteraksi dengan mereka.

Mereka adalah pendengar yang baik, menghindari bahasa yang menghasut (termasuk bahasa komponen), mengakui kontribusi orang lain, berkonsultasi sebelum mengambil keputusan, dan sebagainya. Orang dengan keterampilan komunikasi interpersonal yang efektif sangat dihargai oleh pemberi kerja karena mereka membawa keharmonisan di tempat kerja. Mereka biasanya membuat orang merasa lebih baik tentang diri mereka sendiri dan kontribusinya — yang, tentu saja, penting bagi pemberi kerja dalam hal mempertahankan angkatan kerja yang bahagia dan produktif.

Berikut adalah beberapa pertanyaan tipikal keterampilan komunikasi antarpribadi:

- Apakah Anda senang bekerja dengan orang lain?
- *Bagaimana Anda menggambarkan hubungan Anda dengan orang lain di tempat kerja?*
- *Jelaskan tentang diri Anda. (Meskipun pertanyaan ini tidak terbatas pada keterampilan komunikasi antarpribadi, ini memberikan kesempatan yang sangat baik bagi Anda untuk menyebutkannya secara singkat.)*
- *Ceritakan tentang saat Anda berselisih dengan seseorang di tempat kerja. Bagaimana keadaannya dan bagaimana Anda menghadapinya?*
- *Dapatkah Anda memberi kami contoh ketika Anda harus mengomunikasikan masalah yang kompleks dan sensitif? Bagaimana Anda melakukannya?*
- *Jelaskan rekan kerja yang paling Anda sukai untuk bekerja.*
- *Bagaimana Anda menghadapi orang yang marah di tempat kerja?*
- *Apakah Anda lebih suka dilihat sebagai orang yang disukai atau orang yang efektif?*

Ada tumpang tindih yang jelas antara keterampilan komunikasi interpersonal dan keterampilan pemain tim. Oleh karena itu, banyak poin dapat digunakan secara bergantian. Berikut adalah contoh pertanyaan dan tanggapan wawancara yang mungkin.

Pertanyaan: Bisakah Anda memberi kami contoh ketika Anda harus mengomunikasikan masalah yang kompleks dan sensitif? Bagaimana Anda melakukannya?

Ketika saya bekerja untuk Magellan, saya berada di tim yang bertanggung jawab untuk memperkenalkan sistem penilaian kinerja baru untuk semua kru di kapal kami. Bekerja dalam proyek ini, saya sering diminta untuk mengkomunikasikan informasi yang kompleks dan sensitif kepada individu dan kelompok. Saya ingin menekankan bahwa penilaian kinerja adalah masalah yang sangat sensitif karena gaji orang dikaitkan dengan hasil.

Saya berhasil mengkomunikasikan informasi yang relevan karena saya menganut sejumlah prinsip komunikasi antarpribadi yang sehat — prinsip yang telah berhasil saya terapkan di masa lalu. Misalnya, saya mempertimbangkan kepekaan orang-orang dan mengatasinya sejak awal percakapan kita.

Saya menghindari segala bentuk jargon, dan sering berasumsi bahwa audiens saya memiliki sedikit sekali pengetahuan sebelumnya tentang masalah yang sedang dihadapi. Saya menggunakan bahasa komponen yang positif dan tidak mengancam — terutama ketika saya dihadapkan pada orang-orang skeptis yang meremehkan program tersebut meskipun mereka tidak memiliki pengetahuan tentangnya. Saya juga mengakui pendapat orang lain dan tidak pernah membuat komentar meremehkan tentang saran, tidak peduli betapa anehnya mereka.

Selain itu, saya selalu berusaha untuk berkonsultasi dengan pemangku kepentingan utama sebelum menyelesaikan keputusan. Fakta bahwa Anda berusaha

untuk berkonsultasi dan menjelaskan parameter di mana Anda harus bekerja sering meminimalkan tingkat ketidakpuasan, meskipun orang mungkin tidak sepenuhnya setuju dengan Anda.

Sebagai hasil dari usaha saya, tentangan terhadap program tersebut hampir tidak ada. Para kru menunjukkan sikap konstruktif dan memberikan yang terbaik. Hasilnya, kami berhasil melaksanakan program dalam jangka waktu dan anggaran kami.

Mengatasi perubahan di tempat kerja

Tidak seperti tempat kerja di masa lalu, ketika orang dapat melakukan serangkaian tugas yang sama selama bertahun-tahun, lingkungan kerja saat ini dicirikan oleh perubahan yang terus-menerus.

Faktanya, dapat dikatakan bahwa satu-satunya konstanta adalah perubahan. Semua ini, tentu saja, berarti karyawan yang fleksibel adalah karyawan yang sangat dihargai. Perubahan dapat terjadi dalam berbagai bentuk, termasuk:

- *Mesin baru;*
- *Prosedur atau pedoman baru;*
- *Legislasi baru;*
- *Struktur manajemen baru;*
- *Pengambilalihan perusahaan;*
- *Perampingan;*
- *Perangkat lunak baru;*
- *Efek persaingan baru.*

Organisasi yang tidak dapat beradaptasi dengan cepat terhadap keadaan yang berubah sering kehilangan pangsa pasar dan dapat dengan mudah keluar dari bisnis. Oleh karena itu, cara Anda menanggapi pertanyaan 'menghadapi perubahan' sangat penting. Berikut beberapa contoh bentuk yang mungkin mereka ambil:

- *Ceritakan kepada kami tentang saat Anda harus mempelajari hal-hal baru tentang pekerjaan Anda. Bagaimana Anda mengatasinya?*
- *Apakah Anda senang berganti tugas?*
- *Bagaimana Anda mengatasi perubahan konstan di tempat kerja?*
- *Apakah Anda menganggap diri Anda sebagai orang yang fleksibel?*
- *Menurut Anda, bagaimana reaksi Anda jika tiba-tiba harus meninggalkan proyek yang sedang Anda kerjakan dan memulai yang baru?*
- *Apa pandangan Anda tentang belajar di tempat kerja?*

Sekarang mari kita lihat contoh pertanyaan dan kemungkinan tanggapannya.

Pertanyaan: Ceritakan tentang saat Anda harus mempelajari hal-hal baru tentang pekerjaan Anda. Bagaimana Anda mengatasinya?

Ketika saya bekerja untuk Gedung Tembok Hadrian Legiun Utara, manajemen senior memutuskan untuk berinvestasi besar-besaran dalam teknologi baru yang dirancang untuk meningkatkan kualitas dan menghemat banyak waktu kami. Teknologi baru ini melibatkan serangkaian peralatan, perangkat lunak, dan prosedur kerja baru, dan mewakili perubahan besar dalam cara saya menjalankan tugas.

Awalnya, kami semua sedikit gentar dengan skala besar perubahan itu; namun, saya segera menyadari bahwa perubahan tidak dapat dihindari jika perusahaan kami ingin tetap kompetitif.

Saya juga segera menyadari bahwa, jika saya ingin tetap menjadi anggota perusahaan yang berharga, saya perlu segera belajar bagaimana bekerja di bawah rezim baru. Kesadaran ini memastikan bahwa saya menerima perubahan dengan antusias. Sementara beberapa kolega saya melihatnya sebagai beban, saya melihatnya sebagai cara masa depan, yaitu bagaimana saya melihat perubahan secara umum.

Selain menghadiri semua sesi pelatihan yang diperlukan, saya juga menghadiri sesi tambahan. Saya belajar dengan giat, mengajukan pertanyaan dan memperoleh pengalaman sebanyak yang saya bisa. Saya segera menjadi ahli yang diakui di bidang-bidang tertentu, dan orang-orang mulai mendatangi saya untuk meminta nasihat.

Sebagai hasil dari upaya kami, teknologi baru berhasil diterapkan. Tim saya bekerja dengan teknologi baru sesuai jadwal dan anggaran yang dialokasikan untuk kami. Dan saya belajar cara baru dalam melakukan sesuatu.

Ringkasan Poin-Poin Utama

Pentingnya lima pertanyaan besar adalah bahwa mereka didasarkan pada keterampilan yang dibutuhkan untuk sebagian besar, jika tidak semua, pekerjaan. Hal ini membuat Anda kemungkinan besar akan diminta untuk menjawab sejumlah pertanyaan yang berkaitan dengan ini. Selain universalitas keterampilan ini, keterampilan ini juga sangat penting bagi sebagian besar pemberi kerja (keterampilan komunikasi antarpribadi yang baik, misalnya, dipandang sebagai inti dalam membangun hubungan kerja yang harmonis dan kinerja yang efektif).

BAB 7
MEMBANGUN HUBUNGAN DAN KEPERCAYAAN

Mengelola persepsi dan pandangan yang terbentuk sebelumnya

Sebagian besar wawancara adalah tentang mengelola persepsi dari pewawancara. Studi menunjukkan bahwa orang mencari hal-hal yang mereka yakini (rasakan) akan ada di sana, dan sebaliknya mengabaikan atau kurang memperhatikan hal-hal yang tidak sesuai dengan pandangan mereka yang telah terbentuk sebelumnya.

Jika pewawancara berpikir bahwa kamu adalah prospek yang luar biasa, ada kemungkinan besar mereka akan mencari, dan mendaftarkan, semua hal yang akan mendukung gagasan mereka yang telah terbentuk sebelumnya.

Dengan kata lain, jika dua orang yang diwawancarai melakukan kurang lebih sama dalam sebuah wawancara, orang yang diwawancarai dengan reputasi yang lebih baik sebelum wawancara kemungkinan besar akan dinilai lebih tinggi.

Jadi, sebanyak mungkin, buat kesan terbaik yang kamu bisa sebelum atau di awal wawancara. Kamu dapat melakukannya dengan:

- *Memastikan resume Anda sebaik mungkin;*
- *Mengirimkan surat pra-wawancara yang positif dan sangat singkat yang berterima kasih kepada pewawancara atas kesempatan untuk diwawancarai dan menyatakan betapa Anda sangat menantikan untuk bertemu dengan mereka;*
- *Menghubungi perusahaan untuk mengajukan pertanyaan pra-wawancara yang masuk akal Menghubungi perusahaan sebelum wawancara menunjukkan minat yang sesuai dan tingkat persiapan profesional*

  Kesan pertama

Selain itu, penting untuk dicatat bahwa beberapa menit pertama (beberapa mengatakan detik) dari sebuah wawancara juga sangat penting dalam mempengaruhi pikiran pewawancara. Seperti pepatah lama, kesan pertama cenderung kesan abadi. Secara singkat (kita akan membahasnya lebih detail nanti), hal-hal yang perlu diperhatikan meliputi:

- *Pakaian*
- *Gerakan tangan*
- *Kontak mata*
- *Ekspresi wajah*
- *Nada suara*
- *Kesan terakhir*

Orang cenderung mengingat lebih banyak tentang apa yang terjadi di awal dan akhir suatu peristiwa daripada yang mereka lakukan di tengah. Ini tidak berarti kamu berkonsentrasi pada awal dan akhir wawancara dan mengabaikan bagian tengahnya.

Ini adalah pengingat untuk berhati-hati tentang apa yang kamu lakukan dan katakan menjelang akhir. Beberapa orang yang diwawancara jatuh ke dalam perangkap terlalu santai (biasanya sebagai akibat dari kompensasi berlebihan untuk ketegangan awal mereka) dan tersesat ke perilaku yang tidak pantas seperti menjadi terlalu akrab dan mengadopsi gaya bahasa komponen 'Saya sedang berada di barbekyu'. Jadi, pastikan kamu mempertahankan perilaku wawancara yang sesuai sampai akhir.

Komunikasi lebih dari sekedar kata-kata

Salah satu pelajaran terpenting yang dapat kamu pelajari tentang meningkatkan hubungan dan kemampuan kepercayaan adalah bahwa komunikasi lebih dari sekadar kata-kata yang keluar dari mulutmu.

Pakar komunikasi terus-menerus mengingatkan kita bahwa sekitar 10 persen komunikasi diwakili oleh apa yang kita katakan, 30 persen dengan cara kita mengatakan sesuatu, dan 60 persen oleh bahasa komponen kita! Jadi jika, dalam persiapanmu untuk wawancara, kamu telah menghabiskan seluruh waktumu berkonsentrasi pada isi jawabanmu, kamu telah secara efektif menghabiskan 100 persen dari upayamu pada 10 persen dari keseluruhan komunikasi. Ini mungkin sangat menjelaskan mengapa begitu banyak orang yang memberikan jawaban yang brilian secara teknis tidak mendapatkan pekerjaan itu.

Diakui banyak narasumber memahami secara intuitif bahwa komunikasi antarpribadi yang sukses (komunikasi tatap muka) bergantung pada lebih dari sekedar kata-kata yang digunakan. Namun, karena alasan yang terlalu beragam dan kompleks untuk dibahas di sini, ada banyak orang yang keterampilan komunikasi interpersonalnya tidak diasah dengan baik dan/atau yang tidak dapat menunjukkan keterampilan komunikasi mereka yang efektif selama wawancara — mungkin karena kecemasan yang meningkat.

Begitu kamu memahami bahwa komunikasi yang berhasil bergantung pada berbagai faktor selain kata-kata, dunia komunikasi yang sama sekali baru mulai muncul. Fokus persiapan wawancaramu harus bergeser dari persiapan kata yang ketat menjadi mencakup berbagai macam non-verbal termasuk hal-hal seperti penampilan, cara kamu duduk dan bahkan saat kamu menganggukkan kepala. Kadang-kadang senyuman ramah dan anggukan pengakuan bisa jauh lebih berharga daripada jawaban verbal terbaik.

Mengakui sumber dinamika

Dalam sebagian besar wawancara, ada dinamika penting namun tak terucapkan yang bersembunyi di balik permukaan. Dinamika ini sama tuanya dengan pertama kali manusia saling menatap dan membuka mulut untuk mendengus. Tentu, saya berbicara tentang kekuasaan.

Lebih khusus lagi, saya berbicara tentang mengakui fakta bahwa, dalam sebagian besar kasus, pewawancara memiliki kekuatan nyata. (Pengecualian untuk ini

adalah ketika kamu cukup beruntung memiliki seperangkat keterampilan dan/atau pengetahuan unik yang sangat ingin dimiliki oleh pemberi kerja.) Jika kamu serius ingin memaksimalkan hubunganmu, penting untuk menunjukkan kepada pewawancara bahwa kamu mengerti mereka memiliki semua kekuatan untuk memberimu pekerjaan.

Sebagai orang yang diwawancarai, kamu juga memiliki kekuatan — terutama melalui fakta bahwa kamu mengontrol apa yang akan didengar pewawancara. Namun, ini tidak menghilangkan kenyataan bahwa kekuatan untuk mempekerjakan (atau tidak) terletak secara eksklusif pada pewawancara.

Orang yang diwawancarai yang mengakui kekuatan pewawancara memiliki peluang yang lebih baik untuk disukai (dan karena itu memenangkan pekerjaan) karena, terus terang, kebanyakan manusia memiliki kelemahan untuk merasa penting dan ego mereka terbelenggu. Secara intuitif, banyak dari kita memahami dinamika ini tetapi tidak semua orang secara proaktif mendemonstrasikannya selama wawancara. Pewawancara mungkin bahkan tidak menyadari dinamika ini (kamu biasanya dapat memilih orang yang menikmati kekuatan mereka), tetapi bukan berarti tidak ada.

Hindari merendahkan -

Mengakui dinamika kekuatan yang melekat dalam sebagian besar wawancara tidak berarti merendahkan diri. Seperti yang telah disebutkan, melemparkan dirimu ke kaki pewawancara atau menertawakan dirimu sendiri dengan suara serak karena lelucon yang timpang kemungkinan besar akan dilihat sebagai bentuk penipuan.

Pelajaran di sini sederhana: waspadai dinamika kekuatan yang mendasari hadir di sebagian besar wawancara dan hindari perilaku (seperti berdebat suatu poin atau secara terbuka tidak setuju dengan pewawancara) yang kemungkinan besar akan membuat pewawancara pergi.

Masalah bahasa komponen

Duduk, Caramu duduk mengkomunikasikan banyak hal tentang berbagai macam masalah, termasuk seberapa penting menurutmu wawancara itu, seberapa gugup (atau percaya diri) kamu, dan pemahamanmu tentang hubungan kekuasaan yang mendasarinya. Posisi duduk beberapa orang memancarkan keakraban yang berlebihan dan bahkan arogansi, sedangkan yang lain menunjukkan kurangnya kepercayaan diri yang serius.

Aturan emas dalam duduk adalah: hindari apa pun yang akan mengalihkan perhatian pewawancara dari berkonsentrasi pada isi jawabanmu; dan hindari membuat pewawancara merasa tidak nyaman. Pewawancara umumnya tidak merasa nyaman jika kamu duduk dengan cara yang agresif (terlalu condong ke depan) atau dengan cara yang terlalu pasif (bersandar ke belakang dan menyilangkan kaki di paha). Singkatnya, posisi duduk yang baik tidak akan diperhatikan oleh pewawancara.

Berikut beberapa tip tentang apa yang harus Anda hindari:

- *Bersandar. Memberi kesan bahwa Anda tidak menanggapi wawancara dengan serius.*
- *Menyilangkan kaki Anda di paha. Terlalu familiar, terutama di awal wawancara.*
- *Duduk dengan kaki terbuka lebar. Terlalu familiar untuk situasi wawancara, dan bisa mengganggu dan tidak nyaman bagi pewawancara.*
- *Terlalu condong ke depan. Mungkin membuat beberapa pewawancara merasa tidak nyaman, terutama jika Anda secara fisik besar dan berbicara dengan keras.*
- *Bungkuk. Memberi kesan bahwa Anda tidak menanggapi wawancara dengan serius dan kemungkinan akan bungkuk dalam tugas Anda.*

Tips latihan duduk yang baik meliputi:

- *Komponen lurus dan tegak. Ini adalah posisi duduk netral yang diharapkan pewawancara.*
- *Kaki laki-laki. Laki-laki dapat menjaga kaki atas mereka menghadap lurus ke depan dan mengadopsi apa yang biasa disebut sebagai posisi awal yaitu, kaki dominan menapak di tanah dengan kaki lainnya hanya bagian depan yang menyentuh tanah.*
- *Kaki wanita. Betina bisa menyilangkan kaki di pergelangan kaki dan memposisikan kaki sedikit ke satu sisi.*
- *Ekspresi wajah dan kontak mata*

Ekspresi wajah adalah komunikator yang sangat kuat. Jika kamu duduk dengan benar, pewawancara harus menghabiskan sebagian besar wawancaranya dengan melihat wajah dan matamu.

Dua aturan emas duduk juga berlaku di sini: jangan melakukan apa pun yang akan mengganggu pewawancara atau membuat mereka merasa tidak nyaman. Apa pun yang berlebihan hampir pasti akan membuat pewawancara berhenti sejenak untuk khawatir, apakah itu terlalu banyak tersenyum, mengangguk, atau melakukan kontak mata.

Selama wawancara, sangat penting untuk mengontrol ekspresi wajahmu, terutama jika kamu merasa wawancara tersebut tidak mencapai kepuasanmu atau kamu mendengar sesuatu yang tidak kamu sukai — jika tidak, kamu mungkin menyampaikan informasi yang tidak diinginkan kepada pewawancara.

Kegagalan untuk mengontrol ekspresi wajahmu akan merusak kredibilitasmu dengan mengirimkan sinyal yang bertentangan kepada pewawancara.

Misalnya, pewawancara tiba-tiba memberi tahumu bahwa pekerjaan itu akan menyertakan tugas baru dan penting yang tidak disebutkan dalam iklan pekerjaan dan reaksi firasat mu adalah,

'Oh tidak, saya tidak mempersiapkan tugas baru ini, dan apa yang mereka lakukan mengubah pekerjaan pada tahap akhir ini dan saya tidak tahu apa-apa tentang tugas berdarah baru ini!' Tapi Anda berkata (atau mencoba mengatakan), 'Baru tugas, itu bagus. Saya terbiasa mengambil tugas baru. Saya adalah pembelajar yang cepat dan menikmati tantangannya. "

Dalam situasi ini, ada kemungkinan besar bahwa teror yang muncul di wajahmu akan merusak kata-katamu dan membuat pewawancara tidak yakin meskipun ada jawaban yang masuk akal.

Mengontrol ekspresi seseorang lebih sulit dilakukan daripada yang disadari banyak orang. Seringkali wajah kita bekerja secara independen dari keinginan kita. Dan biasanya mereka mengomunikasikan perasaan terdalam (tergelap) kita, yang mungkin bukan kepentingan terbaik kita untuk diungkapkan. Tapi dengan sedikit pengetahuan dan latihan kita bisa mengendalikan apa yang wajah kita katakan.

Menyadari kekuatan komunikatif ekspresi wajah merupakan awal yang baik untuk mengendalikan komunikasi yang tidak diinginkan. Lain kali kamu merasa bahwa wajahmu mungkin mengomunikasikan sesuatu yang tidak kamu inginkan, hentikan dan paksa dirimu untuk mengubahnya. Kamu mungkin akan merasa sedikit canggung pada awalnya, tetapi dengan sedikit ketekunan, kamu dapat mengendalikannya sesuka hati. Dengan latihan yang cukup, itu akan menjadi kebiasaan.

Tersenyum

Jika kamu berdiri di luar ruangan beberapa detik dari diundang untuk wawancara dan saya kebetulan lewat dan kamu menarik saya dengan tatapan putus asa di matamu meminta satu nasihat dari saya, saya akan berkata, 'Jangan lupa tersenyum '. Tersenyum adalah komunikator yang sangat efektif dan mengirimkan semua sinyal yang benar kepada pewawancara, terutama untuk membangun hubungan baik.

Senyuman sering kali dapat mencapai apa yang tidak bisa diperoleh jawaban terbaik — melembutkan pewawancara. Yang paling penting, ketika kamu tersenyum kepada orang lain, hal itu biasanya membuat mereka merasa lebih baik, yang cenderung menunjukkan sifat baik mereka persis seperti yang kamu ingin lakukan dalam wawancara. Ini juga menandakan kepada pewawancara bahwa kamu memiliki keterampilan sosial yang berkembang dengan baik, adalah orang yang baik dan tidak menderita kecenderungan anti-sosial.

Berikut beberapa tip tentang tersenyum:

- *Bersikaplah tulus.* Hindari menyeringai atau memaksakan senyum. Tidak ada yang lebih buruk dari seseorang yang mencoba tersenyum tetapi hanya berhasil menunjukkan seni mengatupkan gigi.
- *Jangan berlebihan*. Melakukannya secara berlebihan dapat membuatmu berisiko terlihat tidak jujur.
- *Hindari meniru pewawancara berwajah muram.*

Tidak jarang meniru ekspresi wajah (dan bahasa komponen) orang lain, meskipun kita sering tidak menyadari bahwa kita sedang melakukannya.

Jika kamu bertemu dengan pewawancara berwajah muram, cobalah untuk tidak jatuh ke dalam perangkap berwajah muram. Ini tidak semudah kedengarannya karena manusia, sebagaimana adanya kita, biasanya membutuhkan umpan balik positif untuk terus berperilaku dengan cara tertentu.

Dengan kata lain, jika kamu tersenyum dan orang lain menolak untuk membalasnya, ada kemungkinan kamu akan berhenti tersenyum. Jadi: jangan biarkan pewawancara yang buruk membuatmu marah. Tetap berpegang pada senjatamu dan tunjukkan senyum terhangatmu, apa pun yang terjadi!

Mengangguk

Mengangguk-angguk mewakili komunikator lain yang sangat kuat. Ketika kamu menganggukkan kepala pada sesuatu, orang mengatakan kamu mengatakan bahwa kamu setuju dengan mereka, dan kamu melakukannya tanpa menyela, yang merupakan teknik membangun hubungan yang ideal ketika pewawancara memutuskan untuk menjelaskan suatu topik. Tapi hati-hati: seperti dalam tersenyum, bahayanya menganggukkan kepala adalah berlebihan.

Kontak mata

Kunci kontak mata yang sukses adalah menghindari hal-hal ekstrem. Melakukannya secara berlebihan dapat membuat orang menjauh, seperti halnya hampir tidak membuat kontak mata sama sekali. Menatap hampir pasti akan menimbulkan tanda tanya besar tentang keterampilan sosialmu. Lebih buruk lagi, itu mungkin membuat takut pewawancara.

Tidak cukup melakukan kontak mata kemungkinan besar akan menandakan bahwa kamu kurang percaya diri dan mungkin menderita masalah harga diri yang rendah. Ingatlah bahwa wawancara sebagian besar tentang menyampaikan kesan. Kamu mungkin pada kenyataannya adalah orang yang percaya diri dan ramah yang menikmati kehidupan sosial yang hebat, tetapi jika kamu gagal melakukan kontak mata yang cukup dengan pewawancara, kamu mungkin akan gagal untuk mengkomunikasikan kenyataan itu.

Seperti kebanyakan komunikator non-verbal, tingkat kontak mata yang sesuai selama wawancara berbeda antar budaya. Penting bagimu untuk memastikan norma budaya sebelum melakukan wawancara.

Tangan dan lengan

Kesalahan besar dengan lengan adalah melipatnya di dadamu. Melakukannya sama saja dengan menempatkan penghalang antara kamu dan pewawancara. Pelanggaran lainnya termasuk duduk di atas tanganmu atau berpura-pura tidak memilikinya. Tidak ada salahnya menggunakan tanganmu untuk menekankan suatu

hal yang menunjukkan bahwa kamu adalah manusia. Namun, hindari melakukannya secara berlebihan.

Jabat tangan

Jabat tangan yang baik adalah jabat tangan yang kuat. Jika kamu seorang pria muda, hindari dorongan utama untuk menghancurkan tulang tangan pewawancara. Ingatkan dirimu bahwa tujuan berjabat tangan adalah untuk membangun hubungan, bukan untuk menunjukkan seberapa kuat kamu. Hindari juga jabat tangan yang lemas, jabat tangan yang panjang (ingat untuk melepaskan) dan jabat tangan dengan tiga jari. Jika kamu sangat menderita karena telapak tangan berkeringat, bawalah sapu tangan, tetapi jika kelenjar keringatmu sedang kerusuhan, ada baiknya untuk memperingatkan pewawancara terlebih dahulu sebelum membasahi telapak tangan mereka.

Gaun dan penampilan

Beberapa orang tetap berpikir bahwa penampilan mereka tidak ada hubungannya dengan kemampuan mereka untuk tampil dalam suatu pekerjaan, sehingga memberikan sedikit pertimbangan tentang cara mereka berpakaian untuk wawancara.

Meskipun logika dalam pemikiran ini mungkin tidak dapat disangkal, ini adalah hal yang berbahaya untuk dilakukan karena gagal memperhitungkan bahwa wawancara sebagian besar adalah tentang mengelola persepsi. Pewawancara memiliki ekspektasi tertentu tentang kode pakaian. Gagal memenuhi ekspektasi itu berbahaya.

Aturan praktis untuk pakaian dan penampilan adalah melakukan kesalahan di sisi hati-hati. Secara keseluruhan, pewawancara cenderung berhati-hati dan konservatif saat mempekerjakan seseorang.

Hal terakhir yang ingin dilakukan pemberi kerja adalah mempekerjakan orang yang salah. Keandalan, loyalitas, konsistensi, kepercayaan, dan ketergantungan adalah kualitas yang dicari semua pemberi kerja pada karyawan, apa pun jenis pekerjaannya. Tugasmu saat wawancara adalah memberi isyarat kepada pewawancara bahwa kamu memiliki semua kualitas itu, dan berpakaian dengan tepat merupakan awal yang baik.
Berikut beberapa tip:

- *Selalu pastikan untuk mengenakan pakaian dan sepatu bersih.*
- *Jeans (atau apa pun) dengan lubang di dalamnya mungkin membuat kesan positif di lantai dansa, tetapi kemungkinan tidak akan menginspirasi pewawancara.*
- *Hindari perhiasan dan riasan berlebihan.*
- *Tunggul seorang desainer mungkin membuat Anda terlihat gagah dan mewakili kata-kata terbaru dari pakar mode; namun, pewawancara kemungkinan akan berpikir bahwa menurut Anda pekerjaan itu tidak cukup penting sehingga Anda tidak perlu repot-repot bercukur.*
- *Hindari gaya rambut yang ekstrim.*
- *Hindari menampilkan terlalu banyak kulit.*

Ada pemikiran masuk akal yang menganjurkan berpakaian sesuai dengan sifat pekerjaan yang kamu lamar. Jadi, jika kamu melamar posisi akuntan, kamu mengenakan setelan bisnis, sedangkan jika kamu melamar posisi buruh di lokasi bangunan, setelan bisnis tidak pantas.

Semua ini benar; Namun, tips berpakaian dan penampilan di atas tetap penting.

Perilaku wawancara

Bahasa komponen dan penampilan pribadi mewakili satu sisi persamaan untuk membangun hubungan dan kepercayaan selama wawancara. Sisi lain yang sama pentingnya adalah bagaimana kamu bersikap dan mengekspresikan diri selama wawancara.

Jangan pernah membantah

Salah satu hal terburuk yang dapat kamu lakukan saat wawancara adalah berdebat dengan pewawancara. Bahkan argumen yang sangat sopan tidak boleh dipertimbangkan. Perdebatan kemungkinan besar akan meyakinkan pewawancara bahwa kamu pada dasarnya argumentatif, yang bukan merupakan sifat yang menggairahkan pemberi kerja.

Ini adalah poin yang dilupakan oleh beberapa orang yang diwawancarai — terutama ketika mereka yakin bahwa mereka benar atau pewawancara mengatakan sesuatu yang ternyata salah.

Juga, beberapa pewawancara (biasanya yang tidak berpengalaman) cenderung meremehkan beberapa hal yang mereka katakan dan menambahkan informasi mereka sendiri atau bahkan membuat koreksi (atau apa yang mereka yakini sebagai koreksi). Menghadapi pewawancara jenis ini bisa menjadi pengalaman yang sangat membuat frustrasi. Pada saat-saat seperti inilah senyummu bisa berubah menjadi seringai dan seluruh komponenkamu terlihat siap untuk memulai pertempuran.

Namun, orang yang diwawancarai yang efektif akan menjaga disiplin dan terus tersenyum, mengangguk dengan gembira dan mengucapkan kata-kata kecil seperti, 'Ya, itu benar,' dan 'Saya sangat setuju'.

Kamu mungkin berpikir, 'Saya tidak akan pernah ingin bekerja untuk pewawancara yang sangat tidak menyenangkan, jadi mengapa saya harus begitu menyenangkan?'

Meskipun ini bukan pemikiran yang tidak masuk akal, ada alasan bagus untuk mengabaikannya:

- *Pewawancara mungkin bukan pemberi kerja atau atasan langsung Anda.*
- *Pewawancara yang buruk tidak selalu membuat majikan yang buruk.*
- *Pewawancara mungkin tidak berpengalaman, gugup atau mengalami hari yang buruk.*

Selalu lakukan yang terbaik saat wawancara, tidak peduli betapa tidak menyenangkannya kamu menganggap pewawancara. Seluruh gagasan menghadiri wawancara akan ditawari pekerjaan. Terserah kamu, apakah kamu menerima tawaran atau tidak nanti.

Hindari hal-hal negatif

Tidak ada gunanya menghadiri wawancara jika kamu akan duduk di sana dan menyoroti banyak kekurangan dan kekuranganmu. Berikut adalah beberapa contoh pernyataan negatif yang membuat pewawancara merunduk untuk berlindung:

- 'Saya akan dapat menyelesaikan proyek jika saya tidak bentrok dengan rekan satu tim saya.' (Anda mungkin telah bekerja dengan rekan satu tim dari neraka, tetapi pewawancara kemungkinan akan mempertanyakan kemampuan pemain tim Anda.)
- 'Saya suka bekerja di Perusahaan perangkat lunak, tetapi terkadang pertanyaan pelanggan atau Bos saya membuat saya gila.'
- 'Saya biasanya menikmati mengelola orang kecuali ketika mereka mulai mengeluh tentang pekerjaan mereka.
- Saya tidak suka merengek. "(Kebanyakan orang mengeluh tentang pekerjaan dari waktu ke waktu, tugas manajer yang baik adalah mendengarkan dan membantu, bukan menganggap staf sebagai pengeluh.)
- "Saya tidak suka hal-hal berubah setiap saat. Tepat ketika Anda mempelajari satu hal, Anda perlu melepaskannya dan mempelajari sesuatu yang berbeda. Ada terlalu banyak ketidakstabilan di beberapa tempat kerja. '(Kecuali jika Anda melamar pekerjaan langka yang keadaannya selalu sama, jawaban yang diberikan dengan laju perubahan yang cepat saat ini dapat dengan mudah memasuki hall of fame untuk jawaban yang buruk.)
- 'Saya tidak suka tekanan.' (Hindari yang ini kecuali Anda melamar pekerjaan fantasi yang Anda buat di kepala Anda.)
- 'Saya tidak suka diberi tahu apa yang harus dilakukan.' (Anda harus memberikan pertimbangan serius untuk memulai bisnis Anda sendiri.)
- 'Saya menderita tingkat stres yang tinggi, jadi saya membutuhkan pekerjaan yang bebas stres.' (Pekerjaan fantasi lain.)
- 'Saya tidak suka bekerja lembur.' (Banyak orang tidak suka bekerja lembur tetapi itu bukan hal yang harus dikatakan dalam wawancara. Kecuali jika komitmen mendesak tidak mengizinkan Anda, sebagian besar pekerjaan mengharuskan orang untuk tetap bekerja. kembali kadang-kadang.)
- 'Saya kesal ketika orang tidak mengerti apa yang saya bicarakan.' (Mungkin Anda punya masalah komunikasi.)
- "Saya tidak tahu mengapa, tetapi orang-orang tampaknya takut pada saya." (Mungkin Anda punya masalah terkait dengan orang lain.)
- 'Aku lambat belajar.'

Pernyataan negatif sangat menakutkan bagi pewawancara mengingat, mereka adalah kelompok yang konservatif. Menjadi kritis tentang kinerja masa lalumu sama saja dengan memberi pewawancara alasan untuk tidak mempekerjakan kamu. Juga, pernyataan negatif karena membuat takut pewawancara cenderung mengundang pertanyaan lanjutan, yang merupakan hal terakhir yang kamu inginkan terjadi dalam wawancara.

Ide keseluruhannya adalah mengatakan hal-hal yang akan mengundang pertanyaan positif, yaitu pertanyaan yang memungkinkan kamu berbicara tentang semua kekuatan dan pencapaian luar biasa kamu. Beberapa orang berpikir bahwa menunjukkan hal-hal negatif adalah cara untuk menunjukkan kejujuran mereka kepada pewawancara. Sayangnya bagi mereka, pewawancara hanya akan memikirkan cara untuk mengakhiri wawancara.

Selain hal-hal yang akan memiliki pengaruh langsung pada pekerjaan (seperti masalah dalam pekerjaan yang membutuhkan pekerjaan berat), kelemahanmu mungkin bukan urusan siapa pun. Apa yang kamu anggap sebagai kelemahan diri sendiri mungkin tidak dianggap satu oleh orang lain. Pada akhirnya, wawancara adalah tentang membuat kesan terbaik.

Mengatasi kekurangan

Tidak membicarakan hal negatif berbeda dengan berbicara tentang mengatasi kekurangan. Bagi banyak orang yang berprestasi tinggi, pekerjaan sebagian besar adalah tentang mengatasi kekurangan dalam keterampilan dan pengetahuan mereka untuk mencapai tujuan mereka. Alih-alih takut dengan hal-hal baru, mereka menganggapnya sebagai tantangan belajar dan berharap untuk mengatasinya.

Seringkali perbedaan antara karyawan yang sangat efektif dan karyawan yang berjuang tidak ada hubungannya dengan bakat dan lebih berkaitan dengan sikap belajar ini.

Pengusaha tidak lebih suka mendengar tentang bagaimana kamu mengatasi kekurangan keterampilan atau pengetahuan untuk menyelesaikan sebuah proyek. Mengatasi kekurangan menunjukkan kepada pewawancara bahwa kamu adalah tipe orang yang mampu belajar di tempat kerja dan, sebagai hasilnya, menyelesaikan pekerjaan. Berikut adalah jawaban untuk 'mengatasi kekurangan keterampilan/pengetahuan':

- Setelah menerima penugasan, kami segera menyadari bahwa beberapa dari kami di tim tidak memiliki pengetahuan yang dibutuhkan untuk memaksimalkan kontribusi kami. Kekurangan saya adalah memahami bagaimana menggunakan beberapa aplikasi perangkat lunak yang rumit yang sangat penting untuk sisi kontrol kualitas dari tugas tersebut.
- Tantangan saya adalah mempelajari cara menggunakan aplikasi ini dalam waktu yang sangat singkat dan menerapkan pengetahuan ini dengan andal. Karena kami bekerja di bawah garis waktu yang sangat ketat dan anggota tim lainnya mengandalkan saya, margin kesalahannya sangat kecil.
- Untungnya, saya dapat menerapkan pengetahuan yang baru saya temukan, seperti yang dilakukan anggota tim lainnya, dan kami berhasil menyelesaikan tugas tersebut.
- Jawaban ini tidak hanya memberi tahu pemberi kerja bahwa kamu dapat mempelajari informasi yang rumit saat mengerjakan tugas, tetapi Anda juga dapat melakukannya di bawah tekanan dan memberikan hasil yang diminta.

Menangani pertanyaan kelemahan: Apa yang tidak boleh dilakukan

Pertanyaan 'Apa kelemahan Anda?' Bukanlah pertanyaan yang ideal untuk ditanyakan oleh pewawancara. Beberapa masalah yang melekat dalam pertanyaan ini meliputi:

- *Banyak narasumber tidak menyadari bahwa mereka memiliki kelemahan sejak awal.*
- *Ada pula yang merasa memiliki kelemahan, tetapi sebenarnya tidak memiliki kelemahan sama sekali.*
- *Beberapa orang yang diwawancarai secara keliru melihat pertanyaan ini sebagai kesempatan untuk menunjukkan seberapa jujur mereka dan mengatakan lebih dari yang seharusnya.*
- *Banyak orang yang diwawancarai sangat enggan untuk mengungkapkan kelemahan mereka dalam sebuah wawancara.*

Terlepas dari masalah ini, banyak pewawancara tetap bertanya tentang kelemahanmu. Tugasmu adalah mempelajari cara terbaik untuk menangani pertanyaan semacam itu. Paling tidak, kamu harus meminimalkan potensi kerusakan dan paling baik kamu harus membalikkan pertanyaan dan menunjukkan kepada pewawancara bahwa kamu adalah tipe orang yang tidak hanya dapat mengatasi kelemahan, tetapi dengan melakukannya mencapai tujuanmu.

Salah satu hal terburuk yang dapat kamu lakukan untuk menjawab pertanyaan ini adalah mengatakan kamu tidak memiliki kelemahan apa pun. Ini akan memberi sinyal kepada pewawancara bahwa kamu telah kehilangan sebagian peganganmu pada kenyataan dan/atau bahwa kamu memiliki ego yang mengerikan, yang keduanya tidak akan membantumu. Berikut beberapa hal lain yang harus dihindari:

- *Jangan menawarkan lebih dari satu kelemahan dan jangan memulai wacana monumental tentang kegagalan Anda dan kemungkinan asal-usulnya. Tetap berpegang pada satu kelemahan kecuali didesak sebentar.*
- *Hindari berbicara tentang kelemahan tipe kepribadian/karakter seperti ketidaksabaran, cepat marah atau intoleransi terhadap kesalahan.*
- *Secara umum, jenis kelemahan ini lebih menakutkan pengusaha daripada kekurangan keterampilan. Dimana yang terakhir biasanya dapat diperbaiki dengan sedikit pelatihan, kelemahan tipe kepribadian/karakter mungkin kurang mudah untuk diperbaiki dan lebih sulit untuk ditangani.*
- *Hindari kalimat klise seperti: 'Saya bekerja terlalu keras. Saya tidak tahu kapan harus berhenti. Saya tidak tahu bagaimana mengatakan tidak pada permintaan kerja. 'Masalah dengan jawaban ini ada dua: pertama, banyak orang lain menggunakannya, yang berarti Anda gagal menonjol dari yang lain; dan kedua, semua jawaban di atas mungkin memberi sinyal kepada pewawancara yang cerdik bahwa Anda memiliki masalah serius dalam mengelola beban kerja Anda.*

- *Jangan menyebutkan hal-hal yang benar-benar akan menyakiti Anda. Kesalahan yang Anda buat di masa lalu yang redup harus tetap ada di masa lalu.*
- *Jangan langsung memahaminya — terutama jika Anda telah mempelajari kesalahan cara Anda dan telah pindah.*

Mudah-mudahan, kamu tidak akan melamar pekerjaan yang tidak cocok untukmu. Jika, misalnya, kamu sangat takut akan ketinggian dan sebagian pekerjaan melibatkan bekerja di lokasi yang tinggi, maka kamu tidak boleh menyia-nyiakan waktu siapa pun dengan melamar.

Namun, jika pekerjaan yang sama juga membutuhkan keterampilan yang kamu miliki secara berlimpah, silakan hubungi mereka terlebih dahulu dan beri tahu mereka tentang situasimu. Majikan mungkin menghargai keterampilan lain itu dan setidaknya bersedia berbicara dengan mu.

Peringatan: Jika kamu telah melakukan pelanggaran hukum yang mungkin relevan dengan pekerjaan yang kamu lamar, kamu harus menyelidiki apa kewajiban hukummu dalam hal pengungkapan sebelum menghadiri wawancara. Hindari desas-desus. Undang-undang pengungkapan terkadang berubah dan mungkin berbeda dari satu negara bagian ke negara bagian lain.

Berurusan dengan pertanyaan kelemahan: Apa yang harus dikatakan

Cara efektif untuk menangani pertanyaan kelemahan adalah dengan menemukan kelemahan (lebih disukai kekurangan keterampilan) pada suatu waktu di masa lalu dan kemudian menjelaskan langkah-langkah yang kamu ambil untuk mengatasinya (mirip dengan mengatasi kekurangan, lihat di atas). Idenya adalah kamu menunjukkan kepada pewawancara bahwa kamu mampu mengatasi kelemahanmu. Ada baiknya juga untuk mencoba menyelesaikan jawabanmu dengan nada positif. Seperti inilah suara pertukaran itu.

Pertanyaan: Ceritakan tentang kelemahan Anda

Ketika saya bekerja untuk Chaos beberapa tahun lalu, salah satu kelemahan saya adalah di bidang membuat presentasi kepada klien dan staf internal. Bukan berarti presentasi saya jauh dari bencana — tetapi presentasi saya kurang dipoles oleh presenter lain yang lebih berpengalaman.

Jadi saya mendekati seorang presenter yang gayanya saya kagumi dan bertanya apakah dia bisa memberi saya beberapa tip tentang bagaimana saya bisa meningkatkan keterampilan saya. Untungnya, dia sangat senang membantu saya, termasuk duduk di salah satu presentasi saya dan memberi saya umpan balik tentang kelemahan saya. Saya menerima umpan baliknya dan membuat beberapa perubahan, yang menyebabkan presentasi saya meningkat secara signifikan.

Jika pewawancara tidak senang dengan jenis jawaban ini karena gagal berbicara tentang kelemahan saat ini, berikan saja kelemahan berbasis keterampilan yang tidak akan merusak peluangmu untuk memenangkan pekerjaan — dengan kata lain, kelemahan yang tidak sangat relevan dengan pekerjaan itu.

Menangani Keberatan

Keberatan pemberi kerja biasanya berupa 'Saya menyukaimu tapi. . .' Statements. Misalnya, 'Saya menyukaimu, tetapi perhatian utama saya adalah bahwa sebagian besar pengalaman mu terletak di ritel yang tidak relevan dengan kebutuhan kita.' kamu akan paling sering menghadapi keberatan saat pergi untuk promosi atau pekerjaan di berbagai pekerjaan atau industri. Meskipun tidak ada satu cara yang benar untuk menangani keberatan, ada metode tiga langkah yang mungkin berguna bagi Anda:

Setuju dengan keberatan: ‘Ya, itu benar. Sebagian besar pengalaman saya memang terletak pada retail. 'Menyetujui cenderung melunakkan interaksi penonton. Tidak setuju mungkin akan membuat Anda terdengar tidak masuk akal, jika tidak putus asa.

Nyatakan mengapa menurut Anda keberatan tersebut tidak menunjukkan masalah: ‘Saya ingin menunjukkan bahwa secara eceran sebagian besar pekerjaan yang saya lakukan secara langsung relevan dengan pekerjaan ini. Padahal industrinya tidak sama ketrampilannya. Misalnya, keterampilan yang dibutuhkan dalam memberikan layanan pelanggan tingkat tinggi dan menyelesaikan keluhan pelanggan adalah sama dengan yang Anda butuhkan. "

Tegaskan bahwa perbedaan bukanlah masalah dan selesaikan dengan catatan positif: ‘Sebenarnya, saya melihat membawa perspektif baru ke bisnis Anda sebagai keuntungan. Saya yakin saya dapat memperkenalkan ide-ide baru yang akan memajukan bisnis Anda. "

Hindari ketidakpastian

Salah satu aturan emas dalam wawancara adalah menghindari keraguan atau keraguan sebanyak mungkin. Mengatakan kamu dapat mencapai sesuatu dengan ragu-ragu dalam suaramu atau menggunakan bahasa tentatif hampir sama dengan mengatakan bahwa kamu tidak dapat benar-benar melakukannya. Jauhi ekspresi seperti:

- *Saya pikir Saya dapat...*
- *Saya tidak yakin tentang itu tapi mungkin...*
- *Mungkin saya dapat...*
- *Saya rasa Saya akan mampu...*

Keyakinan adalah salah satu kunci untuk membangun hubungan baik dalam sebuah wawancara. Pewawancara senang mendengar jawaban yang meyakinkan karena itu membantu mereka mengatasi keraguan mereka tentang kemampuan orang yang diwawancarai. Bahkan jika kamu ditanyai tentang tugas yang belum pernah kamu lakukan sebelumnya, lebih baik mengatakan kamu belum pernah melakukannya tetapi merasa yakin untuk menyelesaikannya karena semua keterampilan dan pengetahuan yang kamu bawa ke pekerjaan itu, daripada mengaku tidak pernah melakukan tugas dan mengungkapkan serangkaian ketidakpastian.

Ingat, bagaimana kamu mengatakan sesuatu lebih penting daripada apa yang kamu katakan. Bandingkan jawaban dari dua kandidat berikut, yang keduanya menjawab pertanyaan 'Menurut Anda, bagaimana Anda akan mengatasi pengelolaan tim profesional?'

Kandidat satu: Saya tidak sepenuhnya yakin apakah saya dapat mengelola tim profesional. Saya belum pernah melakukannya sebelumnya, jadi ini akan menjadi pengalaman yang sama sekali baru bagi saya, tetapi saya rasa dengan sedikit aplikasi, saya dapat mengelolanya. Tentu saja saya ingin mencobanya. Ini adalah bidang yang sangat saya minati.

Kandidat dua: Saya yakin saya bisa melakukan pekerjaan dengan baik. Saya merasa nyaman bekerja dengan orang yang berprestasi tinggi, saya memiliki keterampilan komunikasi interpersonal yang baik dan mengelola orang adalah bidang yang sangat saya minati. Meskipun saya belum pernah mengelola tim sebelumnya, saya merasa siap untuk menghadapi tantangan baru ini karir saya.

Intinya, kedua kandidat di atas mengatakan hal yang sama. Keduanya mengaku tak punya pengalaman mengelola tim profesional, namun keduanya tertarik mengemban tanggung jawab baru ini. Namun, awal dari jawaban kandidat pertama mungkin akan membuat mereka kehilangan pekerjaan. Saya ragu banyak pewawancara akan dengan serius mendengarkan apa pun setelah kalimat fatal pertama itu.

Ada upaya untuk memulihkan dalam dua kalimat terakhir tetapi sudah terlambat saat itu. Di sisi lain, calon kedua membangkitkan rasa percaya diri sejak awal. Tidak ada ketidakpastian sama sekali dalam jawaban ini, meskipun kandidat tersebut mengaku tidak memiliki pengalaman dalam mengelola profesional.

Banyak orang yang diwawancarai kesulitan menggunakan bahasa yang sangat positif saat berbicara tentang tugas yang belum pernah mereka lakukan sebelumnya. Ini bukan hal yang aneh, mengingat dalam konteks non-wawancara kebanyakan orang menggunakan bahasa tentatif ketika berbicara tentang hal-hal yang belum pernah mereka coba sebelumnya. Tujuan kamu harus meninggalkan semua tentativeness di luar pintu wawancara. Jika kamu tidak akan percaya diri melakukan pekerjaan dengan baik, bagaimana kamu mengharapkan pewawancara percaya diri tentang dirimu?

Pernyataan positif

Cara efektif untuk membiasakan diri menggunakan bahasa positif adalah dengan berlatih menggunakan pernyataan positif sebelum wawancara. Buat daftar pernyataan positif yang relevan dengan situasimu dan mulailah mengucapkannya dengan lantang. Kamu mungkin merasa sedikit canggung pada awalnya, tetapi pengulangan akan segera menyelesaikannya.

Teruslah berlatih sampai kamu merasa sangat nyaman. Berikut beberapa langkah awal untuk membantumu memulai:

- *Saya pasti bisa melakukan/menyelesaikan/menulis/menganalisis ...*
- *Saya yakin tentang ...*
- *Saya merasa sangat nyaman dengan prospek ...*
- *Saya sangat yakin mengetahui bahwa ...*
- *Saya merasa nyaman melakukan semua hal yang Anda sebutkan ...*
- *Saya selalu bersikap positif tentang mengambil ...*
- *Pernyataan orang ketiga.*

Alih-alih menggunakan pernyataan orang pertama (pernyataan 'Saya') sepanjang waktu, seperti 'Saya melakukan ini dan itu. . . 'Dan' Saya sangat ahli dalam. . . ', Seringkali lebih baik menggunakan pernyataan orang ketiga. Keuntungan dari jenis pernyataan ini adalah memungkinkan kamu mengutip apa yang orang lain katakan tentang pencapaianmu, bukan apa yang kamu pikirkan. Berikut beberapa contohnya:

- Atasan saya sering berkomentar tentang seberapa cepat saya dapat menyelesaikan pekerjaan saya. (berbeda dengan ‘Saya sering kali dapat menyelesaikan pekerjaan saya dengan sangat cepat’)
- Rekan-rekan saya, dengan sangat murah hati, memilih saya sebagai pemain tim yang paling berharga.
- Klien sering memberi saya umpan balik positif tentang keterampilan layanan pelanggan saya.
- Tim tempat saya bekerja secara konsisten memberi saya nilai tertinggi untuk keterampilan komunikasi pribadi saya dan kesediaan untuk membantu orang lain.

Puji orang lain.

Terlepas dari apa yang mungkin dipikirkan sebagian orang, dalam sebagian besar kasus, menyelesaikan sesuatu di tempat kerja membutuhkan bantuan dan kerja sama dari orang lain. Mengakui masukan berharga dari orang lain dalam hal pencapaianmu adalah cara yang bagus untuk mencapai kerendahan hati dalam wawancara. Jawabannya mungkin seperti ini: Berhasil menyelesaikan proyek tepat waktu dan sesuai anggaran sangat berarti bagi perusahaan saya. Seandainya saya tidak mengantarkan barang, ada kemungkinan orang dibuat mubazir.

Namun, saya ingin menekankan bahwa satu-satunya alasan saya sukses adalah karena bantuan berharga yang saya terima dari rekan-rekan saya. Tanpa dukungan mereka yang tak kenal lelah, saya akan gagal. Tanpa mengakui masukan dari kolega, jawaban ini berisiko terdengar agak arogan, tetapi memuji orang lain memastikan bahwa pembicara tampil sebagai orang yang rendah hati tanpa mengurangi nilai pencapaiannya.

Hindari mengulangi pencapaian utamamu. Dalam konteks sosial yang normal, kami tidak suka orang-orang melanjutkan pencapaian mereka, adnauseum satu penyebutan secara umum sudah cukup. Hal yang sama berlaku dalam wawancara. Meskipun penting bagimu untuk mempelajari cara membicarakan pencapaian utamamu, Kamu sebaiknya hanya menyatakan pencapaian satu kali. Jika kamu mengulanginya, kamu berisiko memberi kesan bahwa kamu tidak memiliki banyak hal untuk dibicarakan atau kamu pamer.

Hindari diri sendiri yang 'mencatat banyak hal'. Ini mungkin terdengar sedikit aneh berasal dari buku keterampilan wawancara, tetapi sangat penting jika kamu ingin menghindari menggambarkan dirimu terlalu besar untuk sepatu bot kamu. 'Memperhatikan' diri sendiri berarti benar-benar mengatakan bahwa kamu baik atau hebat, atau deskripsi lain yang ingin kamu pilih — misalnya, 'Saya seorang komunikator yang hebat'. Pewawancara harus menyimpulkan hal ini dengan mendengarkanmu berbicara tentang hal-hal yang telah kamu lakukan di area ini. Dengan kata lain, alih-alih mendeskripsikan diri sendiri,

Katakan apa yang kamu lakukan dan bagaimana kamu melakukannya dan biarkan tindakan itu berbicara sendiri. Berikut beberapa contohnya:

- *Hindari*: Saya adalah manajer orang yang hebat.
- *Katakan:* Dengan menerapkan prinsip-prinsip manajemen orang yang baik, saya dapat memimpin tim saya secara efektif.
- *Hindari:* Saya memiliki keterampilan layanan pelanggan yang hebat.
- *Katakan:* Manajer saya sering memuji saya atas keterampilan layanan pelanggan saya, khususnya pemahaman saya tentang produk kita dan kemampuan saya untuk menghubungkan pengetahuan ini dengan kebutuhan pelanggan kita.
- *Hindari:* Saya pekerja keras.
- *Katakan:* Dalam pekerjaan saya sebelumnya, saya selalu memastikan pekerjaan dilakukan dengan benar sebelum saya pulang. Jika itu berarti tetap di belakang, maka itulah yang saya lakukan.
- *Hindari mengkritik atasan.* Kita semua tahu bahwa ada manajer yang biasa-biasa saja hingga manajer yang buruk di luar sana, dan tidak diragukan lagi banyak pewawancara mengalami kesialan bekerja untuk mereka.

Meskipun demikian, aturan emas lainnya adalah: jangan pernah mengkritik atasanmu. Alasannya sederhana: pewawancara tidak memiliki manfaat untuk mendengarkan kedua sisi cerita dan oleh karena itu tidak dalam posisi untuk mengetahui siapa yang sebenarnya bersalah. Dengan kata lain, ketika kamu mengkritik atasanmu, kamu secara efektif menciptakan keraguan tentang dirimu sendiri di benak pewawancara. Mengkritik lebih dari satu bos adalah bunuh diri wawancara virtual.

Jika kamu berada dalam situasi di mana kinerja buruk atasanmu menghalangi kamu untuk mencapai pencapaian utama dan kamu dihadapkan dengan pewawancara yang gigih yang bersikeras untuk mengetahui seluk beluk apa yang terjadi, alih-alih melontarkan sesuatu yang kritis tentang atasanmu, seperti 'Kami tidak mencapai target kami karena pemimpin tim kami tidak dapat mengikat tali sepatunya', kamu dapat mencoba sesuatu seperti ini:

Sayangnya kami gagal mencapai target kami. Salah satu alasannya adalah karena beberapa anggota tim kami tidak memiliki pengalaman yang diperlukan untuk mengatasi beberapa kendala yang kami temui. Seandainya kami memiliki pengalaman yang tepat,

- *Saya yakin kami akan berhasil.*
- *Hindari mengatakan apa pun yang terdengar seperti berikut ini:*
- *Saya memiliki bos yang buruk.*
- *Bos saya adalah seorang Nazi sejati.*
- *Bos saya terlalu banyak omong*
- *Bos saya bodoh*
- *Saya tidak tahan bos saya dan dia tidak tahan dengan saya.*
- *Saya tidak akan memberi makan mantan bos saya.*
- *Bos saya menderita IQ yang sangat rendah.*
- *Tidak ada yang menyukai bos saya karena dia terlihat seperti monyet.*
- *Rekrut suara Anda*

Orang yang diwawancarai yang tahu bagaimana menggunakan suara mereka dengan benar menikmati keuntungan dibandingkan mereka yang tidak. Suaramu adalah kendaraan yang kamu gunakan untuk menyampaikan kalimatmu, dan kamu mengabaikannya dengan resiko sendiri.

Suara wawancara yang baik jelas dan menekankan poin-poin penting tanpa terlalu banyak keributan. Itu percaya diri dan terkendali, tetapi tidak pernah sombong. Itu naik ke kesempatan secara halus dan tak terlihat memudar ketika harus tetapi selalu menjaga perhatianmu. Menyenangkan untuk didengarkan.

Inilah yang tidak boleh dilakukan:

Jika kamu pernah diberi tahu, atau kamu curiga, kamu memiliki suara yang datar atau tidak bersemangat, berlatih adalah kunci untuk mengubahnya. Mintalah bantuan teman baik atau pelatih vokal.

Wawancara telepon

Semakin banyak perusahaan mulai menyadari bahwa, karena sebagian besar pekerjaan mereka dilakukan melalui telepon, masuk akal untuk mewawancarai kandidat menggunakan media ini. Jika kamu menginginkan pekerjaan di bagian penjualan Software, pertanyaan pelanggan, dll., Sebaiknya persiapkan dirimu untuk wawancara telepon.

Beberapa perusahaan cukup murah hati untuk memberi tahumu kapan tepatnya mereka akan menelepon dirimu, tetapi banyak yang tidak. Keluhan nomor satu yang saya dengar tentang wawancara telepon adalah bahwa panggilan tersebut selalu datang pada saat orang tidak siap untuk itu.

Satu menit mereka asyik dengan percakapan pribadi, selanjutnya mereka berbicara dengan pewawancara yang bersikeras mengajukan berbagai pertanyaan buruk kepada mereka. Mengingat bahwa kamu tidak dapat menunda hidupmu untuk satu panggilan telepon itu, sangat masuk akal untuk menyiapkan ringkasan jawabanmu dan meletakkannya di samping teleponmu sehingga ketika panggilan itu benar-benar datang, kamu akan mendapatkan poin-poin utama. Jawabanmu tepat di ujung jarimu, dan dapat membacanya jika perlu.

Strategi sederhana ini tidak dimaksudkan sebagai pengganti persiapan yang tepat, tetapi dapat membantumu untuk fokus dengan sangat cepat. Semuanya ada dalam apa yang kamu katakan dan bagaimana kamu mengatakannya. Dalam hal konten, jawaban untuk wawancara telepon tidak boleh berbeda dengan jawaban yang kamu berikan dalam wawancara biasa.

Fakta bahwa wawancara telepon tidak memberimu kesempatan untuk 'mengalihkan' pewawancara dengan senyummu yang mempesona dan bahasa komponen yang indah berarti bahwa ada lebih banyak penekanan pada apa yang kamu katakan. Gagasan bahwa kamu tidak perlu mempersiapkan sebanyak mungkin karena kamu tidak akan duduk berhadap-hadapan dengan pewawancara adalah ide yang berbahaya.

Perbedaan besar dengan wawancara telepon terletak pada suaranya. Meskipun suara penting dalam semua wawancara, secara alami suara dianggap jauh lebih penting dalam wawancara telepon. Faktanya, dapat dikatakan bahwa suara adalah bahasa komponen dalam wawancara telepon. Berikut beberapa hal yang harus dihindari saat diwawancarai melalui telepon:

- *Jeda panjang;*
- *Terlalu banyak ‘e’ dan ‘em’;*
- *Batuk atau bersin langsung;*
- *Suara latar belakang termasuk televisi, musik, teriakan anak-anak, dll.;*
- *Desahan panjang.*
- *Menegosiasikan gaji*

Seringkali wawancara berisi diskusi tentang ekspektasi gaji. Jika ditangani dengan benar, ini dapat membantu Anda memaksimalkan penghasilan. Inilah yang harus dilakukan

Berikan wawancara yang bagus

Penting untuk dipahami bahwa negosiasi gaji dimulai saat kamu masuk ke ruang wawancara, bukan ketika diskusi berubah menjadi uang. Dengan kata lain, salah satu hal terpenting yang perlu kamu lakukan untuk memaksimalkan apa yang bersedia dibayar oleh pemberi kerja kepadamu adalah untuk benar-benar menonjol selama wawancara berlangsung. Jelas, majikan jauh lebih cenderung memberikan lebih banyak uang mereka jika mereka pikir mereka akan mendapatkan nilai.

Lakukan riset Anda

Mencoba menegosiasikan gajimu tanpa melakukan penelitian dasar sama seperti mencoba mencapai target dengan mata tertutup. Penelitianmu harus fokus pada dua area. Pertama, cari tahu apa yang dibayar pasar untuk orang-orang sepertimu. Kamuharus mempertimbangkan semua kualifikasi, pengalaman, dan pencapaian utamamu. Yang penting, kamu juga perlu mempertimbangkan industri tempatmu akan bekerja karena beberapa industri membayar lebih banyak daripada yang lain untuk orang-orang dengan pengalaman dan kemampuan yang sebanding. Begitu pula dengan lokasi.

Perusahaan survei gaji, konsultan rekrutmen yang baik, dan organisasi profesional terkait biasanya dapat memberimu informasi gaji yang andal. Pastikan semua sumbermu kredibel dan kamu menggunakan lebih dari satu. Kasusmu akan segera runtuh (termasuk kredibilitasmu) jika sumbermu ditemukan menginginkan — dan akan terjadi jika kamu menghadapi negosiator berpengalaman yang mengetahui pasar.

Jangan pernah berdebat dan tidak pernah mengutip apa yang diklaim temanmu sebagai penghasilan mereka. Area penelitian keduamu harus fokus pada perusahaan itu sendiri. Kamu mungkin tidak bisa mendapatkan semua informasi yang kamu inginkan, tetapi ini seharusnya tidak menghentikanmu untuk mencoba (jangan sampai mengganggu dirimu sendiri). Hal-hal yang perlu diselidiki meliputi:

Kebijakan remunerasi.

Terkadang, terutama dengan perusahaan kecil, kebijakan semacam itu tidak ada. Namun, jika mereka memang ada dan kamu dapat mengaksesnya, kamu mungkin dapat menggunakan informasi ini untuk keuntunganmu. Misalnya, jika kamu tahu bahwa perusahaan meninjau kinerja dan gaji setiap enam bulan, kamu mungkin dapat menegosiasikan penangguhan gaji yang lebih tinggi sampai kamu memiliki enam bulan untuk membuktikan dirimu pada pekerjaan daripada menerima jumlah yang lebih rendah. untuk waktu yang tidak terbatas.

Tingkat gaji

Ini bisa jadi rumit karena informasi tentang gaji orang-orang sering kali diselimuti misteri. Tetapi jika kamu bisa mendapatkan wawasan, kamu setidaknya akan tahu apa yang kamu hadapi. Misalnya, mengetahui bahwa perusahaan cenderung membayar karyawannya di atas nilai pasar dapat menjadi informasi yang sangat berguna saat menegosiasikan gaji.

Seberapa baik perusahaan melakukan perjalanan.

Perusahaan yang berkinerja baik umumnya lebih cenderung membayar lebih daripada perusahaan yang berjuang secara finansial. Hal terakhir yang ingin kamu lakukan adalah menjual dirimu sendiri untuk perusahaan yang sedang naik daun.

Betapa putus asa mereka untuk mengisi posisi itu.

Beberapa pekerjaan lebih sulit diisi daripada yang lain, sementara pekerjaan lain sangat penting untuk keberhasilan perusahaan. Jika penelitianmu menunjukkan bahwa posisi yang kamu lamar kebetulan termasuk dalam salah satu kategori ini, maka masuk akal untuk mengasumsikan bahwa kamu memiliki pengaruh yang lebih besar dalam negosiasimu.

Hindari menyebut uang di muka

Prinsip penting dalam menegosiasikan gaji adalah meninggalkan diskusi sampai akhir. Idenya adalah untuk membuat kesan sebaik mungkin secara manusiawi sebelum pembicaraan tentang uang muncul. Ini tidak berbeda dengan penjual mana pun yang mencoba menjual produk.

Harga hanya disebutkan setelah semua fitur dan manfaat hebat dari produk yang sedang dibahas. Berbicara tentang harga sebelum menyoroti fitur dan manfaat tidak menghasilkan pendekatan penjualan yang baik, juga tidak menghasilkan negosiasi gaji yang baik.

Pertama, bicarakan tentang keterampilan dan pengetahuanmu dan bagaimana mereka dapat menguntungkan bisnis sebelum mengutip hargamu. Jika kamu kebetulan bertemu dengan pewawancara yang ingin membicarakan uang di muka, cobalah (dengan sopan) untuk meyakinkan mereka sebaliknya.

kamu bisa mencoba mengatakan sesuatu seperti: ‘Saya lebih suka meninggalkan diskusi tentang gaji sampai pembicaraan kita selesai. Saya benar-benar ingin Anda mendapatkan pemahaman yang lebih baik tentang apa yang saya tawarkan kepada perusahaan dan bagi saya untuk mempelajari lebih lanjut tentang pekerjaan tersebut sebelum uang dibahas. "

Jika itu tidak berhasil dan pewawancara bersikukuh, maka kamu tidak punya pilihan selain menghindari mengutip jumlah tertentu. Sebagai gantinya, kutip rentang (lihat di bawah). Melakukan hal itu akan memberimu ruang yang dapat digunakan untuk bermanuver di kemudian hari.

Prinsip pertama dalam mengutip sejumlah uang yang ingin kamu pertimbangkan adalah realisme. Mengutip jumlah yang terlalu tinggi yang tidak realistis kemungkinan besar akan merusak kredibilitasmu dan dapat merusak banyak pekerjaan baik yang kamu lakukan. Panduan berikut dirancang untuk membantumu menentukan rentang.

Tetapkan garis bawah Anda

Pikirkan baik-baik untuk menentukan apa keuntunganmu— yaitu, jumlah minimum absolut yang ingin kamu usahakan. Tiga faktor yang harus kamu pertimbangkan meliputi:

Biaya hidupmu, dengan mempertimbangkan kenaikan yang diharapkan;

- Apa yang akan ditanggung pasar berdasarkan tingkat pengalamanmu. Jangan pergi ke bawah titik bawah kisaran pasar. Jika kisaran pasar antara Rp. 15.000.000,- dan Rp. 25.000.000,- keuntunganmu tidak boleh di bawah Rp. 15.000.000,-. Sebaliknya, jika keadaan cukup menguntungkan, kamu dapat melampaui titik puncak;
- Betapa kamu menginginkan pekerjaan itu. Orang sering kali bersedia menerima kurang karena berbagai alasan pribadi yang penting seperti jam kerja yang lebih sesuai, waktu perjalanan yang sedikit, atau karena pekerjaan mewakili langkah pertama untuk perubahan karier.

Lakukan latihan rentang

Setelah kamu berhasil mencapai garis bawah, penting bagimu untuk mematuhinya. Menerima jumlah yang lebih kecil kemungkinan besar akan menyebabkan kekecewaan di kemudian hari. Jumlah minimummu akan mewakili titik terendah absolut dari kisaran gajimu.

Seberapa luas kamu ingin membuat kisaran harus bergantung pada semua faktor yang dibahas di atas, tetapi terutama pada apa yang dibayar pasar dan tingkat pengalamanmu.

Berikut satu kemungkinan pendekatan. Katakanlah kamu telah memutuskan bahwa jumlah minimum absolutmu adalah Rp. 20.000.000,-. kamu memiliki banyak pengalaman dan kamu tahu bahwa perusahaan sangat menyukaimu dan bahwa mereka telah mengalami kesulitan untuk mengisi posisi tersebut.

Kamu juga tahu bahwa ujung atas pasar dalam industrimu adalah Rp. 30.000.000,-. Dalam situasi yang menguntungkan seperti itu, bukanlah hal yang tidak masuk akal untuk mengutip kisaran gaji yang dimulai di atas minimummu dan melampaui ujung atas ujung atas pasar — katakanlah , Rp. 25.000.000,- hingga Rp. 35.000.000,-. Sebaliknya, jika kamu mengetahui bahwa terdapat persaingan yang ketat untuk pekerjaan tersebut dan pengalamanmu tidak luar biasa, maka Rp. 20.000.00,- hingga Rp. 28.000.000,- akan lebih masuk akal.

Pendekatan lain, yang kurang konservatif, untuk menetapkan kisaran gaji dalam skenario yang menguntungkan di atas adalah memiliki kisaran tetapi mengutip minimum yang lebih tinggi— katakanlah, Rp. 30.000.000,- hingga Rp. 35.000.000,-. Keuntungan dari pendekatan kedua ini adalah meningkatkan peluang untuk mendapatkan pemberi kerja secara otomatis membayar kuotasi minimummu dan sepenuhnya mengakui posisi tawarmu yang kuat.

Perusahaan yang tidak terlalu bersemangat (tetapi kamu tertarik untuk bekerja) mungkin takut dengan ekspektasimu, tetapi kamu harus dapat mengatasinya dengan setuju untuk membatalkan penawaran minimummu.

Tidak ada aturan yang tegas dan cepat tentang menetapkan jumlah minimum uang yang kamu bersedia untuk bekerja atau kisaran gaji. Panduan di atas hanyalah ilustrasi dari pendekatan yang memungkinkan. Hal yang paling penting adalah melakukan penelitianmu terlebih dahulu dan kemudian hindari mengutip jumlah yang tidak realistis kepada pemberi kerja.

Bagaimana menanggapi ketika Anda telah dipecat dari pekerjaan sebelumnya

Secara keseluruhan, majikan tidak suka memecat orang. Memecat seseorang sangat sulit dan sering kali menimbulkan kecemasan yang besar bagi kedua belah pihak. Sayangnya, bagaimanapun, ada karyawan yang tindakannya tidak memberi pilihan kepada majikan selain menerapkan sanksi pamungkas.

Namun, ada juga contoh di mana karyawan dipecat bukan karena kesalahan mereka sendiri. Pemecatan yang tidak adil ini dapat terjadi karena berbagai alasan, termasuk manajemen yang sangat tidak kompeten, desain pekerjaan yang sangat buruk (beberapa pekerjaan — terutama yang baru — belum dipikirkan dan sering kali membuat orang gagal), praktik perekrutan yang buruk, atau kurangnya pelatihan .

Masalahnya di sini adalah bagaimana seseorang yang telah diberhentikan secara tidak adil menanggapi rentetan pertanyaan pada wawancara berikutnya. Secara khusus, bagaimana mereka menanggapi pertanyaan yang ada di mana-mana, 'Mengapa kamu meninggalkan majikanmu sebelumnya?' Ketika kita dapat berasumsi secara masuk akal bahwa memberi tahu pewawancara bahwa kamu dipecat (meskipun tidak adil) dapat berbatasan dengan wawancara bunuh diri? Seperti yang telah disebutkan, pewawancara cenderung menjadi kelompok yang berhati-hati (umumnya dengan alasan yang bagus) dan hanya mengucapkan kata-kata kamu ketika kamu mencoba menjelaskan betapa sulitnya kamu.

Sayangnya, beberapa perekrut (terutama di pasar tenaga kerja yang kelebihan pasokan) akan menunjukkan keengganan yang cukup besar untuk mempekerjakan seseorang yang dipecat dari pekerjaan terakhir mereka, bahkan jika orang itu tidak bersalah. Sebagian besar keengganan mereka berasal dari ketakutan bahwa orang yang sebelumnya dipecat tidak akan berhasil di pekerjaan baru. Dalam skenario seperti itu, perekrut mungkin akan terlihat tidak kompeten.

Kenyataan yang dingin dan pahit adalah bahwa orang-orang yang dipecat dari pekerjaan terakhir mereka umumnya memulai perlombaan wawancara agak jauh di belakang bidang lainnya. Namun, semuanya bukanlah malapetaka dan kesuraman — itu hanya berarti mereka harus berusaha lebih keras. Ada beberapa hal yang dapat dilakukan orang yang diwawancarai untuk meningkatkan peluang sukses mereka.

Jelaskan apa yang terjadi secara detail

Salah satu opsinya adalah membuat gambaran yang sangat jelas tentang keadaan yang menyebabkan pemecatanmu. Salah satu kuncinya di sini adalah tidak menggunakan istilah yang merendahkan. Hindari berbicara dengan bahasa yang kasar atau menghina mantan majikanmu, meskipun mungkin sulit. Tetap berpegang pada fakta dan sampaikan kasusmu tanpa perasaan, menggunakan bahasa yang terukur. Empat hal yang dapat kamusertakan untuk mendukung casingmu adalah:

- *Pengalaman serupa dengan karyawan lain. Ini adalah argumen yang kuat. Jika orang lain diperlakukan dengan cara yang mirip dengan Anda, maka itu adalah bukti kuat yang mengutuk majikan.*
- *Janji yang tidak ditepati. Pengusaha yang memecat karyawan secara tidak adil biasanya membuat banyak janji yang mereka ingkari.*
- *Contoh praktik manajemen yang buruk. Ini dapat mencakup beberapa hal, termasuk: tidak ada pelatihan yang memerlukan pelatihan; perubahan signifikan tanpa peringatan apapun; nol konsultasi atau umpan balik; perilaku kasar; atau perubahan besar pada tugas pekerjaan Anda tanpa peringatan atau konsultasi apa pun.*
- *Apa yang Anda lakukan untuk menyelamatkan situasi. Ini termasuk upaya yang Anda lakukan untuk memperbaiki masalah, termasuk saran yang Anda buat atau tindakan apa pun yang Anda lakukan. Berikut adalah jawaban yang bagus untuk pertanyaan 'Mengapa Anda meninggalkan majikan Anda sebelumnya?' Mungkin terdengar seperti:*
- *Sayangnya kami berpisah karena serangkaian insiden negatif. Mantan majikan saya berada di bawah tekanan dan mengalami kesulitan besar dalam mengatasinya. Dia sering melampiaskan rasa frustrasinya pada stafnya, termasuk menggunakan bahasa kasar dan membuat segala macam ancaman.*
- *Akibatnya, banyak stafnya yang ketakutan dan aktif mencari pekerjaan lain. Faktanya, pergantian staf sangat tinggi. Dia juga memiliki kebiasaan membuat komitmen penting tetapi sangat jarang menaatinya.*

Salah satu contohnya adalah janji yang dia buat bahwa kami akan menerima pelatihan tentang mesin baru. Pelatihan ini akan meningkatkan tingkat produktivitas kami secara signifikan dan membuat kehidupan semua orang jauh lebih mudah, namun pelatihan tidak pernah tiba. Ketika saya mendekatinya tentang masalah ini, dia mengatakan kepada saya untuk mengurus urusan saya sendiri. Ketika saya mencoba menjelaskan kepadanya bahwa perhatian saya adalah untuk kesejahteraan bisnis, dia menjadi sangat marah dan langsung memecat saya.

Bandingkan jawaban di atas dengan yang berikut ini:

Saya pergi karena saya dipecat, yang merupakan hal terbaik yang dapat terjadi pada saya. Mantan bos saya sangat buruk.

Selain tidak tahu bagaimana menjalankan bisnis, dia juga tidak memiliki keterampilan orang sama sekali. Dia adalah seorang pengganggu dan idiot dan tidak bisa mengatasi tekanan. Tidak ada yang bisa menahannya dan mereka yang tidak melompat kapal sedang mencari pekerjaan lain.

Saya dipecat karena saya mengatakan kepadanya bahwa kami membutuhkan pelatihan tentang pelatihan mesin baru yang dia janjikan akan kami terima dan yang akan meningkatkan tingkat produktivitas kami secara signifikan.

Terakhir saya dengar dia bangkrut, yang tidak mengejutkan saya sama sekali. Meskipun kedua jawaban di atas pada dasarnya mengatakan hal yang sama, pada satu tingkat keduanya sangat berlawanan.

Jawaban pertama tidak memihak, menghindari penggunaan bahasa yang kasar dan membuat kasus yang menarik sebelum mengajukan pemecatan. Pada saat pembicara pertama sampai pemecatan, ada kemungkinan besar dia telah mendapatkan simpati dari pewawancara.

Sedangkan jawaban kedua, selain kasar dan emosional (yang akan membuat khawatir pewawancara mana pun), dimulai dengan berbahaya karena menyebutkan pemecatan dalam kalimat pembukaannya. Menyebutkan pemecatan dalam kalimat pertama Anda tidak memberi Anda kesempatan untuk melunakkan pewawancara.

Hindari menyebut pemecatan

Pilihan kedua melibatkan menjaga mulutmu tetap berteriak. Mengingat stigma yang melekat pada karyawan yang dipecat, tidak masuk akal untuk menyebutkan pemecatan dan menakut-nakuti pewawancara, terutama di mana masa kerjamu untuk waktu yang singkat atau dilakukan di masa lalu.

Beresiko menyinggung orang-orang yang senang menempati moral yang tinggi, menurut saya ada kalanya hal-hal tertentu tidak perlu diungkapkan kepada pewawancara. Pada akhirnya, semua pemberi kerja berhak untuk mengetahui hanya apakah dirimu dapat melakukan pekerjaan itu, apakah kamu akan cocok dengan budaya organisasi mereka dan seperti apa tingkat motivasimu.

Wawancara kelompok

Bentuk wawancara yang semakin populer adalah wawancara kelompok, di mana kumpulan orang yang diwawancarai berkumpul dan diberi serangkaian tugas untuk diselesaikan sebagai kelompok (meskipun beberapa tugas mungkin mengharuskanmu bertindak sendiri, seperti memberikan presentasi) .

Contoh tugas kelompok dapat mencakup latihan apa pun yang membutuhkan pemecahan masalah, menghasilkan solusi kreatif, perencanaan dan pengorganisasian, menentukan dan menetapkan tujuan atau menyelesaikan konflik. Sementara kelompok mengerjakan tugas-tugas ini, situasinya dipantau dengan cermat oleh

seorang penilai, atau sekelompok penilai, yang tugasnya mengamati bagaimana kamu berinteraksi dengan kelompok dan apa kontribusimu.

Berdasarkan perilakumu yang dapat diamati — yaitu, apa yang kamu katakan, bagaimana kamu mengatakannya, apa yang kamu lakukan dan bagaimana kamu berinteraksi dengan orang lain dalam kelompokmu— penilai akan menarik kesimpulan tentang kesesuaian dirimu. Di satu sisi, wawancara kelompok adalah wawancara perilaku yang paling akhir.

Kunci dari wawancara kelompok adalah memastikan bahwa kamu menunjukkan perilaku yang diperlukan dan menghindari perilaku yang tidak diinginkan. Perilaku yang diinginkan dan tidak diinginkan pada wawancara kelompok Pastikan kamu berkontribusi.

Kontribusimu harus dirancang untuk memfasilitasi kelancaran fungsi kelompok dan penyelesaian tugas. Hindari perilaku apa pun yang dapat merusak dua tujuan utama ini. Perilaku merendahkan dapat mencakup apa pun yang dapat dianggap sebagai perilaku agresif atau terlalu mendominasi, seperti:

- *Mengintimidasi orang lain;*
- *Bersikeras dengan cara Anda sendiri;*
- *Tidak mendengarkan atau mengabaikan kontribusi orang lain;*
- *Memonopoli pusat perhatian.*

Sama buruknya dengan perilaku yang terlalu pasif. Duduk di sana dan tidak menyumbang, atau menyumbang sangat sedikit, tidak akan membantumu.

Penting bagimu untuk memiliki kepercayaan diri untuk memberikan kontribusi. Jangan duduk di sana sambil berpikir, 'Ya Tuhan — bagaimana jika mereka semua menertawakan saran saya?' Jauh lebih baik memberikan sumbangan yang tidak terlalu spektakuler daripada duduk diam di sana.

Dengarkan dan akui apa yang orang lain katakan. Jika seseorang memberikan saran yang bagus, mengakuinya akan membuatmu mendapatkan poin brownies. Tapi hindari mengakuinya demi melakukannya. Dan, apa pun yang kamu lakukan, jangan menghargai setiap saran.

Jika memungkinkan, bantu orang lain tetapi lakukan dengan benar. Hindari mempermalukan anggota kelompok atau mengambil alih tugas mereka. Jangan melupakan tujuan tugas. Jika kamu melihat kelompok menyimpang dari tugas, cobalah untuk membawa mereka kembali ke jalur dengan mengingatkan mereka tentang tujuannya.

Cobalah untuk mencari tahu perilaku apa yang telah dirancang untuk ditimbulkan oleh tugas tersebut. Misalnya, jika menurutmu tugas telah dirancang untuk menggambarkan perilaku yang berkaitan dengan pemecahan masalah dalam kelompok, tugasmu adalah mendemonstrasikan perilaku tersebut. Ini mungkin termasuk:

- *Membuat semua orang setuju tentang masalah yang sebenarnya (definisi masalah);*
- *Memulai diskusi tentang kemungkinan penyebab masalah;*
- *Menyelesaikan penyebab yang paling mungkin;*
- *Menyarankan sesi curah pendapat tentang solusi yang mungkin;*
- *Mendapatkan kesepakatan tentang solusi terbaik;*
- *Menyusun rencana tindakan yang dirancang untuk mengimplementasikan solusi;*
- *Mengingat untuk menghindari prosedur mendominasi.*

Mudah-mudahan saya meyakinkan kamu tentang pentingnya membangun hubungan dan kepercayaan dan bahwa memenangkan pekerjaan bergantung pada lebih dari sekadar menjawab pertanyaan dengan benar. Meskipun kita semua berbeda dan membawa gaya komunikasi yang berbeda dalam wawancara, para ahli setuju bahwa beberapa perilaku lebih efektif daripada yang lain dalam hal membangun hubungan dan kepercayaan. Penting untuk membiasakan diri dengan perilaku ini sehingga kamu dapat memaksimalkan keefektifanmu.

Kamu mungkin menemukan beberapa teknik yang dijelaskan di atas agak sulit untuk dikuasai pada awalnya. Itu bukan karena mereka pada dasarnya sulit — pada kenyataannya, kebanyakan dari mereka terus terang. Tantangannya adalah untuk tidak mempelajari perilaku saat ini, tetapi dengan sedikit ketekunanmu akan kagum pada seberapa cepatmu dapat mulai berubah; usaha yang benar-benar berharga untuk terus melakukannya sampai kamu menguasai semua teknik.

Aktivitas yang Disarankan

Untuk membantumu menguasai teknik-teknik ini, berikut beberapa aktivitas yang disarankan untuk membantumu selama ini.

Seperti disebutkan di atas, mulailah mencontohkan perilaku orang-orang yang keterampilan interpersonalnya kamu kagumi.

Kamu dapat mempraktikkan banyak teknik dalam kebanyakan situasi sosial. Lain kali kamu bercakap-cakap dengan seseorang, pikirkan bahasa komponenmu.

Apakah itu cocok untuk meningkatkan komunikasi, hubungan dan kepercayaan? Dan apa yang dapat kamu lakukan untuk memperbaikinya? Setelah beberapa saat, kamu akan menemukan bahwa kesadaran diri seperti ini menjadi kebiasaan.

Ringkasan poin-poin penting

Membangun hubungan dan kepercayaan membutuhkan tiga hal: menjawab pertanyaan dengan cerdas dan jujur; memastikan semua komunikasi non-verbal Anda (bahasa komponen dan penampilan pribadi) tidak menimbulkan kekhawatiran pada pewawancara; dan menyesuaikan diri dengan perilaku wawancara yang dapat diterima, seperti tidak pernah berdebat.

Waspadai kesan pertama dan terakhir — orang cenderung lebih mengingat apa yang terjadi di awal dan akhir interaksi apa pun, termasuk wawancara. Tersenyum, menggunakan ekspresi wajah yang sesuai dan menganggukkan kepala pada saat yang tepat semuanya memberikan kesan yang positif.

Untuk wawancara telepon, rekrut suara Anda; itu menggantikan bahasa komponen Anda saat berbicara di telepon.

Ingat kunci yang harus dan tidak boleh dilakukan: berikan penghargaan jika sudah waktunya dan hindari mengkritik orang lain, termasuk atasan sebelumnya; gunakan pernyataan positif tetapi hindari membohongi diri sendiri; sebutkan kekurangan atau rintangan yang telah Anda atasi tetapi hindari hiasan.

Saat menegosiasikan gaji Anda, lakukan riset terlebih dahulu — jangan meremehkan diri Anda sendiri, tetapi realistislah dengan apa yang Anda minta. Hindari membahas uang sebelum Anda menyoroti apa yang dapat Anda bawa ke perusahaan. Dalam wawancara panel, pastikan Anda membiasakan diri dengan nama semua orang.

Dalam wawancara kelompok, bersikaplah proaktif dalam menunjukkan perilaku yang dicari pewawancara.

BAB 8
JAWABAN EFEKTIF UNTUK PERTANYAAN UMUM

Sekarang, selain mengenali bahan dasar tanggapan wawancara yang baik, kamu juga harus bisa mengumpulkan jawaban efektifmu sendiri. Kamu harus tahu cara: cari tahu sebanyak mungkin tentang pekerjaan itu sebelum menyelesaikan jawabanmu;

- *Gunakan empat langkah untuk menyatukan bagian-bagian utama dari jawaban Anda, termasuk apa yang Anda lakukan, bagaimana Anda melakukannya, konteks di mana Anda melakukannya, dan hasil;*
- *Gabungkan semua informasi Anda sehingga Anda dapat mengartikulasikan jawaban yang jelas dan koheren yang tidak berliku-liku di semua tempat;*
- *Menjawab berbagai pertanyaan, termasuk yang berkaitan dengan tugas yang telah Anda lakukan sebelumnya, tugas yang belum Anda lakukan tetapi keahliannya telah Anda kuasai dan tugas yang belum Anda lakukan dan belum Anda miliki;*
- *Gunakan bahasa komponen Anda dan keterampilan komunikasi interpersonal lainnya untuk membangun dan memelihara hubungan.*
- *Tidak ada rumus sederhana untuk jawaban yang bagus*

Penting untuk menegaskan kembali pada saat ini bahwa, meskipun ada pedoman yang berguna tentang cara menjawab pertanyaan, tidak ada cetak biru atau struktur tunggal untuk jawaban yang berlaku untuk semua pertanyaan wawancara.

Kadang-kadang mungkin tepat untuk memberikan jawaban tiga bagian yang mencakup konteks, apa yang kamu lakukan dan bagaimana kamu melakukannya, dan hasil. Di lain waktu, mungkin lebih tepat untuk membicarakan tentang kemampuanmu untuk melakukan pekerjaan, kecocokan budaya, dan tingkat motivasimu.

Seringkali, mungkin lebih tepat untuk mencampur dan mencocokkan dari yang di atas. Pada akhirnya, terserah kamu untuk mengenali struktur atau pendekatan yang sesuai untuk setiap pertanyaan. Dan satu pendekatan mungkin sama baiknya dengan yang diingat berikutnya, tidak ada jawaban yang sempurna. Latihan akan memberimu kemampuan untuk memberikan respons terbaik.

Bab ini menyajikan beberapa jawaban yang baik dan tidak terlalu bagus untuk pertanyaan wawancara umum, serta penjelasan singkat mengapa mereka berhasil. Dengan belajar mengenali jawaban yang kurang efektif, kamu seharusnya berada dalam posisi yang lebih baik untuk menghindarinya.

Pertanyaan: Mengapa Anda memilih pekerjaan ini?

Jawaban yang bagus

Sejak saya ingat, saya tertarik dengan bidang pekerjaan ini. Yang membuat saya tertarik adalah kesempatan yang diberikannya untuk berinteraksi dengan orang lain, memecahkan masalah, dan bekerja secara mandiri.

Saya menyukai jika suatu hari saya dapat berada di jalan membantu klien mengatasi masalah mereka sedangkan hari berikutnya saya dapat berada di kantor saya bekerja dengan tim orang yang mencoba memecahkan masalah teknis yang kompleks. Saya sangat menikmati bekerja di industri jasa seperti kami di mana saya dapat memuaskan klien.

Jawaban yang tidak terlalu bagus

Sebenarnya saya tersandung ke dalamnya secara tidak sengaja. Saya selalu ingin menjadi seorang aktor, tetapi mendapatkan pekerjaan hampir mustahil. Saya kira alasan saya masih di bidang pekerjaan ini adalah karena saya telah mengambil semua keterampilan dan pengetahuan dan mengetahui cara mengatasi jebakan. Saya telah melakukannya untuk beberapa waktu sekarang dan saya kira Anda bisa mengatakan bahwa saya sudah tua dan tahu cara mengirim barang.

Komentar

Jawaban pertama menanggapi pertanyaan dengan segera dan kemudian melanjutkan dengan menyoroti tugas utama pekerjaan — berinteraksi dengan orang-orang, pemecahan masalah, dll. —Sebagai alasan mengapa kandidat memilih pekerjaan tersebut. Sama pentingnya, kami sangat memahami tingkat motivasi tinggi kandidat dan keinginan untuk memberikan layanan yang baik. Ini juga menyiratkan bahwa kandidat menikmati bekerja dalam tim dan dapat melakukan pekerjaan itu, sehingga menangani tiga hal yang ingin didengar oleh pemberi kerja.

Dalam jawaban kedua kita harus menunggu sampai kalimat ketiga sebelum pertanyaan itu dibahas — sangat terlambat. Terlepas dari pengalaman kandidat, kami merasakan ketidakpedulian yang kuat terhadap pekerjaan itu.

Kami mendapat kesan bahwa ini hanya sebuah pekerjaan, sedangkan jawaban pertama dipenuhi dengan antusiasme.

Pertanyaan:Menurut Anda, faktor apa yang menentukan kemajuan seseorang dalam suatu organisasi?

Jawaban yang bagus

Menurut saya, ada tiga hal yang menentukan kemajuan seseorang dalam sebuah organisasi. Ini adalah, pertama, kemampuan untuk melakukan pekerjaan dengan baik, termasuk kemauan untuk mempelajari hal-hal baru dan beradaptasi dengan keadaan yang berubah; kedua, untuk dapat menyesuaikan diri dengan budaya

organisasi (yaitu dapat bergaul dengan rekan kerja); dan ketiga, memiliki dorongan dan motivasi tingkat tinggi.

Tentunya ini adalah tiga hal yang saya tekankan untuk diri saya sendiri di tempat kerja. Jika suatu saat saya merasa saya tidak dalam kondisi terbaik saya di ketiga area tersebut, saya berhenti dan bertanya pada diri sendiri apa yang dapat saya lakukan untuk memperbaiki keadaan. Saya tidak berpikir ada orang yang benar-benar bahagia dalam pekerjaan mereka jika ketiga area tersebut tidak terpenuhi. Sejauh ini mereka telah mendukung saya dengan baik.

Jawaban yang tidak terlalu bagus

Mempertahankan sisi baik bos mungkin adalah hal nomor satu yang dapat saya pikirkan. Tidak peduli seberapa baik Anda jika Anda tidak cocok dengan atasan Anda, saya pikir hari-hari Anda sudah dihitung. Tentu saja, ini juga membantu untuk menjadi baik dalam pekerjaan Anda, tetapi bisa memainkan permainan itu, menavigasi melalui ranjau politik organisasi menurut saya lebih penting. Saya menyadari ini mungkin terdengar agak sinis, tetapi kita semua tahu bahwa untuk mencapai manajemen senior, seseorang perlu mengetahui cara memainkan permainan.

Komentar

Pertanyaan seperti ini harus segera dikenali sebagai peluang untuk menyoroti kekuatan Anda. Jawaban pertama berbicara langsung tentang tiga hal yang ingin didengar semua pemberi kerja — kemampuan untuk melakukan pekerjaan, kesesuaian budaya, dan motivasi, kemudian melangkah lebih jauh dan menyatakan bahwa ketiganya adalah kualitas yang ditawarkan kandidat. Jawaban kedua terlalu sinis dan gagal menekankan kekuatan kandidat. Ada sedikit keraguan bahwa kemampuan untuk 'memainkan permainan' dapat mempengaruhi kemajuan seseorang, tetapi membuang semua telur Anda ke dalam keranjang itu adalah kesalahan yang fatal.

Pertanyaan: Mengapa Anda ingin bekerja untuk organisasi kami?

Jawaban yang bagus

Perusahaan Anda adalah jenis perusahaan tempat saya dapat memaksimalkan kontribusi saya. Semua penelitian saya telah mengungkapkan bahwa Anda tidak hanya pemimpin pasar dalam standar layanan dan inovasi produk, tetapi Anda juga memiliki budaya kerja yang hebat.

Setiap orang yang saya ajak bicara telah membicarakan tentang tingkat dukungan, pelatihan, dan pengakuan yang tinggi yang diterima karyawan. Anda menawarkan prospek karier yang hebat, pekerjaan yang menarik, dan kebijakan yang ramah keluarga. Di atas segalanya, saya selalu ingin bekerja untuk perusahaan yang menawarkan pekerjaan yang menantang dan mutakhir.

Jawaban yang tidak terlalu bagus

Saya tahu organisasi Anda benar-benar menjaga orang-orangnya — semua orang yang saya ajak bicara ingin bekerja di sini. Anda membayar dengan baik dan menjaga karyawan Anda. Anda adalah perusahaan besar, yang berarti prospek peningkatan karier saya akan meningkat dan mudah-mudahan saya tidak akan melakukan pekerjaan yang sama sepanjang waktu. Saya suka gagasan dirotasi dan mempelajari hal-hal baru.

Komentar

Nada dari jawaban pertama diatur dalam kalimat pembuka, di mana kandidat berbicara tentang keinginan untuk berkontribusi yang merupakan hal yang menggairahkan pemberi kerja. Jawabannya mengakui semua hal baik tentang perusahaan, tetapi yang paling penting menghubungkan nilai tambah ini dengan kontribusi dari pihak kandidat.

Dengan kata lain, ini bukan hanya tentang apa yang bisa didapat kandidat dari perusahaan tetapi juga apa yang ingin dikembalikan oleh kandidat. Masalah utama dengan jawaban kedua adalah bahwa yang terpenting adalah apa yang dapat diperoleh kandidat dari perusahaan. Tidak ada hubungan yang jelas dibuat antara apa yang ditawarkan perusahaan dan bagaimana faktor-faktor ini dapat meningkatkan kontribusi kandidat.

Pertanyaan: Apa yang ingin Anda lakukan dalam karir Anda lima tahun dari sekarang?

Jawaban yang bagus

Saya ingin melakukan apa yang saya lakukan sekarang — yaitu, menikmati pekerjaan saya, bekerja keras, dan berkontribusi dengan kemampuan terbaik saya. Tentu saja, saya berharap bahwa dalam waktu lima tahun ke depan, pengalaman tambahan saya akan memberi saya manfaat yang baik untuk tanggung jawab yang lebih besar, yang merupakan sesuatu yang sangat saya harapkan untuk diambil ketika saatnya tiba. Namun, yang paling penting adalah menjadi bahagia, produktif, dan menjadi anggota tim yang berharga.

Jawaban yang tidak terlalu bagus

Pada dasarnya, saya ambisius dan pekerja keras, jadi saya berharap dapat memajukan karier saya secara signifikan. Tujuan saya adalah bekerja keras dan sejauh yang saya bisa. Saya pikir saya akan melihat semacam posisi manajemen dengan tanggung jawab yang lebih besar dan tentu saja penghargaan yang lebih besar.

Komentar

Tidak ada yang salah dengan jawaban kedua; pada kenyataannya, itu membuat beberapa poin bagus — yaitu, langsung ke intinya dan mempromosikan kerja keras dan ambisi kandidat untuk maju. Alasan mengapa tidak sebaik jawaban pertama terletak pada pendekatannya yang terbatas: tujuan utama kandidat adalah hanya untuk promosi.

Sub-teksnya adalah jika tidak ada peluang untuk promosi, kandidat tersebut dapat keluar. Di sisi lain, jawaban pertama mengakui pentingnya kerja keras dan promosi tetapi dengan sangat bijak melanjutkan dengan mengatakan bahwa dipromosikan bukanlah satu-satunya hal yang penting. Jawaban pertama kurang egosentris dan lebih sadar akan pentingnya memberikan kontribusi kepada perusahaan.

Pertanyaan: Jelaskan pekerjaan ideal Anda.

Jawaban yang bagus

Pekerjaan yang saya lamar ini mengandung banyak, jika tidak semua, unsur-unsur pekerjaan ideal saya. Ini berisi banyak variasi, menantang secara intelektual, akan memungkinkan saya untuk bekerja sendiri maupun dalam lingkungan tim (yang terbaik dari kedua dunia), Dan juga akan memungkinkan saya untuk menawarkan solusi kreatif kepada klien. Saya selalu berkembang dalam lingkungan yang menantang dan didorong oleh hasil dan pekerjaan ini menawarkan semua itu kepada saya.

Jawaban yang tidak terlalu bagus

Pekerjaan ideal saya adalah di mana saya akan bekerja keras tetapi saya tidak akan terlalu stres sepanjang waktu.

Ini akan memiliki banyak variasi dan sejumlah tantangan yang bagus dengan banyak peluang untuk kemajuan. Ini akan mencakup orang-orang hebat untuk diajak bekerja sama serta bos yang baik.

Komentar

Salah satu alasan mengapa jawaban pertama sangat efektif adalah karena jawaban tersebut mengaitkan pekerjaan ideal kandidat dengan pekerjaan sebenarnya yang dipermasalahkan. Memberi tahu pewawancara bahwa pekerjaan yang Anda lamar adalah pekerjaan yang Anda anggap ideal sangat masuk akal. Perhatikan bahwa semua bahan utama dari variasi pekerjaan, tantangan, bekerja sendiri maupun dalam lingkungan tim, dan memberikan solusi kreatif kepada klien.

Sekali lagi, jawaban kedua bukanlah jawaban yang cacat fatal. Kesalahan utamanya adalah menyebutkan stres. Begitu Anda menyebutkan stres, bel alarm pewawancara akan mulai berdering. Mereka ingin tahu seberapa banyak stres itu terlalu banyak dan hal-hal apa yang membuat Anda stres, bukan apa yang ingin Anda bicarakan dalam sebuah wawancara.

Pertanyaan: Apa yang memotivasi Anda?

Jawaban yang bagus

Ada banyak hal yang memotivasi saya di tempat kerja, tetapi tiga dari motivator terbesar saya pasti adalah pemecahan masalah terutama masalah yang sangat teknis yang membutuhkan pengetahuan khusus; mempelajari hal-hal baru dan mengikuti semua perubahan di bidang saya; dan bekerja dalam lingkungan tim kooperatif tempat kami saling melempar ide dan menghasilkan solusi kreatif. Saya suka persahabatan yang menyertai itu.

Jawaban yang tidak terlalu bagus

Mungkin motivator terbesar saya adalah memiliki pekerjaan yang menyenangkan, yang sangat saya nantikan dan kuasai. Tidak ada yang lebih buruk daripada beralih ke pekerjaan yang tidak Anda nikmati hari demi hari. Juga, saya suka memiliki jam kerja yang menyenangkan. Saya tidak keberatan untuk kembali sesekali dan membantu, tetapi saya tidak ingin melakukannya sepanjang waktu. Saya juga suka bekerja di kota karena mudah untuk mencapai dari tempat saya tinggal dan memberi saya akses mudah ke toko-toko dan restoran yang bagus.

Komentar

Jawaban pertama hanya akan menjadi efektif jika tugas-tugas yang disebutkan di dalamnya menyelesaikan masalah yang sangat teknis, mengikuti inovasi terbaru dan menikmati bekerja secara kreatif dalam tim semuanya merupakan bagian dari deskripsi pekerjaan. . . intinya adalah bahwa strategi yang sangat baik untuk menjawab pertanyaan motivasi adalah pergi ke tugas utama pekerjaan dan membicarakannya. Jawaban kedua dimulai dengan baik tetapi tidak menyebutkan apa yang merupakan pekerjaan yang menyenangkan. Setelah itu, itu adalah jawaban yang sangat salah. Jam kerja dan lokasi kerja mungkin menjadi faktor motivasi, tetapi tidak boleh disebutkan karena gagal menunjukkan bagaimana Anda akan memberi nilai tambah pada pekerjaan.

\

Pertanyaan: Kualitas apa yang menurut Anda penting untuk sukses di bidang ini?

Jawaban yang bagus

Kualitas yang diperlukan untuk sukses di bidang ini akan mencakup keterampilan dan pengetahuan untuk benar-benar melakukan pekerjaan dengan benar.

Saya tidak hanya berbicara tentang semua keterampilan teknis, seperti mengetahui cara mengoperasikan berbagai program perangkat lunak dan pengetahuan komprehensif tentang undang-undang yang relevan dan cara menerapkan undang-undang itu,

Tetapi juga kemampuan untuk bergaul dengan orang lain, memiliki keterampilan komunikasi yang hebat dan tahu bagaimana merencanakan dan mengatur pekerjaan Anda sambil bekerja di bawah tekanan yang cukup besar.

Saya juga berpikir motivasi dan dorongan tingkat tinggi sangat penting. Ini semua adalah kualitas yang saya miliki dan dapat dibawa ke posisi ini sejak hari pertama.

Jawaban yang tidak terlalu bagus

Kualitas yang diperlukan untuk sukses dalam bidang ini akan mencakup pemahaman rinci tentang semua program perangkat lunak yang diperlukan untuk menyelesaikan operasi. Tidak hanya seseorang membutuhkan pengetahuan tentang bagaimana mengoperasikan perangkat lunak tetapi juga bagaimana memperbaiki sesuatu ketika terjadi kesalahan dan selalu ada sesuatu yang tidak beres.

Hal yang sama dapat dikatakan untuk teknis hukum yang kompleks. Seperti yang Anda ketahui, dalam industri kami hal buruk ada pada detail dan pemahaman yang dangkal tentang undang-undang dapat menyebabkan banyak masalah. Selain memiliki pemahaman menyeluruh tentang semua persyaratan pemrograman untuk pekerjaan ini, saya juga memiliki pengetahuan komprehensif tentang seluk-beluk hukum.

Komentar

Jenis pertanyaan ini mengundang Anda untuk langsung ke tugas utama dari pekerjaan yang Anda lamar dan menggunakannya sebagai jawaban Anda (ini adalah strategi yang sama yang digunakan dalam menjawab pertanyaan motivasi).

Jawaban pertama tidak hanya itu. Ini lebih unggul dari respons kedua karena mencakup lebih banyak pangkalan. Selain berbicara tentang keterampilan teknis, juga berbicara tentang bergaul dengan orang, perencanaan dan pengorganisasian, dan komunikasi yang baik (kompetensi umum).

Jawaban kedua bukanlah jawaban yang buruk, tetapi itu jatuh ke dalam perangkap umum yang hanya berfokus pada sisi teknis pekerjaan.

Pertanyaan: Ceritakan tentang saat Anda menangani situasi sulit dengan rekan kerja.

Jawaban yang bagus

Tahun lalu, salah satu kolega kami menunjukkan banyak perilaku agresif, termasuk mendominasi rapat tim, meremehkan ide orang lain, dan tidak bekerja sama.

Saya mendekati rekan-rekan saya yang lain tentang dia dan segera menyadari bahwa semua orang merasakan hal yang sama seperti saya. Kami memutuskan untuk tidak membawa masalah ini kepada manajer kami sampai kami memiliki kesempatan untuk berbicara dengannya terlebih dahulu. Jadi kami memutuskan bahwa pada pertemuan berikutnya kami akan mengangkat masalah ini dengannya. Saya terpilih untuk memulai diskusi.

Pada pertemuan tersebut saya menghindari mempersonalisasi masalah dan saya menghindari menggunakan bahasa yang menghasut. Saya juga mengadopsi nada ceria dan optimis. Hasilnya lebih baik dari yang kami perkirakan. Dia berterima kasih kepada saya atas sikap halus saya dalam mengangkat masalah dan juga berterima kasih kepada kami semua karena telah berbicara dengannya terlebih dahulu sebelum membahasnya lebih jauh. Setelah pertemuan kami, perilakunya berubah menjadi lebih baik.

Jawaban yang tidak terlalu bagus

Ada suatu waktu ketika salah satu rekan saya tidak berusaha keras, dia juga tidak mau bekerja sama dengan anggota tim kami yang lain. Manajer gagal mengambilnya karena beberapa anggota tim menutupi kesalahannya dan dia akan selalu berusaha untuk bersikap sangat ramah ketika manajer ada. Jadi suatu hari ketika dia tidak kooperatif, saya menariknya ke samping dan memberi tahu dia apa yang saya pikirkan tentang dia. Sejak hari itu perilakunya terhadap saya berubah. Dia berusaha keras untuk bersikap ramah terhadap saya dan memastikan semua pekerjaan yang saya butuhkan dilakukan dengan benar. Sayangnya, perilakunya terhadap anggota tim kami yang lain tidak berubah sama sekali. Pelajaran yang saya pelajari adalah Anda harus membela diri sendiri karena tidak ada orang lain yang mau.

Komentar

Jawaban pertama menunjukkan kemampuan untuk berkonsultasi dengan kolega, kemampuan untuk menyelesaikan masalah sendiri daripada segera melaporkannya ke manajemen, dan kemampuan untuk mengomunikasikan informasi yang sangat sensitif dengan cara yang tepat. Ini juga menunjukkan hasil yang bagus untuk semua orang yang terlibat. Jawaban kedua terlalu sempit dalam fokusnya. Ini memecahkan masalah hanya untuk individu itu tetapi gagal untuk mengatasi masalah harmoni dan kerjasama tim yang lebih luas.

Pertanyaan: Ceritakan tentang saat Anda harus memenuhi tenggat waktu yang sangat ketat.

Jawaban yang bagus

Ketika saya bekerja untuk Komisi Antarplanet, saya diminta untuk memenuhi beberapa tenggat waktu yang ketat. Saya dapat secara konsisten memenuhi semua tenggat waktu saya dengan mengikuti prinsip-prinsip perencanaan dan pengorganisasian yang baik.

Ini termasuk merencanakan pekerjaan saya dengan baik ke depan sehingga tidak ada kejutan, memastikan bahwa semua orang di tim saya terlatih dengan baik dan sangat menyadari tanggung jawab mereka, selalu memiliki berbagai rencana darurat ketika ada yang salah, dan tidak pernah menerima lebih banyak pekerjaan daripada yang dapat kami tangani.

Efektivitas praktik ini disorot oleh fakta bahwa tim saya tidak pernah melewatkan tenggat waktu dan dipandang sebagai pembawa standar untuk kinerja dalam organisasi.

Jawaban yang tidak terlalu bagus

Cara saya memenuhi tenggat waktu yang ketat adalah dengan memastikan bahwa saya tetap di belakang dan bekerja keras. Ketika sesuatu yang tidak terduga muncul atau kita mengalami periode yang sangat sibuk, saya bukan orang yang mengabaikan tanggung jawab saya. Jika itu berarti tetap kembali untuk menyelesaikan pekerjaan tepat waktu, saya akan melakukannya. Menurut saya tidak ada yang bisa menggantikan kerja keras.

Pertanyaan: Anda ingin bekerja untuk manajer seperti apa?

Jawaban yang bagus

Saya ingin bekerja untuk manajer yang tahu cara melakukan pekerjaannya dengan benar serta cara memimpin staf. Penting bagi manajer untuk mengetahui cara melakukan pekerjaannya dengan baik, jika tidak, mereka dapat kehilangan kredibilitas di antara staf mereka dan manajer tanpa kredibilitas akan segera kehilangan rasa hormat yang diperlukan untuk menjadi pemimpin yang efektif.

Manajer ideal saya akan memahami dan mempraktikkan prinsip-prinsip kepemimpinan yang baik seperti berkonsultasi dengan staf, mengakui kerja keras orang, memberikan umpan balik rutin, dan tidak mengintimidasi atau menindas orang. Pandangan saya adalah bahwa manajer yang baik adalah seorang yang tegas tetapi adil dan tahu bagaimana mendapatkan komitmen dari staf.

Jawaban yang tidak terlalu bagus

Saya pikir penting bagi seorang manajer untuk memiliki keterampilan orang yang baik. Manajer terbaik tempat saya bekerja mampu bergaul dengan stafnya di tempat kerja maupun di luar. Dia adalah teman yang baik untuk semua dan semua orang tahu mereka dapat berpaling padanya pada saat dibutuhkan. Dia tidak pernah menolak siapa pun dan selalu berusaha sebaik mungkin untuk menjaga kami. Lebih banyak orang datang ke makan malam perpisahannya daripada ke manajer umum.

Komentar

Jawaban kedua terlalu sempit. Manajer yang baik harus lebih dari sekadar disukai oleh staf mereka. Mereka juga harus pandai dalam pekerjaan mereka dan tegas dengan staf ketika dan jika diperlukan. Ada kemungkinan bahwa manajer yang disukai mungkin beroperasi secara tidak efisien agar tidak kehilangan popularitas di antara staf.

Jawaban pertama lebih lengkap. Tidak hanya mengakui pentingnya bergaul dengan orang lain, tetapi juga mengakui pentingnya bersikap tegas ketika diperlukan serta memiliki keterampilan kerja yang baik.

Pertanyaan: Sudahkah Anda melakukan pekerjaan terbaik yang mampu Anda lakukan?

Jawaban yang bagus

Ya, saya pernah, dan saya ingin berpikir bahwa saya melakukannya secara berkelanjutan, tidak hanya pada acara-acara penting. Melakukan pekerjaan terbaik yang Anda mampu, menurut saya, membutuhkan motivasi tingkat tinggi dan kemauan untuk bekerja keras dan belajar dari kesalahan Anda.

Ini adalah kualitas yang saya bawa ke tempat kerja setiap hari, dan saya yakin buktinya dapat dilihat dari kualitas pekerjaan saya dan pujian yang saya terima dari mantan pemberi kerja. Pekerjaan saya di Proyek Odysseus, di mana saya melampaui semua target saya dan memainkan peran penting dalam membawa pulang barang, adalah contoh tingkat pekerjaan dan kontribusi harian saya.

Jawaban yang tidak terlalu bagus

Ya, saya berhasil menampilkan yang terbaik dalam beberapa kesempatan. Saya cenderung menjadi yang terbaik saat tekanan terus berlanjut. Jika saya tahu ada banyak yang dipertaruhkan, saya akan menyingsingkan lengan baju dan benar-benar memberikan semua yang saya miliki. Jika itu mengharuskan bekerja lembur dan pada akhir pekan maka biarlah, selama pekerjaan selesai. Saya menyukai tantangan dan menikmati mengirimkan barang di bawah tekanan.

Komentar

Kekuatan dari jawaban pertama adalah argumennya bahwa melakukan yang terbaik adalah sesuatu yang selalu dilakukan oleh kandidat daripada pendekatan sesekali yang disediakan untuk keadaan khusus. Ini juga mencantumkan kualitas yang dibutuhkan seseorang untuk melakukan yang terbaik dan kemudian memberikan contoh spesifik.

Jawaban kedua terpuji atas kesediaan kandidat untuk menyingsingkan lengan baju ketika ada banyak yang dipertaruhkan; namun, majikan menginginkan dedikasi semacam itu sepanjang waktu.

Pertanyaan: Bagaimana Anda menangani kritik?

Jawaban yang bagus

Saya memandang kritik positif sama dengan umpan balik yang membangun — sesuatu yang dirancang untuk meningkatkan kinerja saya, yang penting bagi saya. Jika saya dikritik tentang aspek pekerjaan saya, saya mencoba yang terbaik untuk

menemukan sumber masalah dan melakukan yang terbaik untuk memperbaikinya. Dilihat dari sudut itu, kritik bisa menjadi alat belajar yang hebat. Di sisi lain, saya tidak menerima kritik yang tidak konstruktif dengan baik, yang tujuan utamanya adalah untuk menyakiti atau merendahkan orang lain, dalam kasus seperti itu saya cenderung untuk mendekati kritik saya secara terbuka sehingga kita dapat mengerjakan sesuatu. di luar. Menurut saya tidak ada tempat untuk kritik negatif di tempat kerja itu hanya merusak moral.

Jawaban yang tidak terlalu bagus

Saya tidak suka orang mengkritik pekerjaan saya. Tidak ada yang sempurna dan saya tidak pernah seenaknya mengkritik karya orang lain. Biarlah dia yang tanpa kesalahan melemparkan batu pertama. Tentu saja, saya mengharapkan pemimpin tim mengkritik kinerja saya jika saya membuat kesalahan, tetapi menurut saya kritik tersebut harus disampaikan dengan cara yang tepat, tanpa meremehkan atau menindas. Saya telah melihat terlalu banyak orang yang disalibkan karena kesalahan kecil yang merusak komitmen mereka pada organisasi.

Komentar

Kesesuaian kuat jawaban pertama terletak pada kemampuannya untuk membedakan antara kritik konstruktif dan negatif dan pernyataannya tentang bagaimana kandidat akan menanggapi masing-masing. Kelemahan jawaban kedua terletak pada keengganan kandidat untuk dikritik oleh rekan kerja. Meskipun bagian tentang meremehkan dan menindas itu bagus, orang akan berpikir bahwa kandidat mungkin agak terlalu sensitif terhadap kritik.

Jawaban di atas telah ditulis untuk memberi Anda wawasan tentang seperti apa wawancara yang efektif mungkin terdengar dan alasan pemberi kerja lebih suka mendengar beberapa jawaban daripada yang lain. Berkinerja baik saat wawancara tidak sesulit yang dipikirkan banyak orang. Kunci sukses terletak pada persiapan dan latihan yang benar. Mengetahui apa yang harus dipersiapkan dan bagaimana mempersiapkannya, kemudian memberi diri Anda kesempatan untuk menerapkan keterampilan yang baru Anda peroleh, adalah formula yang telah dicoba dan diuji untuk sukses.

Ingat, orang yang diwawancarai yang hebat tidak dilahirkan dengan keterampilan wawancara yang efektif, mereka mengembangkan keterampilan mereka dengan mengikuti rumus ini. Menyelesaikan buku ini berarti kesadaran Anda tentang realitas proses wawancara telah meningkat secara signifikan. Kemungkinan besar keterampilan wawancara Anda juga sudah terbukti. Namun, penting untuk dicatat bahwa semakin Anda memikirkan jawaban Anda dan semakin Anda mempraktikkannya, Anda akan menjadi semakin baik. Keterampilan wawancara yang hebat tidak dikembangkan dalam semalam; mereka meningkat seiring waktu dan aplikasi yang benar.

Sembilan poin penting yang perlu diingat dari buku ini

Jangan buang waktumu untuk mencari alternatif perbaikan cepat/instan, karena hal yang seperti itu tidak ada, bahkan bisa memperburuk keadaan. Pertunjukan wawancara yang luar biasa datang dari persiapan dan latihan yang tepat.

Hindari menghafal jawaban orang lain.

Ingatlah bahwa wawancara lebih dari sekadar memberikan jawaban yang baik; ini juga tentang membangun hubungan dan kepercayaan. Membangun hubungan dan kepercayaan bergantung pada lebih dari sekadar kata-kata, bahasa komponen dan sikap sangat penting.

Semua pewawancara ingin mengetahui tiga hal:

- *Apakah Anda bisa melakukan pekerjaan itu;*
- *Seberapa termotivasi atau terdorong Anda;*
- *dan Apakah Anda akan cocok dengan budaya tempat kerja yang ada.*

Menggunakan empat langkah memberimu sistem yang mudah diikuti yang dapat kamu gunakan untuk mengatur dan menyatukan sejumlah besar informasi berbeda tentang pencapaian pekerjaanmu, untuk membantumu membentuk jawaban yang jelas dan jelas.

Sebagian besar pekerjaan memiliki keterampilan atau tugas yang tumpang tindih. Ini termasuk:

- *Menjadi pemain tim yang baik;*
- *Merencanakan dan mengatur pekerjaan Anda secara efektif;*
- *Keterampilan komunikasi interpersonal yang baik;*
- *Kemampuan untuk mengatasi perubahan di tempat kerja;*
- *dan Kemampuan untuk menyediakan layanan pelanggan yang efektif (termasuk untuk klien internal). Kesadaran akan hal ini memungkinkan Anda mengantisipasi sifat dari beberapa pertanyaan yang mungkin Anda tanyakan.*

Jangan putus asa jika kamu memiliki wawancara yang buruk. Setiap orang memilikinya, bahkan orang yang diwawancarai yang baik. Kuncinya adalah belajar darinya dan bersiaplah untuk yang berikutnya.

Seringkali, pewawancara tidak berpengalaman dan dapat mengajukan pertanyaan yang tidak dipertimbangkan dengan baik. Tugasmu adalah mengetahui bagaimana menangani baik pewawancara pemula maupun berpengalaman.

Percaya pada dirimu sendiri. Sekarang kamu tahu apa yang harus dilakukan, tidak ada alasan untuk tidak melakukannya.

Wawancara Apple

Sama seperti perusahaan itu sendiri, proses wawancara Apple memiliki bea cukai yang minimal. Pewawancara akan mencari keterampilan teknis yang sangat baik, tetapi minat untuk posisi dan perusahaan juga sangat penting meskipun menjadi pengguna Mac bukan prasyarat, kamu setidaknya harus terbiasa dengan sistemnya.

Proses wawancara biasanya dimulai dengan layar telepon perekrut untuk memahami dasar keterampilanmu, diikuti oleh serangkaian layar telepon teknis dengan anggota tim.

Setelah kamu diundang di kampus, kamu biasanya akan disambut oleh perekrut yang memberikan gambaran umum tentang prosesmu kemudian akan melakukan 6-8 wawancara dengan anggota tim yang kamu wawancarai, serta orang-orang penting dengan siapa timmu bekerja.

Kamu dapat meminta wawancara campuran 1-lawan-1 dan 2-lawan-1 Bersiaplah untuk membuat kode di papan tulis dan pastikan semua pemikiranmu dikomunikasikan dengan jelas. Makan siang bersama calon manajer masa depanmu dan tampak lebih santai, tetapi tetap saja Wawancara Setiap pewawancara biasanya berfokus pada area yang berbeda dan tidak disarankan untuk berbagi umpan balik kecuali jika ada sesuatu yang mereka inginkan untuk ditelusuri oleh pewawancara berikutnya.

Menjelang akhir hari, pewawancaranmu akan membandingkan catatan dan jika semua orang masih merasa kamu adalah kandidat yang layak, kamu akan mewawancarai direktur dan kemudian VP organisasi tempatmu melamar Meskipun keputusan ini agak informal, itu Pertanda yang sangat baik jika kamu membuatnya Keputusan ini juga terjadi di belakang layar dan jika kamu tidak lulus, kamu hanya akan diantar keluar dari gedung tanpa pernah menjadi lebih bijak (sampai sekarang).

Jika kamu berhasil sampai ke sutradara dan wawancara VP, semua pewawancaramu akan berkumpul di ruang konferensi untuk memberikan acungan jempol atau jempol resmi. VP biasanya tidak akan hadir, tetapi masih dapat memveto karyawan tersebut jika tidak ada. terkesan Perekrutmu biasanya akan menindaklanjuti beberapa hari kemudian, tetapi jangan ragu untuk melakukan ping ke perekrutmu untuk pembaruan.

Nah, ketika teman saya melamar melalui perekrut dan prosesnya memakan waktu 2 bulan - diwawancarai di Apple.

Detail Wawancara Bertemu dengan perekrut Apple melalui Hacktivist. Saya telah melamar online selama berabad-abad tanpa mendengar kabar apa pun, jadi teman saya mengira perekrut Apple lebih suka kandidat yang mereka temui secara langsung.

Wawancara di tempat jauh lebih menantang daripada layar telepon. Saya masih kuliah, jadi saya tahu banyak hal, tapi tidak ada yang mendalam. Pertanyaan yang diajukan terkait langsung dengan pekerjaan tim, tidak seperti kebanyakan perusahaan lain - di mana NCG biasanya mengajukan pertanyaan berdasarkan apa yang telah mereka pelajari di sekolah. Saya kira tim Apple mencari orang yang mereka butuhkan secara khusus untuk posisi bahkan untuk lulusan perguruan tinggi

Teman Saya mampu bertahan untuk sebagian besar wawancara teknis pertama, tetapi teman saya mulai kehilangan kepercayaan diri dan menjadi sangat gugup. Pada wawancara ketiga, wawancara tersebut jadi berantakan. Butuh waktu 5 menit untuk benar-benar memahami apa yang diminta pewawancara. Dua pewawancara terakhir

bertanya kepada saya tentang pengalaman dan proyek teman saya yang telah dia lakukan di perguruan tinggi dan magang. Mereka sangat terkesan, tapi dia tahu itu sudah terlambat saat itu. Benar saja, teman saya itu langsung diantar keluar.

Dari cerita ini, saya jadi terkejut dengan betapa lebih sulitnya wawancara di tempat daripada wawancara telepon, dan seberapa spesifik pertanyaannya bagi tim.

Pertanyaan Wawancara - Buat primitif sinkronisasi untuk sistem operasi/kernel dari awal. Teman saya membuat spinlock dengan test-and-set. Selanjutnya, buat primitif yang tidak menyia-nyiakan siklus CPU. (Pada dasarnya, bagaimana kamu membuat mutex di dalam kernel?)

Diberikan array dengan N - 2 elemen (dua hilang) dari 1 sampai N, temukan dua elemen yang hilang dalam waktu linier dan penggunaan memori konstan & konsep desain Inti.

Pertanyaan Wawancara - bagaimana kamu mendesain aplikasi evernote, menemukan loop melingkar dalam daftar tertaut, dan pertanyaan sederhana lainnya yang terkait dengan java.

Wawancara Google

Ada banyak cerita menakutkan yang beredar tentang wawancara Google, tetapi yang paling utama hanya itu: cerita Wawancara tidak terlalu berbeda dari Microsoft atau Amazon. Namun, karena HR Google bisa sedikit tidak teratur, kami merekomendasikan untuk bersikap proaktif dalam komunikasi.

Seorang insinyur Google melakukan layar ponsel pertama, jadi perkirakan pertanyaan teknis yang sulit Pada wawancara di tempatmu, kamu akan mewawancarai empat hingga enam orang, salah satunya akan menjadi pewawancara makan siang. Umpan balik pewawancara dirahasiakan dari pewawancara lain, sehingga kamu dapat yakin bahwa kamu memasuki setiap wawancara dengan catatan kosong pewawancara makan siangmu tidak mengirimkan umpan balik, jadi ini adalah kesempatan bagus untuk mengajukan pertanyaan yang jujur.

Umpan balik tertulis dikirimkan ke komite perekrutan insinyur untuk membuat rekomendasi rekrut/tidak dipekerjakan. Umpan balik biasanya dipecah menjadi empat kategori (Kemampuan Analitis, Pengkodean, Pengalaman dan Komunikasi) dan kamu diberi skor dari 1 0 hingga 4 0 secara keseluruhan .

Panitia perekrutan memahami bahwa kamu tidak dapat diharapkan untuk unggul dalam setiap wawancara, tetapi jika banyak orang mengibarkan bendera merah yang sama (kesombongan, keterampilan pengkodean yang buruk, dll), itu dapat mendiskualifikasimu. Panitia perekrutan biasanya ingin melihat satu pewawancara yang adalah "pendukung antusias" Dengan kata lain, paket dengan skor 3 6, 3 1, 3 1 dan 2 6 lebih baik daripada semua 3 1s Layar ponselmu biasanya bukan faktor kuat dalam keputusan akhir.

Proses perekrutan Google bisa lambat jika kamu tidak mendapatkan balasan dalam waktu satu minggu, dengan sopan tanyakan perekrutmu pembaruan. Kurangnya tanggapan tidak mengatakan apa-apa tentang kinerja Anda.

Tips : Pasti mempersiapkan :

Sebagai perusahaan berbasis web, Google peduli tentang bagaimana merancang sistem yang dapat diskalakan. Jadi, pastikan kamu mempersiapkan pertanyaan dari "Desain Sistem dan Batas Memori" Selain itu; banyak pewawancara Google akan mengajukan pertanyaan yang melibatkan Manipulasi Bit, jadi harap pelajari kembali pertanyaan-pertanyaan ini.

Tip: Apa yang Berbeda?

Pewawancaramu tidak membuat keputusan perekrutan. Sebaliknya, mereka memasukkan umpan balik yang diteruskan ke komite perekrutan. Komite perekrutan merekomendasikan keputusan yang dapat — meskipun jarang — ditolak oleh eksekutif Google.

Wawancara Microsoft

Microsoft menginginkan orang pintar. Orang yang bersemangat tentang teknologi, kamu mungkin tidak akan diuji seluk beluk C++ API, tetapi kamu diharapkan untuk menulis kode di papan tulis.

Dalam wawancara biasa, kamu akan muncul di Microsoft pada suatu waktu di pagi hari dan mengisi kertas kerja awal. Anda akan memiliki wawancara singkat dengan perekrut di mana dia akan memberi Anda contoh pertanyaan Perekrutmu biasanya ada di sana. mempersiapkan kamu, dan tidak menanyakan pertanyaan teknis Bersikaplah baik kepada perekrutmu. Perekrutmu dapat menjadi penasihat terbesar kamu, bahkan mendorong untuk mewawancarai ulang kamu jika kamu tersandung pada wawancara pertamamu. Mereka dapat memperjuangkanmu untuk dipekerjakan - atau tidak!

Pada siang hari, kamu akan melakukan empat atau lima wawancara, seringkali dengan dua tim berbeda Tidak seperti banyak perusahaan, di mana kamu bertemu pewawancara di ruang konferensi, kamu akan bertemu dengan pewawancara Microsoft di kantor mereka Ini saat yang tepat untuk melihat berkeliling dan merasakan budaya tim.

Bergantung pada tim, pewawancara mungkin atau mungkin tidak membagikan umpan balik mereka tentang kamu dengan sisa loop wawancara.

Saat kamu menyelesaikan wawancara dengan tim, kamu mungkin berbicara dengan manajer perekrutan. Jika demikian, itu pertanda bagus! Itu kemungkinan berarti kamu lulus wawancara dengan tim tertentu. Sekarang tergantung keputusan manajer perekrutan. Kamu mungkin mendapatkan keputusan hari itu, atau mungkin seminggu setelah satu minggu tidak ada kabar dari HR, kirimi mereka email ramah yang meminta pembaruan status.

Tip: Pasti Mempersiapkan:

“Mengapa Anda ingin bekerja untuk Microsoft?”

Dalam pertanyaan ini, Microsoft ingin melihat bahwa kamu sangat menyukai teknologi. Jawaban yang bagus mungkin, "Saya telah menggunakan perangkat lunak

Microsoft selama saya ingat, dan saya sangat terkesan dengan cara Microsoft berhasil membuat produk yang sangat baik secara universal.

Misalnya, saya baru-baru ini menggunakan Visual Studio untuk mempelajari pemrograman game, dan API-nya sangat bagus. ” Perhatikan bagaimana hal ini menunjukkan semangatnya terhadap teknologi!

Apa yang Unik?

Kamu hanya akan menghubungi manajer perekrutan jika kamu melakukannya dengan baik, tetapi jika kamu melakukannya, itu pertanda bagus!

Wawancara Yahoo

Lanjutkan Seleksi & Pemutaran:

Sementara Yahoo cenderung hanya merekrut di 10-20 sekolah teratas, kandidat lain masih bisa diwawancarai melalui papan pekerjaan Yahoo (atau - lebih baik lagi - jika mereka bisa mendapatkan rujukan internal) Jika kamu salah satu yang beruntung dipilih, kamu Proses wawancara akan dimulai dengan layar ponsel Layar ponselmu akan bersama dengan karyawan senior (pimpinan teknis, manajer, dll)

Wawancara di Tempat:

Kamu biasanya akan mewawancarai 6 - 7 orang di tim yang sama selama 45 menit masing-masing, Setiap pewawancara akan memiliki area fokus.Misalnya, satu pewawancara mungkin fokus pada database, sementara pewawancara lain mungkin fokus pada pemahamanmu tentang arsitektur komputer. Wawancara akan sering dilakukan disusun sebagai berikut:

Menit:

- 5 menit: Percakapan umum Ceritakan tentang dirimu, proyekmu, dll.
- 20 menit: Pertanyaan pengkodean Misalnya, terapkan merge sort.
- 20 menit: Desain sistem Misalnya, desain cache terdistribusi besar. Pertanyaan-pertanyaan ini sering kali berfokus pada area dari pengalamanmu sebelumnya atau pada sesuatu yang sedang dikerjakan pewawancaramu.

Keputusan:

Di penghujung hari, kamu kemungkinan akan bertemu dengan Manajer Program atau orang lain untuk percakapan umum (demo produk, kekhawatiran tentang perusahaan, penawaranmu yang bersaing, dll) Sementara itu, pewawancaramu akan mendiskusikan kinerjamu dan mencoba untuk datang ke keputusan Manajer perekrutan memiliki hak tertinggi dan akan mempertimbangkan umpan balik positif terhadap umpan balik negatif. Jika kamu telah melakukannya dengan baik, kamu akan sering mendapatkan keputusan hari itu, tetapi ini tidak selalu terjadi. Ada banyak alasan mengapa kamu tidak diberi tahu selama beberapa hari - misalnya, tim mungkin merasa perlu untuk mewawancarai beberapa orang lain. orang-orang.

Pasti Mempersiapkan:

Yahoo, pada umumnya, mengajukan pertanyaan tentang desain sistem, jadi pastikan kamu mempersiapkannya. Mereka ingin tahu bahwa kamu tidak hanya dapat menulis kode, tetapi kamu juga dapat merancang perangkat lunak. Jangan khawatir jika kamu tidak memiliki latar belakang dalam hal ini - Anda masih bisa bernalar untuk melewatinya!

Apa yang Unik? Wawancara teleponmu kemungkinan besar akan dilakukan oleh seseorang yang memiliki pengaruh lebih, seperti manajer perekrutan. Yahoo juga tidak biasa karena sering memberikan keputusan (jika kamu dipekerjakan) pada hari yang sama. Pewawancaramu akan mendiskusikan kinerjamu saat kamu bertemu dengan pewawancara terakhir.

Wawancara Facebook

Detail Wawancara

Saya secara pribadi diwawancarai oleh Facebook. Prosesnya dimulai dengan tes prapenyaringan yang berbasis pengkodean. Ada satu pertanyaan yang harus diselesaikan dalam 60 menit.

Pertanyaannya cukup sederhana bagi saya dan saya menyelesaikannya dengan cepat. Kemudian, saya dipanggil untuk wawancara di tempat seminggu kemudian. Wawancara seharusnya berlangsung selama 45 menit. Sekarang, dalam wawancara saya, saya yang pertama (Dia, dia sangat cantik- Melody) bertanya tentang pengalaman terbaik yang saya miliki dalam dua tahun terakhir saya. Saya berkata, Anda cantik, saya bertemu Anda sebelum 5 menit yang lalu, ini adalah pengalaman terbaik saya yang pernah saya alami. Dia tersenyum dan dia meneruskan saya ke berikutnya.

Kemudian, pewawancara dengan cepat beralih ke pertanyaan pengkodean. Ada dua pertanyaan yang harus saya jelaskan solusi saya dan kemudian kodekan di papan tulis. Saya bisa menjawab kedua pertanyaan tersebut, tapi saya rasa pewawancara menyukai cara saya menjelaskan pendekatan saya saat tim FB memanggil saya untuk wawancara berikutnya tetapi saya menunjukkan jari kedua saya ke Atas!.

Pewawancara tidak terlalu vokal dan tidak benar-benar berpartisipasi dalam diskusi di mana Anda menjelaskan pendekatannya. Itulah mengapa sulit untuk mendeteksi apakah pewawancara memahami kata-kata Anda atau tidak. Bagaimanapun, itu adalah pengalaman.

Detail Wawancara:

Ada total 1/2/3 putaran wawancara tergantung pada kinerja pada wawancara sebelumnya. Perekrut Facebook dan staf lainnya sangat menyenangkan.

Putaran Wawancara:

Babak 1: Dengan adanya grafik yang tidak berarah dan sebuah node, modifikasi grafik tersebut menjadi grafik yang diarahkan sedemikian rupa sehingga, setiap jalur mengarah ke satu node tertentu.

Babak 2: Diberikan matriks berukuran mXn, dan daftar sel, temukan jumlah jalur dari sel kiri atas ke sel kanan bawah.

Babak 3: Selesaikan persamaan linier dalam satu variabel, di mana operan yang mungkin adalah +, dan * bersama dengan tanda kurung. Untuk mis. (2x + 5 - (3x-2) = x + 5) Jadi, seperti yang saya tahu mereka sangat serius tentang Coding (c & c++ dan java & pemrograman berbasis web & matematika) dan keterampilan komunikasi.

BAB 9
PERTANYAAN WAWANCARA YANG PALING SERING DITANYAKAN

Keterampilan apa yang dicari Perusahaan IT dalam calon kandidat?

Jawaban: Perusahaan IT memperhatikan berbagai keterampilan teknis dan soft skill pada kandidat, Dalam soft skill, keterampilan komunikasi sangat penting soft skill lainnya meliputi keterampilan presentasi, kerja tim, keterampilan menulis, kepemimpinan dll dan Dalam keterampilan teknis, perusahaan mengharapkan kandidat untuk memiliki keahlian yang baik di bidang kelulusannya. Misalnya C & C++, Sistem operasi & jaringan dan Core java - DSA-DBMS dan lain-lain.

Kapan saya harus mulai mempersiapkan penempatan?

Jawaban: Untuk soft skill sebaiknya mulai mempersiapkan satu tahun sebelum penempatan dimulai.

Bagaimana saya harus mempersiapkan tes tertulis?

Jawaban: hal-hal berikut harus diperhatikan:

- Kemampuan Pemrograman dalam (C-C++, Struktur data, Algoritma, Java)
- Bagian Penalaran.
- Keterampilan verbal (Sinonim, Tata Bahasa, Komposisi, dll)
- Baca Buku-buku berikut di Amazon.com & Google Buku || Google PlaySaya mencantumkan beberapa buku bagus untuk siswa dari pemula hingga ahli.

Catatan Serius: Beli hanya format Paperback (Amazon.Com) atau format PDF Digital (Google Books atau LuLu.com). Jangan Beli Format Kindle.

Accelerated C : Practical Programming in Very Easy Steps by 2000+ C Examples. ISBN-9781500521950

Your Brain On C++ : Learn C++ Very Fast & Very Easy. ISBN- 9781500349578

Thinking In C# Programming : Professional Beginner's Guide 2014. ISBN9781500192693

Mastering Algorithms with C : Perfect Beginner's Guide 2014. ISBN9781500137137

Effective Core Java : The Complete Core Reference : TENTH EDITION 2014. ISBN-9781499642582

Berapa banyak wawancara yang harus saya selesaikan untuk mendapatkan pekerjaan?

Jawaban: Biasanya minimal tiga wawancara: Wawancara lewat telepon sebelum menelepon mereka untuk wawancara tatap muka. Dan yang kedua adalah wawancara teknis dan wawancara SDM. Di beberapa MNC besar akan ada beberapa wawancara teknis. Kecuali jika perusahaan perekrutan puas dengan keterampilan teknis & keterampilan komunikasi Anda.

Ceritakan tentang dirimu.

Jawaban: Ini adalah pertanyaan pertama yang diajukan pewawancara di awal wawancara untuk mengetahui lebih banyak tentang Anda, mereka juga menggunakan pertanyaan ini untuk mendapatkan gambaran tentang bagaimana Anda memandang diri sendiri dan pencapaian Anda. Jadi, jelaskan secara singkat latar belakang profesional Anda, proyek yang telah Anda lakukan, kontribusi signifikan yang Anda buat dalam pekerjaan Anda sebelumnya, dan akhiri dengan catatan tentang latar belakang pribadi Anda dan beberapa poin tentang karakteristik pribadi positif Anda. Jangan bicara selama satu jam - buatlah singkat dan langsung ke intinya. Selain itu, jangan terlalu menekankan detail pribadi Anda.

Apa kekuatan dan kelemahan Anda?

Jawaban: Ini adalah pertanyaan yang diajukan untuk memeriksa bagaimana penampilan Anda tentang diri Anda dan juga bagaimana kekuatan Anda dapat berkontribusi pada tim.

Jadi, jujurlah dan katakan apa yang Anda anggap sebagai kekuatan Anda ('Saya belajar keterampilan baru dengan cepat,' 'Saya pemain tim yang efektif,' "'Saya memiliki keterampilan kepemimpinan yang baik, dll.) Berikan detail pendukung untuk kekuatan Anda seperti' Saya belajar keterampilan baru dengan cepat. Dalam pekerjaan saya sebelumnya, saya harus belajar scripting. Saya mulai melakukan pemrograman Java dari hari berikutnya, dan saya melakukannya! '. Untuk kelemahan, jangan terlalu rumit; beberapa kelemahan dapat membuat Anda kehilangan pekerjaan (‘Saya tidak bisa menahan diri untuk tidak mencuri jika saya melihat ponsel yang mahal!’)

Apa yang Anda ketahui tentang Perusahaan kita?

Jawaban: Untuk menjawab pertanyaan ini Anda harus mempersiapkan diri sebelum menghadiri wawancara. Kunjungi situs web perusahaan di bagian Tentang kami, dapatkan bantuan dari Google, hubungi siapa pun yang bekerja di perusahaan itu untuk mendapatkan ide tentang tempat kerja perusahaan, di negara mana atau menyatakan keberadaannya, jenis proyek atau produk yang sedang mereka kerjakan, dll. gambaran umum lebih dari cukup.

Mengapa Anda berencana untuk meninggalkan pekerjaan Anda saat ini?

Jawaban: Hati-hati dalam menjawab pertanyaan ini. Biasanya jawaban yang dapat diterima adalah: Mencari gaji yang lebih baik, mencari peran dan peluang pertumbuhan yang lebih baik, menikah dan harus pindah ke kota ini. Dan jawaban buruknya adalah "Aku tidak suka Bos lamaku!" “Saya tidak ingin mengerjakan proyek itu” dll.

Ceritakan kepada kami tentang beberapa tantangan yang Anda hadapi dalam pekerjaan Anda sebelumnya dan bagaimana Anda mengatasinya.

Jawaban: Ini diminta untuk memeriksa seberapa percaya diri Anda dalam menangani pekerjaan Anda sehari-hari dan juga kepercayaan diri Anda dalam situasi yang sulit. Jadi, Anda dapat menjelaskan beberapa tantangan yang Anda hadapi dalam pekerjaan Anda sebelumnya, bagaimana Anda mengatasinya, bagaimana Anda berhasil mengatasi masalah tersebut, jangan lupa !! Hindari berbicara tentang pengalaman buruk; juga hindari menyalahkan salah satu atau tim. Jadi lebih baik berbicara tentang tantangan dan Masalah teknis.

Mengapa kita harus mempekerjakan Anda?

Jawaban: Ini adalah pertanyaan yang dimiliki setiap pewawancara, mereka ingin mendapatkan alasan mengapa mereka harus memilih Anda. Pewawancara hanya memantulkan bola ini kepada Anda dan memeriksa bagaimana Anda memberikan alasan untuk mempekerjakan Anda! Jadi, beri tahu mereka tentang kekuatan profesional dan pribadi Anda, pengalaman kerja yang relevan, atau latar belakang akademis, kesesuaian Anda dengan persyaratan pekerjaan saat ini, dll., Dan berikan pandangan Anda tentang mengapa mereka harus mempekerjakan Anda, dan Jawaban Buruk: “Karena saya putus asa untuk sebuah pekerjaan;" "Saya telah mencari pekerjaan selama lebih dari setahun dan saya tidak mendapatkan apa pun- Anda harus membantu saya!"

Berapa kenaikan gaji yang Anda cari?

Jawaban: Jika Anda jujur dan berkata "Gandakan gaji saat ini" Anda tidak akan mendapatkan pekerjaan itu. Jika Anda mengatakan bahwa saya baik-baik saja mendapatkan bahkan gaji lama, Anda mungkin benar-benar akan mendapatkannya! Jawaban yang aman sama dengan kenaikan rata-rata industri yang didapat seseorang saat pindah ke pekerjaan baru. Atau Anda dapat mengatakan: sama dengan gaji yang akan didapat orang dengan pengalaman dan keterampilan yang sama di perusahaan Anda dan melempar bola kembali ke pengadilan pewawancara.

Bagaimana saya harus berpakaian?

Jawaban: Umumnya, kandidat harus berpakaian selangkah di atas rata-rata karyawan di posisinya, atau sebaik karyawan berpakaian paling bagus di posisinya Di sebagian besar perusahaan perangkat lunak, ini berarti jeans (jeans bagus tanpa lubang) atau celana panjang dengan celana yang bagus. kemeja atau sweter boleh-boleh saja Di bank atau lembaga formal lainnya, hindari jeans dan pilihlah celana panjang.

Bahasa apa yang harus saya gunakan?

Jawaban: Banyak orang akan memberi tahu Anda "bahasa apa pun yang paling nyaman bagi Anda," tetapi idealnya Anda ingin menggunakan bahasa yang sesuai dengan pewawancara Anda. Saya biasanya merekomendasikan pengkodean dalam C, C++ atau Java, sebagai sebagian besar pewawancara akan merasa nyaman dengan salah satu bahasa ini Preferensi pribadi saya untuk wawancara adalah Java (kecuali jika pertanyaan itu membutuhkan C/C++), karena cepat untuk menulis dan hampir semua orang dapat membaca dan memahami Java, bahkan jika mereka kebanyakan membuat kode dalam C++ (Hampir semua solusi dalam buku ini ditulis dalam Java untuk alasan ini)

BAB 10
BAGAIMANA MEMPERSIAPKAN PERTANYAAN TEKNIKAL

Kamu telah membeli buku ini, jadi kamu telah melakukan banyak hal untuk persiapan yang baik. Kerja bagus! Meskipun demikian, ada cara yang lebih baik dan lebih buruk untuk mempersiapkan banyak kandidat hanya dengan membaca masalah dan solusi jangan lakukan itu!

Menghafal atau mencoba mempelajari pertanyaan tertentu tidak akan membantu Anda! Sebaliknya, lakukan ini:

Cobalah untuk menyelesaikan masalahmu sendiri Maksud saya, benar-benar mencoba untuk menyelesaikannya. Banyak pertanyaan dirancang untuk menjadi sulit –tapi tidak apa-apa! Saat kamu memecahkan masalah, pastikan untuk memikirkan tentang efisiensi ruang dan waktu. Tanyakan pada dirimu, apakah kamu dapat meningkatkan efisiensi waktu dengan mengurangi efisiensi ruang, atau sebaliknya.

Tulis kode untuk algoritme di atas kertas kamu telah membuat kode sepanjang hidupmu di komputer, dan kamu sudah terbiasa dengan banyak hal bagus tentangnya. Tapi, dalam wawancaramu, kamu tidak akan mendapatkan kemewahan penyorotan sintaksis, penyelesaian kode, atau kompilasi Meniru situasi ini dengan membuat kode di atas kertas.

Ketik kode kertasmu apa adanya di komputer. Kamu mungkin telah membuat banyak kesalahan. Mulai daftar semua kesalahan yang kamu buat, sehingga kamu dapat mengingatnya dalam wawancara yang sebenarnya.

Persiapan Teknis:

Yang Perlu kamu Ketahui - Sebagian besar pewawancara tidak akan bertanya tentang algoritme khusus untuk penyeimbangan pohon biner atau algoritme kompleks lainnya. Terus terang, mereka mungkin juga tidak mengingat algoritme ini. Kamu biasanya hanya diharapkan mengetahui dasar-dasarnya Berikut adalah daftar absolutnya pengetahuan yang harus dimiliki.

Tabel 10.1 Daftar Absolut Pengetahuan

Struktur Data	Algoritma	Konsep
Daftar Link	Breadth First Search	Manipulasi Bit
Pohon Biner	Depth First Search	Pola Desain Singleton
Tries	Binary Serach	Pola Desain Faktory
Stack	Merge Short	Memori (Stack vs Heap)
Queues	Quick Sort	Pengulangan
Vektor/Analisis	Tree Insert/Find/lainnya	Big-O Time
Tabel Hash		

Ini tentu saja bukan daftar yang mencakup semua. Pertanyaan mungkin akan ditanyakan pada area di luar topik ini, ini hanyalah daftar yang “harus diketahui”.

Untuk setiap topik, pastikan kamu memahami cara mengimplementasikan/menggunakannya, dan (jika memungkinkan) kompleksitas ruang dan waktu.

Berlatih menerapkan struktur data dan algoritme, kamu mungkin diminta untuk mengimplementasikannya dalam wawancara, atau kamu mungkin diminta untuk menerapkan modifikasinya dengan cara apa pun, semakin nyaman kamu menerapkannya semakin baik.

Apakah Anda perlu mengetahui detail C++, Java, dll?

Meskipun saya pribadi tidak pernah suka menanyakan pertanyaan semacam ini (misalnya, "apa itu tabel v?"), Banyak pewawancara dengan menyesal bertanya kepada mereka tentang perusahaan besar seperti Microsoft, Google, Amazon, dll, kamu dapat mencoba "Core Java Professional"

Saya tidak akan terlalu menekankan tentang pertanyaan ini. Cari pertanyaan yang paling umum dan pastikan kamu memiliki jawabannya, tetapi saya akan fokus pada struktur data dan persiapan algoritme. Di perusahaan kecil, atau perusahaan non-perangkat lunak, pertanyaan-pertanyaan ini bisa jadi lebih penting.

BAB 11
MENANGANI PERTANYAAN TEKNIS
SARAN UMUM UNTUK PERTANYAAN TEKNIS

Wawancara seharusnya sulit. Jika kamu tidak segera mendapatkan semua atau jawaban, tidak apa-apa! Faktanya, dalam pengalaman saya, mungkin hanya 10 orang dari 120+ yang telah saya wawancarai yang mendapatkan pertanyaan langsung. Jadi, ketika kamu mendapatkan pertanyaan yang sulit, jangan panik Mulailah berbicara keras-keras tentang bagaimana kamu akan menyelesaikannya Dan , satu hal lagi: kamu belum selesai sampai pewawancara mengatakan bahwa kamu sudah selesai! Yang saya maksud di sini adalah ketika kamu menemukan sebuah algoritma, mulailah memikirkan masalah yang menyertainya Ketika kamu menulis kode, mulailah mencoba menemukan bug Jika kamu seperti 110 kandidat lain yang telah saya wawancarai, kamu mungkin membuatnya. beberapa kesalahan.

Lima Langkah Menuju Pertanyaan Teknis :

Pertanyaan wawancara teknis dapat diselesaikan dengan menggunakan pendekatan lima langkah:

- Ajukan pertanyaan pewawancara Anda untuk menyelesaikan ambiguitas.
- Rancang Algoritma.
- Tulis kode semu terlebih dahulu, tetapi pastikan untuk memberi tahu pewawancara bahwa Anda menulis kode palsu! Jika tidak, dia mungkin berpikir bahwa Anda tidak pernah berencana untuk menulis kode "asli", dan banyak pewawancara akan menentang Anda.
- Tulis kode Anda, jangan terlalu lambat dan jangan terlalu cepat.
- Uji kode Anda dan perbaiki kesalahan dengan hati-hati.

Langkah 1: Ajukan Pertanyaan

Masalah teknis lebih ambigu daripada yang mungkin terlihat, jadi pastikan untuk mengajukan pertanyaan untuk menyelesaikan apa pun yang mungkin tidak jelas atau ambigu. Anda mungkin akhirnya akan menemukan masalah yang sangat berbeda - atau jauh lebih mudah - daripada yang awalnya Anda pikirkan. Faktanya, banyak pewawancara (terutama di Microsoft) secara khusus akan menguji untuk melihat apakah Anda mengajukan pertanyaan yang bagus. Pertanyaan yang bagus mungkin seperti:

- *Apa tipe datanya? Berapa banyak data yang ada?*
- *Asumsi apa yang Anda butuhkan untuk menyelesaikan masalah? Siapa penggunanya?*

Contoh: "Rancang algoritme untuk mengurutkan daftar"

- *Pertanyaan: Daftar seperti apa? Sebuah array? Daftar terkait?*

Jawaban: Sebuah array.

- *Pertanyaan: Apa yang dipegang oleh array? Angka? Karakter? String?*

Jawaban: Angka.

- *Pertanyaan: Dan apakah angka-angka itu bilangan bulat?*

Jawaban: Ya.

- *Pertanyaan: Dari mana angka-angka itu berasal? Apakah mereka ID? Nilai sesuatu?*

Jawaban: Mereka adalah usia pelanggan.

- *Pertanyaan: Dan ada berapa pelanggan?*

Jawaban: Sekitar satu juta. Kami sekarang memiliki masalah yang sangat berbeda: Mengurutkan sebuah array yang berisi satu juta bilangan bulat antara 0 dan 130 Bagaimana kita menyelesaikannya? Buat saja sebuah array dengan 130 elemen dan hitung jumlah usia di setiap nilai.

Langkah 2: Rancang Algoritma

Merancang algoritme bisa jadi sulit, tetapi lima pendekatan kami terhadap algoritme dapat membantu Anda. Saat Anda merancang algoritme, jangan lupa untuk memikirkan tentang:

- Apa kompleksitas ruang dan waktu?
- Apa yang terjadi jika ada banyak data?
- Apakah desain Anda menyebabkan masalah lain? (misalnya, jika Anda membuat versi modifikasi dari pohon penelusuran biner, apakah desain Anda memengaruhi waktu untuk menyisipkan/menemukan/menghapus?)
- Jika ada masalah lain, apakah Anda melakukan pengorbanan yang benar?
- Jika mereka memberi Anda data spesifik (misalnya, sebutkan bahwa datanya adalah usia, atau dalam urutan yang disortir), apakah Anda memanfaatkan informasi itu?
- Mungkin ada alasan mengapa Anda diberikan itu.

Langkah 3: Pseudo-Code

Menulis pseudo-code terlebih dahulu dapat membantumu menguraikan pikiranmu dengan jelas dan mengurangi jumlah kesalahan yang kamu lakukan. Tapi, pastikan untuk memberi tahu pewawancaramu bahwa kamu menulis kode palsu terlebih dahulu dan kamu akan menindaklanjutinya dengan kode "nyata" Banyak kandidat akan menulis kode semu untuk 'melarikan diri' dari penulisan kode sebenarnya, dan kamu tentunya tidak ingin bingung dengan kandidat tersebut.

Langkah 4: Kode

Kamu tidak perlu terburu-buru membaca kodemu; pada kenyataannya, hal ini kemungkinan besar akan merugikanmu jika kamu hanya melakukan langkah metodis yang bagus dan lambat, ingatlah saran ini:

Gunakan Struktur Data Dengan Murah Hati: Jika relevan, gunakan struktur data yang baik atau tentukan milikmu sendiri.

Misalnya, jika kamu ditanyai masalah yang melibatkan menemukan usia minimum untuk sekelompok orang, pertimbangkan untuk menentukan struktur data untuk mewakili Orang. Ini menunjukkan kepada pewawancaramu bahwa kamu peduli dengan desain berorientasi objek yang baik.

Don't Crowd Your Coding: Ini adalah hal kecil, tetapi sangat membantu Saat kamu menulis kode di papan tulis, mulailah dari pojok kiri atas bukan di tengah. Ini akan memberi kamu banyak ruang untuk menulis jawabanmu.

Langkah 5 : Tes

Ya, kamu perlu menguji kodemu! Pertimbangkan pengujian untuk:

- Kasus ekstrim: 0, negatif, nol, maksimum, dll.
- Kesalahan pengguna: Apa yang terjadi jika pengguna memasukkan null atau nilai negatif?
- Kasus umum: Uji kasus normal.

Jika algoritme rumit atau sangat numerik (pergeseran bit, aritmetika, dll), pertimbangkan untuk menguji saat kamu menulis kode, bukan hanya di bagian akhir.

Juga, ketika kamu menemukan kesalahan (yang akan kamu lakukan), pikirkan baik-baik mengapa bug itu terjadi. Salah satu hal terburuk yang saya lihat saat mewawancarai adalah kandidat yang mengenali kesalahan dan mencoba membuat perubahan "acak" untuk memperbaiki kesalahan.

Misalnya, Bayangkan seorang kandidat menulis fungsi yang mengembalikan angka Ketika dia menguji kodenya dengan angka '5' dia memperhatikan bahwa itu mengembalikan 0 ketika harus mengembalikan 1 Jadi, dia mengubah baris terakhir dari "return ans" menjadi " return ans + 1, ”tanpa memikirkan mengapa hal ini akan menyelesaikan masalah. Hal ini tidak hanya terlihat buruk, tetapi juga mengirimkan kandidat pada serangkaian bug dan perbaikan bug yang tak ada habisnya. Saat kamu melihat masalah pada kodemu, pikirkan baik-baik tentang mengapa kodemu gagal sebelum memperbaiki kesalahan.

BAB 12
SEPULUH KESALAHAN TERATAS YANG DILAKUKAN KANDIDAT

Berlatih di Komputer

Jika kamu berlatih untuk balapan sepeda yang serius di pegunungan, apakah kamu akan berlatih hanya dengan bersepeda di jalanan? Saya harap udaranya tidak berbeda, medannya berbeda Ya, saya yakin kamu akan berlatih di pegunungan Menggunakan kompiler untuk berlatih pertanyaan wawancara adalah seperti ini - dan pada dasarnya kamu telah bersepeda di jalanan sepanjang hidupmu Singkirkan kompiler dan keluarkan pena dan kertas lama Gunakan kompiler hanya untuk memverifikasi solusimu.

Bukan Melatih Pertanyaan Perilaku

Banyak kandidat menghabiskan seluruh waktunya untuk mempersiapkan pertanyaan teknis dan mengabaikan pertanyaan perilaku. Coba tebak? Pewawancaramu juga menilai itu! Dan, tidak hanya kinerjamu pada pertanyaan perilaku mungkin bias persepsi pewawancaramu tentang kinerja teknis kamu Persiapan perilaku relatif mudah dan sepadan dengan waktumu. Melihat proyek dan posisimu.

Tidak Melakukan Wawancara Mock

Bayangkan kamu sedang mempersiapkan pidato besar Seluruh sekolahmu, atau perusahaan, atau apa pun yang akan ada di sana Masa depanmu bergantung pada ini Dan yang kamu lakukan untuk mempersiapkan adalah membacakan pidato untuk dirimu sendiri Secara Diam-diam terasa Gila, bukan? Tidak melakukan wawancara tiruan untuk mempersiapkan wawancaramu yang sebenarnya adalah seperti ini Jika kamu seorang insinyur, kamu harus mengenal insinyur lain. Ambil seorang teman dan minta dia untuk melakukan wawancara tiruan untukmu, kamu bahkan dapat membalas budi!

Mencoba Menghafal Solusi

Kualitas mengalahkan kuantitas Cobalah untuk berjuang melalui dan memecahkan pertanyaan sendiri; Jangan langsung beralih ke solusi saat kamu buntu Menghafal cara menyelesaikan masalah tertentu tidak akan banyak membantu kamu dalam wawancara Persiapan nyata adalah tentang mempelajari cara mendekati masalah baru.

Berbicara Terlalu Banyak

Saya tidak dapat memberi tahumu berapa kali saya telah mengajukan pertanyaan sederhana kepada kandidat seperti "bug apa yang paling sulit di Project Pod?", Hanya untuk membuat mereka terus mengoceh tentang hal-hal yang tidak saya

mengerti. Lima menit kemudian, ketika mereka akhirnya mengudara, saya tidak belajar apa-apa - kecuali bahwa mereka adalah komunikator yang buruk. Ketika ditanya, bagi jawaban kamu menjadi tiga bagian (Situasi/Tindakan/Tanggapan, Edisi 1/Edisi 2/Edisi 3 , dll) dan berbicara hanya untuk beberapa kalimat tentang masing-masing Jika saya ingin detail lebih lanjut, saya akan bertanya!

Berbicara Terlalu Sedikit

izinkan saya memberi tahumu sebuah rahasia: Saya tidak tahu apa yang terjadi di kepalamu. Jadi jika kamu tidak berbicara, saya tidak tahu apa yang kamu pikirkan Jika kamu tidak berbicara untuk waktu yang lama, saya akan berasumsi bahwa kamu tidak membuat kemajuan apa pun Sering-seringlah berbicara, dan mencoba untuk berbicara tentang solusi Ini menunjukkan pewawancara mu bahwa kamu sedang mengatasi masalah dan tidak terjebak Dan ini memungkinkan mereka memandu kamu ketika kamu keluar jalur , Membantumu mendapatkan jawaban lebih cepat Dan itu menunjukkan keterampilan komunikasi kamu yang luar biasa Apa yang tidak disukai?

Bergegas

Coding bukanlah perlombaan, dan juga wawancara. Luangkan waktumu dalam masalah coding, jangan terburu-buru! Terburu-buru menyebabkan kesalahan, dan menunjukkan bahwa kamu ceroboh, lakukan dengan perlahan dan metodis, sering menguji dan memikirkan masalah secara menyeluruh, kamu akan menyelesaikan masalah dalam waktu yang lebih singkat pada akhirnya, dan dengan lebih sedikit kesalahan.

Bukan Debugging

Apakah kamu pernah menulis kode dan tidak menjalankannya atau mengujinya? Saya harap tidak! Jadi mengapa melakukan itu dalam sebuah wawancara? Saat kamu selesai menulis kode dalam wawancara, "jalankan" (atau telusuri) kode untuk mengujinya atau, pada masalah yang lebih rumit, uji kode saat menulisnya.

Pengkodean Ceroboh

Tahukah kamu bahwa kamu dapat menulis kode bebas bug sambil tetap mengerjakan soal pengkodean dengan buruk? Itu benar! Kode yang digandakan, struktur data yang berantakan (misalnya, kurangnya desain berorientasi objek), dll. Buruk, buruk, buruk! Saat kamu menulis kode, bayangkan kamu menulis untuk pemeliharaan dunia nyata. Bagi kode menjadi sub-rutinitas, dan rancang struktur data untuk menautkan data yang sesuai.

Menyerah

Apakah kamu pernah mengikuti tes adaptif komputer? Ini adalah tes yang memberimu pertanyaan yang lebih sulit, semakin baik kamu melakukannya. Ambillah dari saya - itu tidak menyenangkan Terlepas dari seberapa baik kamu sebenarnya melakukannya, kamu tiba-tiba menemukan dirimu tersandung melalui masalah Astaga!

Wawancara adalah seperti ini Jika Anda berhasil melewati masalah yang mudah, Anda akan mendapatkan masalah yang lebih banyak dan lebih sulit atau, pertanyaannya mungkin baru saja dimulai dengan sulit!

Bagaimanapun juga, bergumul dalam suatu pertanyaan tidak berarti bahwa kamu melakukannya dengan buruk, jadi jangan menyerah atau berkecil hati karena kamu melakukannya dengan baik!

BAB 13
SARAN KHUSUS UNTUK WAWANCARA DESAIN PERANGKAT LUNAK

Tidak hanya pengujian master SDET, tetapi mereka juga harus menjadi pembuat kode yang hebat.Oleh karena itu, kami merekomendasikan proses persiapan berikut:

Persiapkan Masalah Pengujian Inti: Misalnya, bagaimana Anda akan menguji bola lampu? Pena? Mesin kasir? Microsoft Word? Topik Pengujian akan memberi Anda lebih banyak latar belakang tentang masalah ini.

Berlatih Pertanyaan Pengodean: Hal # 1 yang ditolak SDET adalah keterampilan pengkodean. Pastikan Anda mempersiapkan semua pertanyaan pengkodean dan algoritme yang sama yang akan didapat oleh pengembang biasa.

Praktik Menguji Pertanyaan Pengkodean: Format yang sangat populer untuk pertanyaan SDET adalah "Tulis kode untuk mengerjakan X," diikuti dengan "OK, sekarang uji" Jadi, meskipun pertanyaan tidak secara khusus menanyakan hal ini, Anda harus bertanya pada diri sendiri , "Bagaimana Anda akan menguji ini?" Ingat: masalah apa pun bisa menjadi masalah SDET!

16 Pertanyaan Wawancara Yang Sering Muncul

- Ceritakan tentang orang yang Anda pekerjakan di posisi terakhir Anda? Berapa lama mereka tinggal? Persentase apa yang berhasil?
- Menguji pengetahuan tentang turn-over, pelatihan, dan kejujuran juga (karena tidak ada yang memiliki tingkat keberhasilan 100%).
- Kata sifat apa yang akan digunakan referensi Anda untuk mendeskripsikan Anda?
- Buatlah tetap singkat dan dapat dibandingkan dengan komentar referensi yang sebenarnya untuk melihat seberapa sadar diri mereka tentang kekuatan dan kelemahan mereka?
- Apa kekuatan terbesar yang akan Anda bawa ke organisasi ini? (Klasik tapi penting)

Secara sengaja terbuka bagi mereka semua untuk menjual kemampuan mereka. Mencari hal-hal spesifik dan pencapaian masa lalu.

Hal-hal apa yang tidak suka atau tidak ingin Anda kerjakan?

- Tes kejujuran dan kesadaran diri. Juga cara yang tidak terlalu mengancam untuk bertanya tentang kelemahan. Kita semua memiliki kelemahan, apakah mereka bersedia mengambil risiko, jujur dan menjelaskan di mana mereka mungkin membutuhkan bantuan? Apakah ada persyaratan dari posisi ini yang membutuhkan tipe kepribadian atau sifat tertentu yang mungkin mereka miliki, atau mungkin tidak?

Pertanyaan "Anti-ref" (Salah satu yang TERBAIK dan paling terbuka menghabiskan waktu untuk menggali lebih dalam.)

Pertanyaan: Pikirkan seseorang yang memiliki masalah dengan Anda dalam karier Anda, seperti kita semua, yang TIDAK PERNAH Anda gunakan sebagai referensi. Beri tahu saya kata sifat (singkatnya) yang mungkin mereka gunakan untuk menggambarkan Anda dan mengapa mereka memiliki persepsi ini? Kemudian kita bisa membahas bagaimana Anda menangani situasi tersebut.

Ini adalah pintu belakang yang bagus untuk pertanyaan kelemahan dan jauh lebih efektif. Ini sangat terbuka dan sering membawa mereka untuk menghadapi peristiwa atau masalah tersembunyi. Mendapatkan poin referensi potensial yang tidak akan mereka berikan secara sukarela dan perusahaan atau lingkungan di mana mereka mungkin kesulitan. Uji kejujuran? Menguji kemampuan mereka untuk menghadapi situasi sulit? Menguji kesan mereka tentang penyelesaian masalah dan apakah misi perusahaan masih terselesaikan meskipun ada masalah pribadi. Ini juga memberi Anda hal-hal spesifik untuk ditanyakan referensi, dengan pengungkapan tersirat, yang memaksa jawaban yang lebih jujur.

P*ertanyaan: Beri tahu saya apa saja lima hal pertama yang akan Anda lakukan jika mendapatkan posisi ini?*

Menguji tingkat yang mereka pikirkan, bagaimana mereka memecahkan masalah, seberapa cepat mereka akan menggali. Berapa banyak penelitian dan penyelidikan yang akan mereka lakukan sebelum menerapkan perubahan agar peka terhadap organisasi, sejarah, dan masalah khusus perusahaan lainnya.

Pertanyaan: Prestasi apa dalam karir Anda hingga saat ini yang paling Anda banggakan?

Di tingkat apa pencapaian itu? Besar atau kecil?

Pertanyaan: Di mana Anda ingin berada dalam 3-5 tahun dalam karir Anda? Apa yang ingin Anda hasilkan?

Menunjukkan ambisi, kemampuan untuk berpikir ke depan dan merencanakan dan menguji rencana mereka terhadap tujuan perusahaan untuk orang dan posisi tersebut.

Pertanyaan: Ceritakan bagaimana Anda akan melanjutkan (menginstal sistem baru, menerapkan prosedur baru)?

Ini adalah pertanyaan khusus posisi untuk menguji pengetahuan teknis atau manajerial tertentu. Terlalu banyak orang yang tidak menanyakan pertanyaan teknis mendalam yang spesifik, karena mereka tidak mengetahuinya atau khawatir akan menyinggung perasaannya. Anda memerlukan beberapa pertanyaan pengetahuan domain yang mendalam, teknis, dan esoteris untuk membuktikan bahwa mereka memahami lanskap teknis mereka.

Pertanyaan: Menurut Anda, apa lima hal terpenting agar Anda berhasil dalam posisi ini?

Kandidat akan paling sering menunjukkan apa yang mereka yakini sebagai kekuatan mereka, yang mungkin setuju atau tidak dengan prioritas dan tujuan perusahaan Anda.

Pertanyaan: Apa saja hal yang dapat dilakukan majikan Anda secara berbeda untuk menjadi lebih sukses?

Anggur asam atau kritik membangun? Apa tingkat yang mereka pikirkan pada gagasan kecil atau besar? Jika tidak ada apa-apa selain keluhan, mereka bisa jadi tidak puas, yang tidak mengambil tindakan untuk memperbaiki situasi, atau akan berdampak negatif pada moral perusahaan.

Pertanyaan: Mengapa Anda tertarik dengan pekerjaan ini? Apa yang Anda ketahui tentang perusahaan kami?

Minat asli di sini atau hanya pekerjaan lain? Menunjukkan pengetahuan tentang perusahaan Anda Apakah mereka melakukan pekerjaan rumah di perusahaan Anda pada tingkat informasi apa yang mereka fokuskan dan anggap penting? Apakah mereka berbicara tentang jalur karier yang masuk akal di dalam perusahaan Anda?

Pertanyaan: Apa kegagalan dan frustrasi terbesar dalam karier Anda?

Menunjukkan sikap tentang kegagalan, risiko, dan tanggung jawab diri versus hanya menyalahkan orang lain dan faktor luar.

Pertanyaan: Mengapa Anda memutuskan untuk meninggalkan posisi Anda saat ini/sebelumnya?

Gali lebih dalam dengan pertanyaan lanjutan atas jawaban mereka? Apa pun yang mendorong orang ini sangat penting untuk cara mereka memandang dunia. Apa yang mereka lakukan untuk mencoba mengoreksi apa yang membuat mereka menjauh? Apakah itu di luar kendali mereka, atau mungkin proyeksi dari masalah dan kelemahan mereka sendiri?

Pertanyaan: Risiko apa yang Anda ambil di posisi terakhir Anda?

Studi menunjukkan bahwa orang yang mengambil risiko umumnya lebih sukses daripada mereka yang tidak! Diskusi tentang ini bisa sangat mengungkap. Dalam organisasi tahap awal, Anda tidak akan ingin mempekerjakan orang yang tidak terlalu menghindari risiko, karena mereka mungkin melompat pada kesempatan baru pertama setelah mempelajari bagaimana naik turunnya keadaan.

Menurut pendapat saya, ini adalah beberapa pertanyaan wawancara terbaik dan paling terbuka. Lebih banyak lagi harus ditambahkan dengan pertanyaan spesifik pekerjaan atau industri dan ini dapat dilakukan dengan wawancara empat puluh lima menit. Gunakan pertanyaan kecil untuk orang-orang dengan posisi dan proses yang sama sehingga Anda dapat membandingkan apel dengan apel dan memiliki beberapa konsistensi.

Saya juga selalu mencatat jumlah karyawan dan/atau pendapatan setiap perusahaan di resume mereka, karena keterampilan seringkali disesuaikan dengan ukuran perusahaan. Seseorang di perusahaan kecil pasti memiliki lebih banyak ruang lingkup dan tanggung jawab untuk suatu tugas daripada di perusahaan besar, di mana tugas tersebut lebih mungkin dibagi di antara beberapa spesialis. Klaim sukses perlu dipahami dalam konteks kontribusi aktual. ("Setiap kesuksesan memiliki banyak ibu/ayah & anak, tetapi kegagalan adalah yatim piatu".)

Selalu pahami persyaratan gaji dan jarak perjalanan mereka jika mereka adalah kandidat yang kuat, karena ini dapat sangat bervariasi menurut kandidat dan menjadi faktor utama yang mempengaruhi kemampuan Anda untuk menutup kesepakatan.

Ingat gaji lebih sering merupakan fungsi ego dan latar belakang daripada nilai atau efektivitas. Permintaan gaji yang tinggi tidak membuat kandidat yang lebih baik, meskipun kandidat terbaik tahu nilai mereka di pasar. Tidak ada keputusan yang memiliki dampak lebih dari keputusan perekrutan, karena mereka akan membuat banyak keputusan untuk Anda dan organisasi sejak saat itu. Diperkirakan bahwa biaya keputusan perekrutan yang buruk bisa mencapai tiga kali lipat gaji di posisi senior.

Jadi luangkan waktu Anda dan lakukan dengan benar, jangan pernah berkompromi karena keterbatasan waktu, tetapi dapatkan solusi sementara sampai Anda memiliki kandidat terbaik yang tersedia.

BAB 14
PERTANYAAN DAN JAWABAN WAWANCARA I

Mengapa Java

Seperti semua bahasa komputer lainnya, elemen Java tidak ada secara terpisah. Sebaliknya, mereka bekerja sama untuk membentuk bahasa secara keseluruhan. Namun, keterkaitan ini dapat mempersulit untuk mendeskripsikan satu aspek java tanpa melibatkan beberapa aspek lainnya. Seringkali diskusi tentang satu fitur menyiratkan pengetahuan sebelumnya tentang fitur lainnya.

Oleh karena itu, buku ini menyajikan gambaran singkat tentang beberapa fitur utama Java. Materi yang dijelaskan di sini akan memberimu pijakan yang memungkinkan Anda untuk menulis dan memahami program sederhana & umum. Sebagian besar topik yang dibahas akan diperiksa secara lebih rinci pada bab-bab selanjutnya.

Seperti yang kita ketahui dalam beberapa tahun terakhir mendokumentasikan fakta berikut: Web telah secara tidak dapat dibatalkan menyusun kembali wajah komputasi dan pemrogram yang tidak ingin menguasai lingkungannya akan tertinggal. Sebelumnya adalah pernyataan yang kuat. Itu juga benar.

Semakin banyak, aplikasi harus terhubung ke Web. Tidak lagi menjadi masalah apa aplikasinya, akses Web yang hampir universal menyeret, mendorong, dan membujuk pemrogram untuk membuat program untuk dunia online, dan Java adalah bahasa yang akan digunakan banyak orang untuk melakukannya.

Terus terang, kefasihan di Java bukan lagi pilihan bagi programmer profesional, ini adalah persyaratan. Buku ini akan membantumu mendapatkannya.

Selain sebagai bahasa unggulan di Internet, Java juga penting karena alasan lain: Java telah mengubah arah perkembangan bahasa komputer. Banyak fitur yang pertama kali diarusutamakan oleh Java kini menemukan jalannya ke bahasa lain.

Misalnya, bahasa C # baru sangat dipengaruhi oleh Java. Pengetahuan tentang Java membuka pintu ke inovasi terbaru dalam pemrograman. Secara langsung, Java adalah salah satu bahasa komputer terpenting di dunia.

Cerita di Baliknya:

Pada saat detail java sedang dikerjakan, faktor kedua, dan yang pada akhirnya lebih penting, muncul yang akan memainkan peran penting di masa depan Java.

Kekuatan kedua ini, tentu saja, adalah World Wide Web. Seandainya Web tidak terbentuk pada waktu yang sama dengan Java diimplementasikan, Java mungkin tetap menjadi bahasa yang berguna tetapi tidak jelas untuk pemrograman elektronik konsumen.

Namun, dengan munculnya World Wide Web, Java didorong ke garis depan desain bahasa komputer, karena Web juga menuntut program portabel.

Sebagian besar programmer belajar di awal karir mereka bahwa program portabel sama sulitnya dengan yang diinginkan.

Meskipun pencarian cara untuk membuat program yang efisien dan portabel (tidak bergantung platform) hampir setua disiplin pemrograman itu sendiri, ia telah mengambil kursi belakang dari masalah lain yang lebih mendesak.

Pada tahun 1993, menjadi jelas bagi anggota tim desain Java bahwa masalah portabilitas yang sering ditemui saat membuat kode untuk pengontrol tertanam juga ditemukan saat mencoba membuat kode untuk Internet.

Faktanya, masalah yang sama yang awalnya dirancang untuk diselesaikan oleh Java dalam skala kecil juga dapat diterapkan ke Internet dalam skala besar. Realisasi ini menyebabkan fokus Java untuk beralih dari elektronik konsumen ke pemrograman Internet. Jadi, sementara keinginan untuk bahasa pemrograman netral arsitektur memberikan percikan awal, Internet pada akhirnya menyebabkan kesuksesan skala besar Java.

Seperti yang disebutkan sebelumnya, Java mendapatkan banyak karakternya dari C dan C++. Ini dengan sengaja. Para desainer Java tahu bahwa menggunakan sintaks C yang sudah dikenal dan menggemakan fitur berorientasi objek dari C++ akan membuat bahasa mereka menarik bagi banyak pemrogram C/C++ berpengalaman.

Selain kemiripan permukaan, Java memiliki beberapa atribut lain yang membantu membuat C dan C++ berhasil. Pertama, Java dirancang, diuji, dan disempurnakan oleh programmer yang bekerja secara nyata. Ini adalah bahasa yang didasarkan pada kebutuhan dan pengalaman orang-orang yang merancangnya.

Jadi, Java juga merupakan bahasa pemrogram. Kedua, Java adalah kohesif dan konsisten secara logis. Ketiga, kecuali untuk batasan yang diberlakukan oleh lingkungan Internet, Java memberimu, programmer, dan kendali penuh. Jika kamumemprogram dengan baik, programmu mencerminkannya.

Jika kamu memprogram dengan buruk, programmu juga mencerminkan hal itu. Dengan kata lain, Java bukanlah bahasa dengan roda pelatihan. Ini adalah bahasa untuk programmer profesional.

Pengantar

Java adalah bahasa pemrograman berorientasi objek yang kuat yang dikembangkan oleh Sun Microsystems Inc. pada tahun 1991. Java dikembangkan untuk perangkat elektronik konsumen tetapi kemudian dialihkan ke Internet. Sekarang Java telah menjadi bahasa pemrograman yang banyak digunakan untuk Internet. Java adalah bahasa platform netral (Mesin Independen). Program yang dikembangkan oleh Java dapat berjalan di perangkat keras apa pun atau di sistem operasi apa pun di dunia ini.

Ketika kronik bahasa komputer ditulis, berikut ini yang akan dikatakan: B mengarah ke C, C berevolusi menjadi C++, dan C++ mengatur panggung untuk Java. Memahami Java berarti memahami alasan yang mendorong penciptaannya, kekuatan yang membentuknya, dan inheritance yang diwarisi.

Seperti bahasa komputer sukses sebelumnya, Java adalah perpaduan elemen terbaik dari inheritancenya yang kaya dikombinasikan dengan konsep inovatif yang dibutuhkan oleh lingkungannya yang unik.

Sementara bab-bab selanjutnya dari buku ini menjelaskan aspek-aspek praktis Java — termasuk sintaksis, perpustakaan, dan aplikasinya — dalam bab ini, kamu akan mempelajari bagaimana dan mengapa Java muncul, dan apa yang membuatnya begitu penting.

Meskipun Java telah menjadi tidak terpisahkan terkait dengan lingkungan online Internet, penting untuk diingat bahwa Java adalah bahasa pemrograman yang pertama dan terutama. Inovasi dan pengembangan bahasa komputer terjadi karena dua alasan mendasar:

- Untuk beradaptasi dengan perubahan lingkungan dan penggunaan.
- Untuk menerapkan perbaikan dan peningkatan dalam seni pemrograman.

Seperti yang akan Anda lihat, pembuatan Java didorong oleh kedua elemen dalam ukuran yang hampir sama.

Bahasa Sebelum Java

Pertama-tama bahasa BCPL (Basic Combined Programming Language) dikembangkan oleh Martin Richards yang mempengaruhi bahasa B yang dikembangkan oleh Ken Thompson setelah itu C dikembangkan oleh Dennis Ritchie pada tahun 1972 di AT&T Bell Laboratories, USA pada sistem Operasi Unix .

Setelah beberapa tahun C++ dikembangkan oleh Bjarne Stroustroup pada awal 1980-an karena C tidak dapat menangani program yang besar. Nama awal C++ adalah “C with Classes” tetapi diganti namanya menjadi “C++” pada tahun 1983

Java terkait dengan C++, yang merupakan turunan langsung dari C. Sebagian besar karakter Java diwarisi dari dua bahasa ini. Dari C, Java mendapatkan sintaksnya. Banyak fitur berorientasi objek Java dipengaruhi oleh C++. Faktanya, beberapa karakteristik penentu Java berasal dari — atau merupakan respons terhadap — pendahulunya.

Selain itu, pembuatan Java sangat mengakar dalam proses penyempurnaan dan adaptasi yang telah terjadi dalam bahasa pemrograman komputer selama tiga dekade terakhir. Untuk alasan ini, bagian ini mengulas urutan peristiwa dan kekuatan yang mengarah ke Java. Seperti yang akan kamu lihat, setiap inovasi dalam desain bahasa didorong oleh kebutuhan untuk memecahkan masalah mendasar yang tidak dapat diselesaikan oleh bahasa sebelumnya. Java tidak terkecuali.

Sejarah java & Penciptaan Java

Pada dasarnya Java dikembangkan oleh James Gosling, Patrick Naughton, ChrisWarth, Ed Frank, dan Mike Sheridan di Sun Microsystems, Inc. pada tahun 1991. Nama awal dari bahasa ini adalah "Oak" tetapi diganti namanya pada tahun 1995 sebagai "Java". Butuh waktu 18 bulan untuk mengembangkan versi kerja pertama. Bahasa ini awalnya disebut "Oak" tetapi diganti namanya menjadi "Java" pada tahun 1995.

Antara implementasi awal Oak pada musim gugur 1992 dan pengumuman publik Java pada musim semi 1995, lebih banyak orang berkontribusi pada desain dan evolusi bahasa. Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin, dan Tim Lindholm adalah kontributor utama dalam pematangan prototipe asli.

Agak mengherankan, pendorong asli untuk Java bukanlah Internet! Sebaliknya, motivasi utamanya adalah kebutuhan akan bahasa platform-independent (yaitu, arsitektur-netral) yang dapat digunakan untuk membuat perangkat lunak yang akan disematkan di berbagai perangkat elektronik konsumen, seperti oven microwave dan kendali jarak jauh.

Seperti yang mungkin bisa Anda tebak, banyak jenis CPU yang digunakan sebagai pengontrol. Masalah dengan C dan C++ (dan sebagian besar bahasa lainnya) adalah bahwa keduanya dirancang untuk dikompilasi untuk target tertentu. Meskipun dimungkinkan untuk mengompilasi program C++ untuk hampir semua jenis CPU, untuk melakukannya diperlukan kompiler C++ lengkap yang ditargetkan untuk CPU tersebut.

Masalahnya adalah kompiler itu mahal dan memakan waktu untuk membuatnya. Diperlukan solusi yang lebih mudah dan lebih hemat biaya. Dalam upaya untuk menemukan solusi seperti itu, Gosling dan yang lainnya mulai bekerja pada bahasa portabel, platform-independen yang dapat digunakan untuk menghasilkan kode yang akan berjalan pada berbagai CPU di bawah lingkungan yang berbeda. Upaya ini akhirnya mengarah pada penciptaan Java.

Panggung Diatur untuk Java

Pada akhir 1980-an dan awal 1990-an, pemrograman berorientasi objek menggunakan C++ mulai diterapkan. Memang, untuk sesaat tampaknya para pemrogram akhirnya menemukan bahasa yang sempurna. Karena C++ memadukan efisiensi tinggi dan elemen gaya C dengan paradigma berorientasi objek, itu adalah bahasa yang dapat digunakan untuk membuat berbagai macam program.

Namun, seperti di masa lalu, kekuatan yang muncul, sekali lagi, akan mendorong evolusi bahasa komputer ke depan. Dalam beberapa tahun, World Wide Web dan Internet akan mencapai masa kritis. Peristiwa ini akan memicu revolusi lain dalam pemrograman.

Kebutuhan akan Java

Java dikembangkan karena kebutuhan akan bahasa platform netral yang dapat digunakan untuk membuat perangkat lunak yang akan disematkan di berbagai perangkat elektronik konsumen, seperti oven microwave dan kendali jarak jauh.

Program yang ditulis dalam C dan C++ dikompilasi untuk perangkat keras dan perangkat lunak tertentu dan program itu tidak akan berjalan pada perangkat keras atau perangkat lunak lain. Jadi kita membutuhkan kompiler C/C++ satu untuk setiap jenis perangkat keras untuk mengkompilasi satu program. Tetapi kompiler mahal dan memakan waktu untuk membuatnya. Jadi ada kebutuhan akan bahasa platform netral. Sehingga program yang dikompilasi dari kompilator itu dapat berjalan di perangkat keras apa pun. Kebutuhan ini menyebabkan terciptanya Java.

Perpustakaan Kelas Java

Program Java terdiri dari bagian-bagian yang disebut kelas. Kelas terdiri dari bagian-bagian yang disebut metode yang melakukan tugas dan mengembalikan informasi saat mereka menyelesaikan tugasnya. Kamu dapat memprogram setiap bagian yang mungkin kamu perlukan untuk membentuk program Java. Namun, sebagian besar programmer Java memanfaatkan koleksi kelas yang ada di perpustakaan kelas Java. Pustaka kelas juga dikenal sebagai Java API (Antarmuka Pemrograman Aplikasi).

Jadi, sebenarnya ada dua bagian untuk mempelajari "dunia" Java. Yang pertama adalah mempelajari bahasa Java itu sendiri sehingga kamu dapat memprogram kelasmu sendiri; yang kedua adalah mempelajari cara menggunakan kelas di perpustakaan kelas Java yang ekstensif.

Di sepanjang buku ini, kami membahas banyak kelas perpustakaan. Perpustakaan kelas disediakan terutama oleh vendor kompilator, tetapi banyak perpustakaan kelas disediakan oleh vendor perangkat lunak independen (ISV). Selain itu, banyak perpustakaan kelas yang tersedia dari Internet dan World Wide Web sebagai freeware atau shareware. Kamu dapat mengunduh produk perlengkapan gratis dan menggunakannya secara gratis dengan tunduk pada batasan apa pun yang ditentukan oleh pemilik hak cipta.

Kamu juga dapat mendownload produk shareware secara gratis, sehingga kamu dapat mencoba software tersebut. Produk Shareware sering kali gratis untuk penggunaan pribadi. Namun, untuk produk shareware yang kamu gunakan secara teratur atau digunakan untuk tujuan komersial, kamu diharapkan membayar biaya yang ditentukan oleh pemilik hak cipta.

Banyak produk freeware dan shareware juga open source. Kode sumber untuk produk sumber terbuka tersedia secara gratis di Internet, yang memungkinkan Anda untuk belajar dari kode sumber, memvalidasi bahwa kode tersebut memenuhi tujuan yang dinyatakan dan bahkan memodifikasi kodenya. Seringkali, produk sumber terbuka mengharuskanmu memublikasikan penyempurnaan apa pun yang kamu buat

sehingga komunitas sumber terbuka dapat terus mengembangkan produk tersebut. Salah satu contoh produk open-source yang populer adalah sistem operasi Linux.

Bahasa Tingkat Tinggi Lainnya

Ratusan bahasa tingkat tinggi telah dikembangkan, tetapi hanya sedikit yang diterima secara luas. FORTRAN (FORMULA TRANSLATOR) dikembangkan oleh IBM Corporation antara tahun 1954 dan 1957 untuk digunakan dalam aplikasi ilmiah dan teknik yang memerlukan perhitungan matematika yang kompleks. FORTRAN masih banyak digunakan.

COBOL (Common Business Oriented Language) dikembangkan pada tahun 1959 oleh sekelompok produsen komputer dan pemerintah serta pengguna komputer industri. COBOL digunakan terutama untuk aplikasi komersial yang membutuhkan manipulasi data dalam jumlah besar secara tepat dan efisien.

Saat ini, sekitar setengah dari semua perangkat lunak bisnis masih diprogram dalam COBOL. Sekitar satu juta orang secara aktif menulis program COBOL. Pascal dirancang pada waktu yang hampir bersamaan dengan C. Ini dibuat oleh Profesor Nicklaus Wirth dan dimaksudkan untuk penggunaan akademis.

Kami membahas Pascal lebih lanjut di bagian selanjutnya. Basic dikembangkan pada tahun 1965 di Dartmouth College sebagai bahasa sederhana untuk membantu para pemula merasa nyaman dengan pemrograman. Bill Gates menerapkan Basic pada beberapa komputer pribadi awal. Saat ini, Microsoft, perusahaan yang dibuat Bill Gates, adalah organisasi pengembangan perangkat lunak terkemuka di dunia.

Lingkungan pengembangan terintegrasi Java (IDE), seperti Forte untuk Java Community Edition, NetBeans, Borland's JBuilder, Symantec's Visual Cafe dan IBM Visual Age telah membangun editor yang terintegrasi ke dalam lingkungan pemrograman.

Kami berasumsi pembaca tahu cara mengedit file. Bahasa seperti Java adalah berorientasi objek — pemrograman dalam bahasa seperti itu disebut pemrograman berorientasi objek (OOP) dan memungkinkan desainer untuk mengimplementasikan desain berorientasi objek sebagai sistem kerja. Bahasa seperti C, di sisi lain, adalah bahasa pemrograman prosedural, jadi pemrograman cenderung berorientasi pada tindakan.

Di C, unit pemrograman adalah fungsinya. Di Java, unit pemrograman adalah kelas tempat objek akhirnya dibuat (istilah keren untuk "dibuat"). Kelas Java berisi metode (yang mengimplementasikan perilaku kelas) dan atribut (yang mengimplementasikan data kelas).

Pemrogram C berkonsentrasi pada fungsi penulisan. Grup tindakan yang melakukan beberapa tugas umum dibentuk menjadi fungsi, dan fungsi dikelompokkan untuk membentuk program. Data tentu saja penting di C, tetapi pandangannya adalah bahwa data ada terutama untuk mendukung tindakan yang dilakukan fungsi. Kata kerja dalam spesifikasi sistem membantu pemrogram C menentukan sekumpulan fungsi yang diperlukan untuk mengimplementasikan sistem itu.

Pemrogram Java berkonsentrasi pada pembuatan tipe yang ditentukan pengguna mereka sendiri yang disebut kelas dan komponen. Setiap kelas berisi data dan sekumpulan fungsi yang memanipulasi data tersebut. Komponen data kelas Java disebut atribut.

Komponen fungsi kelas Java disebut metode. Sama seperti instance dari tipe bawaan seperti int disebut variabel, instance dari tipe yang ditentukan pengguna (yaitu, kelas) disebut objek. Pemrogram menggunakan tipe built-in sebagai "blok bangunan" untuk membangun tipe yang ditentukan pengguna.

Fokus di Java adalah pada kelas (di mana kita membuat objek) daripada pada fungsi. Kata benda dalam spesifikasi sistem membantu programmer Java menentukan sekumpulan kelas dari mana objek akan dibuat yang akan bekerja sama untuk mengimplementasikan sistem.

Kelas adalah untuk objek seperti cetak biru untuk rumah. Kita bisa membangun banyak rumah dari satu cetak biru, dan kita bisa membuat banyak objek dari satu kelas. Kelas juga dapat memiliki hubungan dengan kelas lain.

Misalnya, dalam desain bank yang berorientasi objek, kelas "teller bank" perlu dikaitkan dengan kelas "pelanggan". Hubungan ini disebut asosiasi.

Kita akan melihat bahwa, ketika perangkat lunak dikemas sebagai kelas, kelas-kelas ini dapat digunakan kembali dalam sistem perangkat lunak di masa mendatang. Grup kelas terkait sering dikemas sebagai komponen yang dapat digunakan kembali.

Sama seperti pialang real estat memberi tahu klien mereka bahwa tiga faktor terpenting yang memengaruhi harga real estat adalah "lokasi, lokasi, dan lokasi", banyak orang dalam komunitas perangkat lunak percaya bahwa tiga faktor terpenting yang memengaruhi masa depan pengembangan perangkat lunak adalah "Gunakan kembali, gunakan kembali, dan gunakan kembali".

Memang, dengan teknologi objek, kita dapat membangun banyak perangkat lunak yang kita perlukan dengan menggabungkan "bagian standar yang dapat dipertukarkan" yang disebut kelas. Buku ini mengajarimu cara "membuat kelas yang berharga" untuk digunakan kembali.

Setiap kelas baru yang kamu buat akan memiliki potensi untuk menjadi aset perangkat lunak yang berharga yang dapat kamu dan pemrogram lain gunakan untuk mempercepat dan meningkatkan kualitas upaya pengembangan perangkat lunak di masa mendatang — kemungkinan yang menarik.

Hubungan Java dengan C, C++, & C #

Dari C Java memperoleh sintaksnya dan dari C++ ia memperoleh fitur berorientasi objek. Ini bukan versi C++ yang ditingkatkan. Java tidak kompatibel ke atas maupun ke bawah dengan C++.

Satu hal penting yang ingin saya sampaikan kepada kamu adalah bahwa bahasa Java tidak dirancang untuk menggantikan C++ dan C #. Bahasa lain yang dikembangkan oleh Microsoft untuk mendukung .NET Framework, C # terkait erat dengan Java karena

keduanya berbagi sintaks gaya C++ dan C, mendukung pemrograman terdistribusi, dan menggunakan model objek yang sama.

Tujuan Utama Java adalah untuk mencapai:

Keamanan: Tidak ada ancaman infeksi virus saat kami menggunakan Browser Web yang kompatibel dengan Java. Juga tidak ada ancaman program jahat yang dapat mengumpulkan informasi pribadi, seperti nomor kartu kredit, saldo rekening bank dan kata sandi dari mesin lokal.

Java menyediakan firewall antara aplikasi jaringan dan komputer kita.

Portabilitas: Program Java bersifat portabel dari satu komputer ke komputer lain yang menjalankan berbagai jenis sistem operasi dan memiliki perangkat keras yang berbeda.

Java Bytecode:

Output dari kompilator Java adalah bytecode, bukan kode mesin (file ".class"). Bytecode adalah sekumpulan instruksi yang sangat dioptimalkan yang dirancang untuk dijalankan oleh sistem run-time Java, yang disebut sebagai JVM (Java Virtual Machine).

JVM adalah interpreter yang menafsirkan bytecode. Program yang dikompilasi berjalan lebih cepat tetapi Java masih menggunakan interpreter untuk mencapai portabilitas sehingga program Java berjalan sedikit lebih lambat.

Sekarang program yang dikompilasi melalui kompilator Java dapat berjalan di lingkungan apa pun tetapi JVM perlu diimplementasikan untuk setiap platform. Program Java diinterpretasikan.

Ini juga membantu membuatnya aman karena eksekusi setiap program Java berada di bawah kendali JVM.

JIT (Just In Time)

JIT adalah penerjemah yang digunakan oleh JVM untuk menerjemahkan bytecode ke dalam kode mesin yang sebenarnya. Itu tidak menerjemahkan seluruh bytecode melainkan menerjemahkan sepotong demi sepotong berdasarkan permintaan.

Jenis aplikasi yang dapat dikembangkan Java:

- Aplikasi Mandiri-

Aplikasi mandiri adalah program yang berjalan di komputer lokal kita di bawah sistem operasi komputer itu seperti program C atau C++.

- Applet-

Applet adalah program kecil yang berjalan melintasi Internet dan dijalankan oleh browser web yang Kompatibel dengan Java, seperti Internet Explorer atau Netscape Navigator, pada mesin klien.

Applet sebenarnya adalah program Java kecil, diunduh secara dinamis di seluruh jaringan. Program Applet disimpan di server web dan berjalan ke mesin klien berdasarkan permintaan dari mesin klien.

Applet tidak dapat dijalankan seperti aplikasi mandiri. Applet dapat dijalankan hanya dengan menyematkannya ke dalam halaman HTML seperti file suara atau file gambar atau klip video.

Sekarang halaman HTML yang memiliki applet yang tertanam di dalamnya dapat dijalankan setelah mengunduh halaman HTML tersebut oleh browser web di mesin lokal. Applet adalah program yang dapat bereaksi terhadap input pengguna dan dapat berubah secara dinamis. Itu tidak menjalankan animasi atau suara yang sama berulang-ulang.

Aplikasi Web-

Ini adalah program yang berjalan di Server Web. Ketika kita mengakses situs web dengan menentukan URL (Universal Resource Locator) di web browser maka web browser mengirimkan permintaan ke web server untuk situs Web tertentu. Setelah menerima permintaan ini, server menjalankan program dan program ini disebut Aplikasi Web. Kami menggunakan Java Servlets dan JSP (Java Server Pages) untuk menulis program semacam itu.

Program ini berjalan di server dan kemudian mengirimkan hasil/responnya ke klien. Halaman JSP dapat dianggap sebagai kombinasi dari HTML dan Kode Java. Server Web mengubah halaman JSP menjadi Java Servlets sebelum dieksekusi. Ketika klien meminta URL tertentu dan URL sesuai dengan halaman HTML, server web hanya mengembalikan halaman HTML ke klien, yang kemudian menampilkannya. Jika URL sesuai dengan servlet atau JSP maka dijalankan di Server dan hasil/respon dikembalikan ke klien, yang kemudian ditampilkan oleh klien.

Aplikasi Terdistribusi-

Aplikasi Java dibagi menjadi beberapa program kecil yang dapat dijalankan di mesin terpisah. Objek yang digunakan dalam program ini dapat berkomunikasi satu sama lain. Aplikasi ini dikenal sebagai Aplikasi Terdistribusi. Ini memungkinkan objek pada dua komputer berbeda untuk menjalankan prosedur dari jarak jauh. Untuk RMI ini (Remote Method Invocation) digunakan.

Karakteristik Java

Sederhana- Sintaks Java hampir mirip dengan C dan C++ sehingga seorang programmer yang terbiasa dengan C/C++ tidak harus mempelajari sintaks tersebut dari awal. Tetapi banyak fitur C/C++, yang kompleks atau mengakibatkan ambiguitas, telah dihapus di Java.

Java tidak mendukung multiple inheritance, karena konsepnya agak rumit dan dapat menyebabkan ambiguitas.

Java tidak mendukung variabel global, yang juga menyebabkan banyak bug dalam program C/C++.

Java tidak menggunakan pointer dan tidak mengizinkan aritmatika pointer, yang merupakan penyebab sebagian besar bug dalam program C/C++ karena kompleksitas yang melekat.

Java tidak mendukung kelebihan beban operator karena dapat menimbulkan kebingungan.

Tidak ada konsep nilai sampah di java. Kita harus menginisialisasi variabel sebelum digunakan.

Aman- Program Java dijalankan di dalam JVM (Java Virtual Machine) dan tidak dapat diakses ke bagian lain. Ini sangat meningkatkan keamanan. Program Java jarang mengalami hang karena fitur ini. Ini tidak seperti program C/C++, yang sering hang. Model keamanan Java memiliki tiga komponen utama:

- *Loader kelas.*
- *Pemverifikasi Bytecode.*
- *Manajer keamanan.*

Java menggunakan pemuat kelas yang berbeda untuk memuat file kelas (file yang dapat dieksekusi) dari mesin lokal dan mesin jarak jauh. Kelas yang dimuat dari mesin jarak jauh seperti kelas Applet tidak diizinkan untuk membaca atau menulis file di mesin lokal. Ini mencegah program jahat merusak sistem file lokal.

Pemverifikasi Bytecode memverifikasi bytecode segera setelah pemuat kelas menyelesaikan tugasnya. Ini memastikan bahwa bytecode adalah kode Java yang valid. Ini hampir menghilangkan kemungkinan program Java melakukan beberapa aktivitas berbahaya seperti mengakses memori di luar JVM.

Manajer Keamanan mengontrol banyak operasi penting seperti penghapusan file, pembuatan utas, dll. Operasi ini diizinkan hanya jika program Java memiliki izin yang memadai, jika tidak, Manajer Keamanan tidak mengizinkan operasi dan menghasilkan Pengecualian Keamanan.

Portable- Program Java tidak bergantung pada platform. Mereka mengikuti kebijakan tulis-sekali-lari-di mana saja. Program Java yang ditulis untuk Platform Windows dapat berjalan di platform lain (Unix, Linux, Sun Solaris, dll.) Hanya dengan menyalin bytecode (file ".class").

Tidak perlu menyalin kode sumber dan mengkompilasinya lagi seperti pada program C/C++. Fitur ini membuat Java menjadi bahasa yang ampuh. Kita dapat menjalankan bytecode di mesin manapun asalkan mesin tersebut memiliki JVM. JVM sendiri bergantung pada platform tetapi membuat platform kode Java independen. Ini sebenarnya JVM yang mengubah bytecode menjadi kode mesin dan mengeksekusinya.

Jadi kita dapat mengatakan bahwa Java adalah bahasa portabel. Satu lagi fitur yang membuat Java sangat portabel adalah tipe data primitif memiliki panjang tetap terlepas dari platformnya. Misalnya int akan selalu 4 byte di Java. Ini tidak seperti C/C++ di mana ukuran int bisa 2 byte pada beberapa mesin dan 4 byte pada mesin lain.

Berorientasi Objek- Java hampir murni bahasa berorientasi objek tetapi mendukung tipe data primitif seperti byte, short, int, long, float, double, char, boolean untuk alasan kinerja.

Kuat: - Sebagian besar program gagal salah satu dari dua alasan berikut:

- Manajemen memori.
- Kondisi luar biasa pada waktu berjalan.

Saat merancang bahasa, salah satu tujuannya adalah untuk memastikan bahwa program Java sekuat mungkin, yaitu jarang gagal. Jadi sangat penting diberikan dua faktor di atas di java.

Dalam alokasi memori Java dan de-alokasi ditangani dalam bahasa itu sendiri, yang menghilangkan banyak masalah yang disebabkan karena fitur manajemen memori dinamis di C/C++. Java juga mendukung fitur penanganan luar biasa berorientasi objek untuk menangani kondisi luar biasa, yang terjadi pada waktu proses. Hal ini memungkinkan program Java untuk memulihkan dan melanjutkan eksekusi bahkan setelah kondisi luar biasa terjadi.

Multithreaded: - Java dirancang untuk memenuhi kebutuhan dunia nyata dalam membuat program jaringan dan interaktif. Java menyediakan dukungan untuk menulis program multi-thread untuk mencapai ini. Hal ini memungkinkan programmer untuk menulis program yang dapat melakukan banyak hal secara bersamaan.

Misalnya aplikasi berbasis GUI (Graphical User Interface) mungkin mendengarkan kejadian pengguna dan mengambil tindakan yang sesuai, utas terpisah mungkin sedang mencetak dan utas terpisah mungkin mengunduh file dari beberapa mesin di seluruh jaringan, semua ini dilakukan secara bersamaan. Ini menghasilkan kinerja yang lebih baik dan pemanfaatan CPU yang lebih baik.

Dimungkinkan juga untuk menulis program multi-utas dalam bahasa lain juga tetapi itu dicapai hanya dengan menggunakan panggilan Sistem sementara dalam kasus Java itu dapat dicapai dengan menggunakan fitur-fitur bahasa itu sendiri.

Arsitektur-netral- Salah satu masalah utama yang dihadapi programmer adalah tidak adanya jaminan bahwa jika kita menulis program hari ini, program itu akan berjalan besok-bahkan pada mesin yang sama. Peningkatan sistem operasi, peningkatan prosesor, dan perubahan sumber daya sistem inti semuanya dapat digabungkan untuk membuat malfungsi program. Tetapi tujuan dari program Java adalah "menulis sekali berjalan di mana saja".

Ditafsirkan dan Kinerja Tinggi- Program Java diinterpretasikan tetapi tetap berjalan cepat dibandingkan dengan penerjemah lain.

Terdistribusi- Java dirancang untuk lingkungan terdistribusi di Internet. Java memiliki dukungan bawaan untuk berbagai protokol berbasis TCP/IP untuk tujuan ini. Faktanya, mengakses sumber daya menggunakan URL mirip dengan mengakses file di mesin lokal. Java juga memiliki fitur untuk Remote Method Invocation, yang mirip dengan Remote Procedure Calls (RPC). Ini memungkinkan objek di komputer yang berbeda untuk menjalankan prosedur dari jarak jauh. Java memiliki API bawaan untuk tujuan ini yang disebut RMI.

Dinamis- Setiap kelas Java adalah unit eksekusi terpisah. Kelas dimuat pada waktu proses hanya jika diperlukan. Mekanisme default untuk metode pengikatan di Java juga dinamis (pengikatan run-time).

Strategi Pemrograman Pembelajaran:

Kursus pemrograman sangat berbeda dari kursus lainnya. Dalam kursus pemrograman, kamu belajar dari contoh, dari latihan, dan dari kesalahan. Kamu perlu mencurahkan banyak waktu untuk menulis program, mengujinya, dan memperbaiki kesalahan.

Bagi pemrogram pemula, mempelajari Java sama seperti mempelajari bahasa pemrograman tingkat tinggi apa pun. Poin fundamentalnya adalah untuk mengembangkan keterampilan kritis dalam merumuskan solusi programatik untuk masalah nyata dan menerjemahkannya ke dalam program menggunakan pernyataan seleksi, loop, metode, dan array.

Setelah kamu memperoleh keterampilan dasar menulis program menggunakan loop, metode, dan array, Anda dapat mulai mempelajari cara mengembangkan program besar dan program GUI menggunakan pendekatan berorientasi objek.

Ketika kamu tahu bagaimana memprogram dan kamu memahami konsep pemrograman berorientasi objek, mempelajari Java menjadi masalah mempelajari Java API. Java API menetapkan kerangka kerja bagi pemrogram untuk mengembangkan aplikasi menggunakan Java.

Kamu harus menggunakan kelas dan antarmuka di API dan mengikuti konvensi dan aturan mereka untuk membuat aplikasi. Cara terbaik untuk mempelajari Java API adalah dengan meniru contoh dan melakukan latihan.

Spesifikasi Bahasa Java, API, JDK, dan IDE

Bahasa komputer memiliki aturan penggunaan yang ketat. Jika kamu tidak mengikuti aturan saat menulis program, komputer tidak akan dapat memahaminya. Spesifikasi bahasa Java dan Java API menentukan standar Java.

Spesifikasi bahasa Java adalah definisi teknis dari bahasa yang mencakup sintaks dan semantik bahasa pemrograman Java. Spesifikasi lengkap bahasa Java dapat ditemukan di java.sun.com/docs/books/jls.

Antarmuka program aplikasi (API) berisi kelas dan antarmuka yang telah ditentukan untuk mengembangkan program Java. Spesifikasi bahasa Java stabil, tetapi API masih berkembang. Di Situs Sun Java (java.sun.com), kamu dapat melihat dan mendownload Java API versi terbaru.

Java adalah bahasa yang lengkap dan kuat yang dapat digunakan dengan berbagai cara. Muncul dalam tiga edisi: Java Standard Edition (Java SE), Java Enterprise Edition (Java EE), dan Java Micro Edition (Java ME).

Java SE dapat digunakan untuk mengembangkan aplikasi atau applet yang berdiri sendiri di sisi klien. Java EE dapat digunakan untuk mengembangkan aplikasi sisi server, seperti Java servlet dan Java Server Pages. Java ME dapat digunakan untuk mengembangkan aplikasi untuk perangkat seluler, seperti telepon seluler.

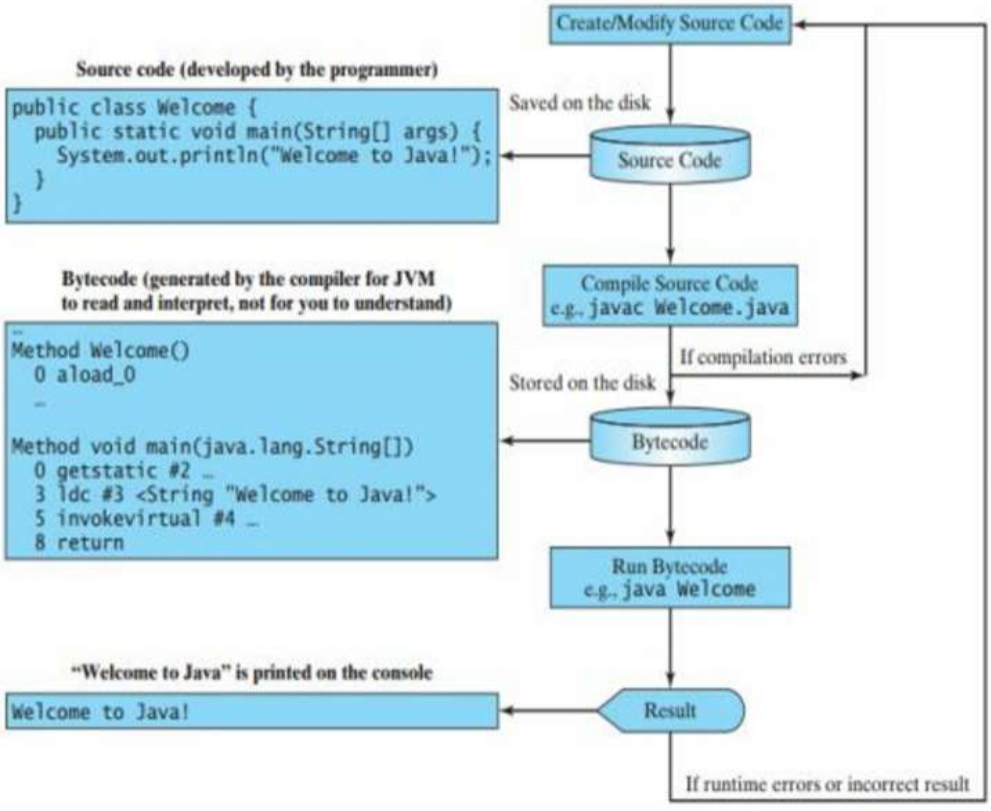
JDK terdiri dari sekumpulan program terpisah, masing-masing dipanggil dari baris perintah, untuk mengembangkan dan menguji program Java. Selain JDK, Anda

dapat menggunakan alat pengembangan Java (misalnya, Net Beans, Eclipse, dan TextPad) —perangkat lunak yang menyediakan lingkungan pengembangan terintegrasi

(IDE) untuk program Java yang berkembang pesat. Pengeditan, kompilasi, pembuatan, debugging, dan bantuan online terintegrasi dalam satu antarmuka pengguna grafis. Cukup masukkan kode sumber dalam satu jendela atau buka file yang ada di jendela, lalu klik tombol, item menu, atau tombol fungsi untuk menyusun dan menjalankan program.

Membuat, Mengompilasi, dan Menjalankan Program Java:

Kamu harus membuat programmu dan mengkompilasinya sebelum dapat dijalankan. Proses ini berulang, seperti yang ditunjukkan pada Gambar 1.1. Jika programmu memiliki kesalahan kompilasi, kamu harus memodifikasi program untuk memperbaikinya,lalu mengkompil asinya ulang. Jika programmu memiliki runtime error atau tidak menghasilkan hasil yang benar, kamu harus memodifikasi program, mengkompilasi ulang, dan menjalankannya lagi. Jendela atau buka file yang ada di jendela, lalu klik tombol, item menu, atau tombol fungsi untuk mengkompilasi dan menjalankan program.



Gambar 14.1 flowchart

Pertanyaan Bisakah kita mendefinisikan pengubah privat dan dilindungi untuk variabel dalam antarmuka? (Core java)

Jawaban Tidak.

Pertanyaan Apa kueri yang digunakan untuk menampilkan semua nama tabel di SQL Server (Query analyzer)? (JDBC)

Jawaban pilih * dari information_schema.tables

Pertanyaan Apa itu Dapat Dieksternalisasikan? (Core java)

Jawaban Externalizable adalah Antarmuka yang memperluas Antarmuka Serializable. Dan mengirimkan data ke Aliran dalam Format Terkompresi. Ini memiliki dua metode, writeExternal (ObjectOuput out) dan readExternal (ObjectInput in)

Pertanyaan Pengubah apa yang diizinkan untuk metode dalam Antarmuka? (Core java)

Jawaban Hanya pengubah publik dan abstrak yang diperbolehkan untuk metode dalam antarmuka.

Pertanyaan Apa itu variabel lokal, anggota dan kelas? (Core java)

Jawaban Variabel yang dideklarasikan dalam suatu metode adalah variabel "lokal". Variabel yang dideklarasikan dalam kelas yaitu tidak dalam metode apa pun adalah variabel "anggota" (variabel global). Variabel yang dideklarasikan dalam kelas yaitu tidak dalam metode apa pun dan didefinisikan sebagai "statis" adalah variabel kelas.

Pertanyaan Ada berapa jenis Driver JDBC dan apa saja? (JDBC)

jawaban Ada 4 tipe JDBC Drivers

Tipe 1: Driver Jembatan JDBC-ODBC

Tipe 2: Native API Partly Java Driver

Jenis 3: Driver protokol jaringan

Tipe 4: JDBC Net murni Java Driver

Pertanyaan Bisakah kita menerapkan antarmuka dalam JSP? (JSP)

Jawaban Tidak

Pertanyaan Apa perbedaan antara ServletContext dan PageContext? (JSP)

Jawaban ServletContext: Memberikan informasi tentang container PageContext: Memberikan informasi tentang Request

Pertanyaan Apa perbedaan dalam menggunakan request.getRequestDispatcher() dan context.getRequestDispatcher()? (JSP)

Jawaban request.getRequestDispatcher (jalur): Untuk membuatnya, kita perlu memberikan jalur relatif dari konteks sumber daya. GetRequestDispatcher (jalur): Untuk membuatnya, kita perlu memberikan jalur absolut sumber daya.

Pertanyaan Bagaimana cara menyampaikan informasi dari JSP ke JSP yang disertakan? (JSP)

Jawaban Menggunakan tag <% jsp: param>.

Pertanyaan Apa perbedaan antara direktif include dan jsp include? (JSP)

Jawaban<% @ include>: Digunakan untuk menyertakan sumber daya statis selama waktu terjemahan. : Digunakan untuk memasukkan konten dinamis atau konten statis selama.

Pertanyaan Apa itu variabel standar atau objek implisit?

Jawaban Untuk menyederhanakan kode dalam ekspresi JSP dan scriptlet, kita dapat menggunakan delapan variabel yang didefinisikan secara otomatis, kadang-kadang disebut objek implisit. Mereka adalah request, response, out, session, application, config, pageContext, dan page.

Pertanyaan Apa itu BDK?

Jawaban BDK, Bean Development Kit adalah alat yang memungkinkan untuk membuat, mengkonfigurasi, dan menghubungkan satu set Beans dan dapat digunakan untuk menguji Beans tanpa menulis kode.

Pertanyaan Apa itu file Jar?

Jawaban File Jar memungkinkan untuk menyebarkan secara efisien satu set kelas dan sumber daya yang terkait. Elemen-elemen dalam file jar dikompresi, yang membuat pengunduhan file Jar jauh lebih cepat daripada mengunduh beberapa file yang tidak dikompresi secara terpisah. Paket java. util. zip berisi kelas yang membaca dan menulis file jar.

Pertanyaan Jelaskan metode, rebind() dan lookup() di kelas Penamaan?

Jawaban rebind() dari kelas Penamaan (ditemukan di java. Rmi) digunakan untuk memperbarui registri RMI di mesin server. Penamaan. rebind ("AddSever", AddServerImpl); lookup() dari kelas Penamaan menerima satu argumen, URL rmi dan mengembalikan referensi ke objek bertipe AddServerImpl.

Pertanyaan apa itu UnicastRemoteObject?

Jawaban Semua objek jarak jauh harus memperluas UnicastRemoteObject, yang menyediakan fungsionalitas yang diperlukan untuk membuat objek tersedia dari mesin jarak jauh.

Pertanyaan Apa itu arsitektur RMI?

Jawaban Arsitektur RMI terdiri dari empat lapisan dan setiap lapisan melakukan fungsi tertentu:

Lapisan aplikasi - berisi definisi objek yang sebenarnya.

Lapisan proxy - terdiri dari rintisan dan kerangka.

Lapisan Referensi Jarak Jauh - mendapatkan aliran byte dari lapisan transport dan mengirimkannya ke lapisan proxy.

Lapisan transportasi - bertanggung jawab untuk menangani komunikasi mesin-ke-mesin yang sebenarnya.

Pertanyaan Langkah apa saja yang terlibat dalam mengembangkan objek RMI?

Jawaban Langkah-langkah yang terlibat dalam mengembangkan objek RMI adalah:

Tentukan antarmuka.

Menerapkan antarmuka ini.

Kompilasi antarmuka dan implementasinya dengan compiler java.

Kompilasi implementasi server dengan compiler RMI.

Jalankan registri RMI.

Jalankan aplikasinya.

Pertanyaan Apa itu RMI?

Jawaban Remote Method Invocation (RMI) memungkinkan objek java yang dieksekusi di satu mesin dan menjalankan metode objek Java untuk dieksekusi di mesin lain.

Pertanyaan Bagaimana servlet menangani beberapa permintaan secara bersamaan?

Jawaban Server memiliki beberapa utas yang tersedia untuk menangani permintaan. Ketika sebuah permintaan masuk, itu ditetapkan ke utas, yang memanggil metode layanan (misalnya: doGet(), doPost() dan service()) dari servlet. Untuk alasan ini, satu objek servlet dapat memiliki metode layanan yang dipanggil oleh banyak utas sekaligus.

Pertanyaan Apa itu rantai Servlet?

Jawaban Servlet chaining adalah teknik di mana dua atau lebih servlet dapat bekerja sama dalam melayani satu permintaan. Dalam rangkaian servlet, satu keluaran servlet disalurkan ke masukan servlet berikutnya. Proses ini berlanjut hingga servlet terakhir tercapai. Outputnya kemudian dikirim kembali ke klien.

Pertanyaan Apakah mungkin untuk memanggil servlet dengan parameter di URL?

Jawaban Ya. Anda dapat memanggil servlet dengan parameter dalam sintaksis sebagai (? Param1 = xxx || m2 = yyy).

Pertanyaan Mengapa kita harus menggunakan komunikasi antar-sergap?

Jawaban Servlets yang berjalan bersama di server yang sama berkomunikasi satu sama lain dalam beberapa cara. Tiga alasan utama untuk menggunakan komunikasi interservlet adalah:

Manipulasi servlet langsung - memungkinkan untuk mendapatkan akses ke servlet lain yang sedang dimuat dan melakukan tugas tertentu (melalui objek ServletContext)

Servlet reuse - memungkinkan servlet untuk menggunakan kembali metode publik servlet lain.

Kolaborasi servlet - perlu berkomunikasi satu sama lain dengan berbagi informasi spesifik (melalui pemanggilan metode)

Pertanyaan Apa itu penggabungan koneksi?

Jawaban Dengan servlet, membuka koneksi database merupakan hambatan utama karena kami membuat dan menghancurkan koneksi baru untuk setiap permintaan halaman dan waktu yang dibutuhkan untuk membuat koneksi akan lebih lama. Membuat kolam koneksi adalah pendekatan ideal untuk servlet yang rumit.

Dengan kumpulan koneksi, kita hanya dapat menduplikasi sumber daya yang kita butuhkan untuk menduplikasi daripada seluruh servlet. Sebuah kumpulan koneksi juga dapat secara cerdas mengelola ukuran dari kumpulan dan memastikan setiap koneksi tetap valid. Sejumlah paket kumpulan koneksi saat ini tersedia. Beberapa seperti DbConnectionBroker tersedia secara gratis dari Java Exchange Works dengan membuat objek yang membagikan koneksi dan ID koneksi berdasarkan permintaan.

Kelas ConnectionPool mempertahankan Hastable, menggunakan objek Connection sebagai kunci dan nilai Boolean sebagai nilai yang disimpan. Nilai Boolean menunjukkan apakah koneksi sedang digunakan atau tidak. Sebuah program memanggil metode getConnection() dari ConnectionPool untuk mendapatkan objek

Connection yang dapat digunakannya; itu memanggil returnConnection() untuk memberikan koneksi kembali ke kumpulan.

Pertanyaan Apakah mungkin untuk berkomunikasi dari applet ke servlet dan berapa banyak cara dan bagaimana?

Jawaban Ya, ada tiga cara untuk berkomunikasi dari applet ke servlet dan mereka adalah:

Komunikasi HTTP (berbasis teks dan berbasis objek)

Komunikasi Soket

Komunikasi RMI

Pertanyaan Apa itu cookie dan bagaimana Anda akan menggunakannya?

Jawaban Cookie adalah mekanisme yang digunakan servlet untuk meminta klien menyimpan sejumlah kecil informasi status yang terkait dengan pengguna.

Buat cookie dengan konstruktor Cookie: Cookie publik (Nama string, nilai String)

Servlet dapat mengirim cookie ke klien dengan meneruskan objek Cookie ke metode addCookie() dari HttpServletResponse: public void HttpServletResponse. addCookie (Cookie cookie)

Sebuah servlet mengambil cookie dengan memanggil metode getCookies() dari HttpServletRequest: public Cookie [] HttpServletRequest. getCookie().

Pertanyaan Apa itu Server-Side Includes (SSI)?

Jawaban Server-Side Includes memungkinkan penyematan servlet dalam halaman HTML menggunakan tag servlet khusus. Dalam banyak servlet yang mendukung servlet, halaman dapat diproses oleh server untuk menyertakan keluaran dari servlet pada titik-titik tertentu di dalam halaman HTML.

Ini dilakukan dengan menggunakan SSINCLUDE internal khusus, yang memproses tag servlet. Servlet SSINCLUDE akan dipanggil setiap kali file dengan ekstensi. ekstensi shtml diminta. Jadi file HTML yang menyertakan penyertaan sisi server harus disimpan dengan. ekstensi shtml.

Pertanyaan Apa itu pelacakan sesi dan bagaimana Anda melacak sesi pengguna di servlet?

JawabanPelacakan sesi adalah mekanisme yang digunakan servlet untuk mempertahankan status tentang serangkaian permintaan dari pengguna yang sama selama beberapa periode waktu. Metode yang digunakan untuk pelacakan sesi adalah:

Otentikasi Pengguna - terjadi ketika server web membatasi akses ke beberapa sumber dayanya hanya untuk klien yang masuk menggunakan nama pengguna dan kata sandi yang dikenali.

Bidang formulir tersembunyi - bidang ditambahkan ke formulir HTML yang tidak ditampilkan di browser klien. Ketika formulir yang berisi bidang dikirim, bidang tersebut dikirim kembali ke server.

Penulisan ulang URL - setiap URL yang diklik pengguna secara dinamis diubah atau ditulis ulang untuk memasukkan informasi tambahan. Informasi tambahan dapat dalam bentuk informasi jalur tambahan, parameter yang ditambahkan, atau beberapa perubahan URL khusus server.

Cookies - sedikit informasi yang dikirim oleh server web ke browser dan yang nantinya dapat dibaca kembali dari browser itu.

HttpSession- menempatkan batasan jumlah sesi yang bisa ada di memori. Batasan ini diatur dalam sesi. properti maxresidents.

Pertanyaan Berapa banyak cara kita melacak klien dan apa saja itu?

Jawaban Servlet API menyediakan dua cara untuk melacak status klien, yaitu:

Menggunakan pelacakan Sesi dan

Menggunakan Cookies.

Pertanyaan Apa perbedaan antara metode doPost dan doGet?

Jawaban :

Metode doGet() digunakan untuk mendapatkan informasi, sedangkan metode doPost() digunakan untuk memposting informasi.

Permintaan doGet() tidak dapat mengirimkan informasi dalam jumlah besar dan dibatasi hingga 240-255 karakter. Namun, permintaan doPost() meneruskan semua datanya, dengan panjang tidak terbatas.

Permintaan doGet() ditambahkan ke URL permintaan dalam string kueri dan ini memungkinkan pertukaran terlihat oleh klien, sedangkan permintaan doPost() lewat langsung melalui koneksi soket sebagai bagian dari badan permintaan HTTP-nya dan pertukaran tidak terlihat kepada klien.

Pertanyaan Bagaimana cara membuat dan memanggil prosedur yang tersimpan?

Jawaban Untuk membuat prosedur tersimpan:

Buat nama prosedur prosedur (tentukan parameter masuk, keluar dan keluar) BEGIN Beberapa pernyataan SQL ganda; AKHIR;

Untuk memanggil prosedur yang tersimpan: CallableStatement csmt = con. preparedCall (”{call procedure name (?,?)}”); csmt. registerOutParameter (nomor kolom, tipe data); csmt. setInt (nomor kolom, nama kolom) csmt. menjalankan();

Pertanyaan Apa Stored prosedur itu?

Jawaban Stored procedure adalah sekelompok pernyataan SQL yang membentuk unit logis dan melakukan tugas tertentu. Prosedur Tersimpan digunakan untuk merangkum serangkaian operasi atau kueri untuk dieksekusi pada database. Prosedur yang disimpan dapat dikompilasi dan dijalankan dengan parameter dan hasil yang berbeda dan dapat memiliki kombinasi parameter input/output.

Memuat driver: Untuk memuat driver, Class. Metode forName() digunakan. Kelas. forName ("sun. jdbc. odbc. JdbcOdbcDriver"); Saat driver dimuat, ia mendaftarkan dirinya dengan java. sql. DriverManager sebagai driver database yang tersedia.

Membuat koneksi dengan database: Untuk membuka koneksi ke database tertentu, DriverManager. Metode getConnection() digunakan. Koneksi con = DriverManager. getConnection (”jdbc: odbc: somedb”, “user”, “password”);

Menjalankan pernyataan SQL: Untuk menjalankan kueri SQL, java. sql. kelas pernyataan digunakan. createStatement() metode Koneksi untuk mendapatkan objek Pernyataan baru. Pernyataan stmt = con. createStatement(); Kueri yang mengembalikan data bisa dieksekusi menggunakan metode ExecutQuery() Pernyataan.

Metode ini menjalankan pernyataan dan mengembalikan java. sql. ResultSet yang merangkum data yang diambil: ResultSet rs = stmt. executeQuery ("PILIH * DARI beberapa tabel");

Memproses hasil: ResultSet mengembalikan satu baris dalam satu waktu. Metode Next() dari objek ResultSet bisa dipanggil untuk pindah ke baris berikutnya. Metode getString() dan getObject() digunakan untuk mengambil nilai kolom: while (rs. Next()) {String event = rs. getString ("event"); Jumlah objek = (Integer) rs. getObject (”count”);

Pertanyaan Apa jenis Model Driver JDBC dan menjelaskannya?

JawabanAda dua jenis Model Driver JDBC dan mereka adalah:

Model dua tingkat dan

Model tiga tingkat.

Model Two Tier: Dalam model ini, aplikasi Java berinteraksi langsung dengan database. Pengandar JDBC diperlukan untuk berkomunikasi dengan sistem manajemen basis data tertentu yang sedang diakses. Pernyataan SQL dikirim ke database dan hasilnya diberikan kepada pengguna.

Model ini disebut sebagai konfigurasi klien/server di mana pengguna adalah klien dan mesin yang memiliki database disebut sebagai server. Model tiga tingkat: Tingkat menengah diperkenalkan dalam model ini. Fungsi model ini adalah:

Kumpulan pernyataan SQL dari klien dan menyerahkannya ke database,

Menerima hasil dari database ke klien dan

Mempertahankan kendali atas akses dan pemutakhiran hal-hal di atas.

Pertanyaan Apa perbedaan antara JDBC dan ODBC?

Jawaban

OBDC untuk Microsoft dan JDBC untuk aplikasi Java.

ODBC tidak dapat langsung digunakan dengan Java karena menggunakan antarmuka C.

ODBC menggunakan pointer yang telah dihapus seluruhnya dari Java.

ODBC menggabungkan fitur sederhana dan lanjutan dan memiliki opsi kompleks untuk kueri sederhana. Tetapi JDBC dirancang untuk menjaga segala sesuatunya tetap sederhana sambil memungkinkan kemampuan lanjutan saat diperlukan.

ODBC memerlukan penginstalan manual dari manajer dan driver driver ODBC pada semua mesin klien. Driver JDBC ditulis dalam Java dan kode JDBC secara otomatis dapat diinstal, aman, dan portabel di semua platform.

JDBC API adalah antarmuka Java alami dan dibangun di atas ODBC. JDBC mempertahankan beberapa fitur dasar ODBC.

Pertanyaan Apa itu JDBC?

Jawaban JDBC adalah sekumpulan Java API untuk mengeksekusi pernyataan SQL. API ini terdiri dari sekumpulan kelas dan antarmuka untuk memungkinkan program menulis aplikasi Database Java murni.

Pertanyaan Apa itu serialisasi dan deserialisasi?

Jawaban Serialization adalah proses menulis keadaan suatu objek ke aliran byte. Deserialization adalah proses mengembalikan benda-benda tersebut.

Pertanyaan Apa perbedaan antara Reader/Writer dan InputStream/Output Stream?

Jawaban Kelas Reader/Writer berorientasi pada karakter dan kelas InputStream/OutputStream berorientasi pada byte.

Pertanyaan Apa itu aliran dan apa jenis Aliran dan kelas Aliran?

JawabanStream adalah abstraksi yang menghasilkan atau menggunakan informasi. Ada dua jenis Aliran dan mereka adalah:

Byte Streams: Menyediakan cara yang nyaman untuk menangani input dan output byte. Aliran Karakter: Menyediakan sarana yang nyaman untuk menangani masukan & keluaran karakter.

Kelas Byte Streams: Didefinisikan dengan menggunakan dua kelas abstrak, yaitu InputStream dan OutputStream.

Kelas Aliran Karakter: Didefinisikan dengan menggunakan dua kelas abstrak, yaitu Pembaca dan Penulis.

Pertanyaan Apa itu Vektor, Hashtable, LinkedList dan Enumeration?

Jawaban

Vektor: Kelas Vektor menyediakan kemampuan untuk mengimplementasikan larik objek yang dapat diperbesar.

Hashtable: Kelas Hashtable mengimplementasikan struktur data Hashtable. Sebuah indeks Hashtable dan menyimpan objek dalam kamus menggunakan kode hash sebagai kunci objek. Kode hash adalah nilai integer yang mengidentifikasi objek.

LinkedList: Menghapus atau menyisipkan elemen di tengah array dapat dilakukan menggunakan LinkedList. LinkedList menyimpan setiap objek dalam tautan terpisah sedangkan array menyimpan referensi objek di lokasi yang berurutan.

Enumerasi: Sebuah objek yang mengimplementasikan antarmuka Pencacahan menghasilkan serangkaian elemen, satu per satu. Ini memiliki dua metode, yaitu hasMoreElements() dan nextElement(). HasMoreElemnts() menguji apakah enumerasi ini memiliki lebih banyak elemen dan metode nextElement mengembalikan elemen rangkaian yang berurutan.

Pertanyaan Bagaimana elemen-elemen tata letak yang berbeda diatur?

Jawaban

FlowLayout: Elemen FlowLayout diatur dengan gaya dari atas ke bawah, dari kiri ke kanan

BorderLayout: Elemen BorderLayout diatur di perbatasan (Utara, Selatan, Timur, dan Barat) dan di tengah container.

CardLayout: Elemen CardLayout ditumpuk, di atas yang lain, seperti setumpuk kartu.

GridLayout: Elemen dari sebuah GridLayout berukuran sama dan ditata menggunakan kotak persegi.

GridBagLayout: Elemen GridBagLayout diatur menurut kisi. Namun, elemen memiliki ukuran yang berbeda dan dapat menempati lebih dari satu baris atau kolom dari kisi. Selain itu, baris dan kolom mungkin memiliki ukuran yang berbeda.

Pertanyaan Apa itu manajer tata letak dan apa saja jenis manajer tata letak yang tersedia di java AWT?

Jawaban Manajer layout adalah sebuah obyek yang digunakan untuk mengatur komponen-komponen dalam sebuah wadah. Tata letak berbeda yang tersedia adalah FlowLayout, BorderLayout, CardLayout, GridLayout, dan GridBagLayout.

Pertanyaan Apa perbedaan antara scrollbar dan scrollpane?

Jawaban A Scrollbar adalah Komponen, tetapi bukan Container sedangkan Scrollpane adalah Conatiner dan menangani kejadiannya sendiri dan melakukan penggulirannya sendiri.

Pertanyaan Apakah sumber dan pendengar itu?

Jawaban: Sumber adalah objek yang menghasilkan peristiwa. Ini terjadi ketika keadaan internal objek itu berubah dalam beberapa cara.

Pendengar: Pendengar adalah objek yang diberi tahu ketika suatu peristiwa terjadi. Ini memiliki dua persyaratan utama. Pertama, itu harus terdaftar dengan satu atau lebih sumber untuk menerima pemberitahuan tentang jenis peristiwa tertentu. Kedua, itu harus menerapkan metode untuk menerima dan memproses pemberitahuan ini.

Pertanyaan Bagaimana Anda mengatur keamanan di applet?

Jawaban menggunakan metode setSecurityManager().

Pertanyaan Bagaimana siklus hidup applet?

Jawaban init() method - Dapat dipanggil ketika applet pertama kali dimuat metode start() Dapat dipanggil setiap kali applet dijalankan. metode paint() - Dapat dipanggil saat applet diminimalkan atau dimaksimalkan. metode stop() - Dapat digunakan saat browser keluar dari halaman applet. Destroy() metode - Bisa dipanggil saat browser selesai dengan applet.

Pertanyaan Kapan Anda menggunakan basis kode di applet?

Jawaban Ketika file kelas applet tidak berada dalam direktori yang sama, basis kode digunakan.

Pertanyaan Bagaimana applet mengenali tinggi dan lebar?

Jawaban Menggunakan metode getParameters().

Pertanyaan Apakah ada variabel global di Java, yang dapat diakses oleh bagian lain dari program Anda?

Jawaban Tidak, ini bukan metode utama untuk mendefinisikan variabel. Variabel global tidak mungkin karena konsep enkapsulasi dihilangkan di sini.

Pertanyaan Apa itu daemon thread dan metode mana yang digunakan untuk membuat daemon thread?

Jawaban Daemon thread adalah thread berprioritas rendah yang berjalan sesekali di latar belakang melakukan operasi Garbage Collection untuk sistem runtime java. Metode setDaemon digunakan untuk membuat daemon thread.

Pertanyaan Kapan Anda akan menyinkronkan sebagian kode Anda?

Jawaban Ketika Anda mengharapkan kode Anda akan diakses oleh utas yang berbeda dan utas ini dapat mengubah data tertentu yang menyebabkan kerusakan data.

Pertanyaan Apa itu sinkronisasi?

Jawaban Synchronization adalah mekanisme yang memastikan bahwa hanya satu utas yang mengakses sumber daya pada satu waktu.

Pertanyaan Apa kelas dan antarmuka di java untuk membuat utas dan metode mana yang paling menguntungkan?

Jawaban Kelas Thread dan antarmuka Runnable dapat digunakan untuk membuat utas dan menggunakan antarmuka Runnable adalah metode yang paling menguntungkan untuk membuat utas karena kita tidak perlu memperluas kelas utas di sini.

Pertanyaan Apa sajakah metode untuk komunikasi antar-utas dan apa kelas di mana metode ini didefinisikan?

Jawaban Metode wait(), notify() dan notifyAll() dapat digunakan untuk komunikasi antar-utas dan metode ini berada di kelas Objek. tunggu():

Saat thread menjalankan metode panggilan untuk menunggu(), ia menyerahkan kunci objek dan masuk ke dalam status menunggu. notify() atau notifyAll(): Untuk menghapus utas dari status menunggu, beberapa utas lain harus membuat panggilan ke metode notify() atau notifyAll() pada objek yang sama.

Pertanyaan Apa itu multithreading?

Jawaban Multithreading adalah mekanisme di mana lebih dari satu thread berjalan independen satu sama lain dalam proses tersebut.

Pertanyaan Apa perbedaan antara proses dan utas?

Jawaban Proses adalah program dalam eksekusi sedangkan utas adalah jalur eksekusi terpisah dalam suatu program.

Pertanyaan Apa perbedaan antara Array dan vector?

Jawaban Array adalah sekumpulan tipe data yang berhubungan dan statis sedangkan vektor adalah array objek yang dapat berkembang dan dinamis.

Pertanyaan Apa perbedaan antara String dan String Buffer?

Objek string adalah konstanta dan tidak dapat diubah sedangkan objek StringBuffer tidak.

Kelas string mendukung string konstan sedangkan kelas StringBuffer mendukung string yang dapat tumbuh dan dimodifikasi.

Pertanyaan Bisakah Anda memiliki inner class di dalam metode dan variabel apa yang dapat Anda akses?

Jawaban Ya, kita dapat memiliki inner class di dalam metode dan variabel terakhir dapat diakses.

Pertanyaan Apa itu antarmuka yang dapat digandakan dan berapa banyak metode yang dikandungnya?

Jawaban Itu tidak memiliki metode apa pun karena ini adalah antarmuka TAGGED atau MARKER.

Pertanyaan Apa perbedaan antara Integer dan int?

Jawaban *Integer adalah kelas yang didefinisikan dalam paket java.lang, sedangkan int adalah tipe data primitif yang didefinisikan dalam bahasa Java itu sendiri. Java tidak secara otomatis mengkonversi dari satu ke yang lain.*

Integer dapat digunakan sebagai argumen untuk metode yang membutuhkan objek, sedangkan int dapat digunakan untuk kalkulasi.

Pertanyaan Apa itu inner class dan kelas anonim?

Jawaban Inner class: kelas yang didefinisikan di kelas lain, termasuk yang didefinisikan dalam metode disebut inner class. Inner class dapat memiliki aksesibilitas apa pun termasuk pribadi. Kelas anonim: Kelas anonim adalah kelas yang didefinisikan di dalam metode tanpa nama dan dibuat instance-nya serta dideklarasikan di tempat yang sama dan tidak dapat memiliki konstruktor eksplisit.

Pertanyaan Pengubah apa yang dapat digunakan dengan kelas tingkat atas?

Jawaban publik, abstrak dan final dapat digunakan untuk kelas tingkat atas.

Pertanyaan Apa yang dimaksud dengan Pra-inisialisasi Servlet?

Jawaban Ketika wadah servlet dimuat, semua servlet didefinisikan dalam file web.xml tidak diinisialisasi secara default. Tetapi kontainer menerima permintaan yang memuat servlet. Tetapi dalam beberapa kasus jika Anda ingin servlet Anda diinisialisasi saat konteks dimuat, Anda harus menggunakan konsep yang disebut pra-inisialisasi Servlet. Dalam kasus Pra-inisialisasi, servlet dimuat saat konteks dimuat. Anda dapat menentukan 1 di antara tag.

Pertanyaan Apa itu ServletContext?

Jawaban ServletContext adalah Antarmuka yang mendefinisikan sekumpulan metode yang digunakan servlet untuk berkomunikasi dengan wadah servletnya, misalnya, untuk mendapatkan tipe MIME file, mengirimkan permintaan, atau menulis ke file log. Ada satu konteks per "aplikasi web" untuk Mesin Virtual Java. ("Aplikasi web" adalah kumpulan servlet dan konten yang dipasang di bawah subset spesifik dari namespace URL server seperti/katalog dan mungkin dipasang melalui file .war.)

Pertanyaan Apa itu Servlet?

Jawaban Java Servlets adalah komponen sisi server yang menyediakan mekanisme yang kuat untuk mengembangkan aplikasi web sisi server. Sebelumnya CGI dikembangkan untuk menyediakan kemampuan sisi server ke aplikasi web. Meskipun CGI memainkan peran utama dalam ledakan Internet, masalah kinerja, skalabilitas, dan dapat digunakan kembali membuatnya menjadi solusi yang kurang optimal. Java Servlets mengubah semua itu. Dibangun dari awal menggunakan sistem tulis Sun yang dapat dijalankan di mana saja, teknologi java servlet menyediakan kerangka kerja yang sangat baik untuk pemrosesan sisi server.

Pertanyaan Di mana CardLayout digunakan?

JawabanCardLayout mengelola dua atau lebih komponen yang berbagi ruang tampilan yang sama. Ini memungkinkan Anda menggunakan satu wadah (biasanya panel) untuk menampilkan satu dari banyak kemungkinan anak komponen (seperti membalik kartu

di atas meja). Sebuah program dapat menggunakan tata letak ini untuk menampilkan komponen anak yang berbeda kepada pengguna yang berbeda.

Misalnya, antarmuka yang ditampilkan kepada administrator mungkin memiliki fungsionalitas tambahan dari antarmuka yang ditampilkan kepada pengguna biasa. Dengan tata letak kartu, program kami dapat menampilkan antarmuka yang sesuai tergantung pada jenis pengguna yang menggunakan program. Penggunaan umum tata letak kartu lainnya adalah membiarkan pengguna akhir beralih di antara tampilan yang berbeda dan memilih tampilan yang mereka sukai. Dalam hal ini, program harus menyediakan GUI bagi pengguna untuk membuat pilihan.

Pertanyaan Apa itu JFC?

Jawaban Kelas Java Foundation meliputi:

Standar AWT 1.1

Antarmuka aksesibilitas

Komponen ringan: yang merupakan komponen antarmuka pengguna yang tidak mensubkelas elemen antarmuka AWT yang ada. Mereka tidak menggunakan elemen antarmuka asli seperti yang disediakan oleh sistem windowing yang mendasarinya. Ini berarti bahwa mereka kurang membatasi daripada komponen AWT standar.

Tampilan dan nuansa Java

Dukungan untuk tampilan dan nuansa asli

Layanan seperti Java2D dan Drag and Drop

Bagaimana Anda berkomunikasi di antara Applet & Servlet?

Kita dapat menggunakan kelas java.net.URLConnection dan java.net.URL untuk membuka koneksi HTTP standar dan "terowongan" ke server web. Server kemudian meneruskan informasi ini ke servlet dengan cara biasa. Pada dasarnya, applet berpura-pura menjadi browser web, dan servlet tidak mengetahui perbedaannya. Sejauh menyangkut servlet, applet hanyalah klien HTTP lainnya.

Pertanyaan Bagaimana cara Anda berkomunikasi antara dua Applet?

Jawaban Metode paling sederhana adalah dengan menggunakan variabel statis dari kelas bersama karena hanya ada satu instance kelas dan karenanya hanya ada satu salinan variabel statisnya. Metode yang sedikit lebih andal bergantung pada fakta bahwa semua applet pada halaman tertentu berbagi AppletContext yang sama. Kami mendapatkan konteks applet ini sebagai berikut:

 AppletContext ac = getAppletContext();

AppletContext menyediakan applet dengan metode seperti getApplet (name), getApplets(), getAudioClip, getImage, showDocument dan showStatus().

Pertanyaan Kapan metode pembaruan dipanggil?

Jawaban Setiap kali layar perlu digambar ulang (misalnya, setelah pembuatan, pengubahan ukuran, validasi) metode pembaruan dipanggil. Secara default, metode update membersihkan layar dan kemudian memanggil metode paint, yang biasanya berisi semua kode gambar.

Pertanyaan Apa urutan pemanggilan metode di Applet?

Jawaban

*public void init():*Metode inisialisasi dipanggil sekali oleh browser.

*public void start():*Metode dipanggil setelah init() dan berisi kode untuk memulai pemrosesan. Jika pengguna meninggalkan halaman dan kembali tanpa menghentikan sesi browser saat ini, metode start() dipanggil tanpa didahului oleh init().

*public void stop():*Menghentikan semua pemrosesan yang dimulai oleh start(). Selesai jika pengguna keluar dari halaman.

*public void destroy():*Dipanggil jika sesi browser saat ini dihentikan. Membebaskan semua sumber daya yang digunakan oleh applet.

Pertanyaan Apa perbedaan antara notify() dan notifyAll()?

Jawaban notify() digunakan untuk membuka blokir satu thread yang menunggu; notifyAll() digunakan untuk membuka blokir semuanya. Menggunakan notify() lebih disukai (untuk efisiensi) ketika hanya satu utas yang diblokir yang dapat memanfaatkan perubahan (misalnya, saat membebaskan buffer kembali ke kumpulan). notifyAll() diperlukan (untuk kebenaran) jika beberapa utas harus dilanjutkan (misalnya, saat melepaskan kunci "penulis" pada file mungkin mengizinkan semua "pembaca" untuk melanjutkan).

Pertanyaan Apa yang dimaksud dengan pembagian waktu?

Jawaban Ini adalah metode penjadwalan tugas. Dengan pembagian waktu, atau "Sistem Round-Robin", beberapa proses dijalankan secara berurutan hingga selesai. Setiap tugas yang dapat dieksekusi diberi kuantum waktu tetap yang disebut potongan waktu untuk dieksekusi.

Pertanyaan Bagaimana sinkronisasi utas terjadi di dalam monitor?

Jawaban JVM menggunakan kunci dalam hubungannya dengan monitor. Monitor pada dasarnya adalah penjaga yang mengawasi urutan kode, memastikan hanya satu utas pada satu waktu yang menjalankan kode. Setiap monitor dikaitkan dengan referensi objek. Ketika sebuah utas tiba di instruksi pertama dalam sebuah blok kode itu harus mendapatkan kunci pada objek yang direferensikan. Utas tidak diizinkan untuk mengeksekusi kode sampai ia mendapatkan kunci. Setelah mendapatkan kunci, utas memasuki blok kode yang dilindungi. Ketika utas meninggalkan blok, tidak peduli bagaimana thread meninggalkan blok, itu melepaskan kunci pada objek terkait.

Pertanyaan Apa itu Model 1?

Jawaban Menggunakan teknologi JSP saja untuk mengembangkan halaman Web. Istilah tersebut digunakan dalam spesifikasi JSP sebelumnya. Arsitektur Model 1 cocok untuk aplikasi yang memiliki aliran halaman yang sangat sederhana, memiliki sedikit kebutuhan untuk kontrol keamanan terpusat atau pencatatan, dan sedikit berubah seiring waktu. Aplikasi Model 1 seringkali dapat direfraktorisasi menjadi Model 2 saat persyaratan aplikasi berubah.

Pertanyaan Apa itu Model 2?

Jawaban Menggunakan JSP dan Servlet bersama-sama untuk mengembangkan halaman Web. Aplikasi model 2 lebih mudah dipelihara dan diperluas, karena tampilan tidak merujuk satu sama lain secara langsung.

Pertanyaan Apa yang akan menjadi nilai default dari semua elemen array yang didefinisikan sebagai variabel instan?

Jawaban Jika array adalah array tipe primitif, maka semua elemen dari array akan diinisialisasi ke nilai default yang sesuai dengan tipe primitif tersebut. misalnya Semua elemen array int akan diinisialisasi ke 0, sedangkan tipe boolean akan diinisialisasi ke false. Sedangkan jika array adalah array referensi (jenis apapun), semua elemen akan diinisialisasi menjadi null.

Pertanyaan Ketika thread memblokir di I/O, status apa yang dimasuki?

Jawaban Sebuah thread memasuki status menunggu ketika ia memblokir di I/O.

Pertanyaan Mengapa utas diblokir pada I/O?

Jawaban Threads memblokir pada i/o (yang memasuki status menunggu) sehingga thread lain dapat dieksekusi saat Operasi i/o dilakukan.

Pertanyaan Apa perbedaan antara JSP dan JSF?

Jawaban JSP hanya menyediakan Halaman yang mungkin berisi markup, kode Java tertanam, dan "tag" yang merangkum logika/html yang lebih rumit.

JSF dapat menggunakan JSP sebagai templatnya, tetapi menyediakan lebih banyak lagi. Ini termasuk validasi, model komponen yang kaya dan siklus hidup, EL yang lebih canggih, pemisahan data, penanganan navigasi, teknologi tampilan yang berbeda (bukan JSP), kemampuan untuk menyediakan fitur yang lebih canggih seperti AJAX, dll.

Pertanyaan Apa itu JSF?

JawabanJavaServer Faces (JSF) adalah kerangka kerja Java standar baru untuk membangun aplikasi Web. Ini menyederhanakan pengembangan dengan menyediakan pendekatan yang berpusat pada komponen untuk mengembangkan antarmuka pengguna Web Java. JavaServer Faces juga menarik bagi beragam audiens pengembang Java/Web. "Pengembang korporat" dan desainer Web akan menemukan bahwa pengembangan JSF dapat sesederhana menyeret dan melepaskan komponen antarmuka pengguna (UI) ke halaman, sementara "pengembang sistem" akan menemukan bahwa JSF API yang kaya dan kuat menawarkan mereka kekuatan dan pemrograman yang tak tertandingi fleksibilitas.

JSF juga memastikan bahwa aplikasi dirancang dengan baik dengan pemeliharaan yang lebih baik dengan mengintegrasikan pola desain Model-View-Controller (MVC) yang mapan ke dalam arsitekturnya. Terakhir, karena JSF adalah standar Java yang dikembangkan melalui Java Community Process (JCP), vendor alat pengembangan diberdayakan sepenuhnya untuk menyediakan lingkungan pengembangan yang mudah digunakan, visual, dan produktif untuk JavaServer Faces.

Pertanyaan Apa perbedaan antara superclass dan subclass?

Jawaban Kelas super adalah kelas yang diwarisi sedangkan sub kelas adalah kelas yang melakukan peinheritance.

Pertanyaan Apa perbedaan antara overloading dan overriding?

Jawaban

Pada overloading terdapat hubungan antara metode yang tersedia pada kelas yang sama sedangkan pada overriding terdapat hubungan antara metode superclass dan metode subclass.

Overloading tidak memblokir peinheritance dari superclass sedangkan overriding memblokir peinheritance dari superclass.

Dalam overloading, metode terpisah berbagi nama yang sama sedangkan pada overriding, metode subclass menggantikan superclass.

Overloading harus memiliki tanda tangan metode yang berbeda sedangkan penggantian harus memiliki tanda tangan yang sama.

Pertanyaan Apa itu overloading method dan method overriding?

Jawaban Metode Overloading: Ketika sebuah metode di kelas yang memiliki nama metode yang sama dengan argumen yang berbeda dikatakan metode overloading.Metode Overriding: Ketika sebuah metode di kelas yang memiliki nama metode yang sama dengan argumen yang sama dikatakan metode menimpa.

Pertanyaan Apa metode finalize()?

Jawaban Metode finalize() digunakan tepat sebelum objek dimusnahkan dan bisa dipanggil sebelum Garbage Collection.

Pertanyaan Apa itu karakter Unicode?

Jawaban Unicode digunakan untuk representasi internal karakter dan string dan menggunakan 16 bit untuk mewakili satu sama lain.

Pertanyaan Apa itu final, finalize() dan akhirnya?

Jawaban

final: keyword final dapat digunakan untuk kelas, metode dan variabel. Kelas terakhir tidak dapat disubkelas dan mencegah pemrogram lain dari subkelas kelas aman untuk memanggil metode tidak aman. Metode terakhir tidak dapat diganti. Variabel akhir tidak dapat berubah dari nilai yang diinisialisasi.

finalize(): finalize()Metode ini digunakan tepat sebelum objek dimusnahkan dan bisa dipanggil sebelum Garbage Collection.

finally: akhirnya, keyword yang digunakan dalam exception handling, membuat blok kode yang akan dieksekusi setelah blok coba/tangkap selesai dan sebelum kode yang mengikuti blok coba/tangkap. Finally block akan dieksekusi apakah pengecualian dilemparkan atau tidak. Misalnya, jika metode membuka file saat keluar, Anda tidak ingin kode yang menutup file dilewati oleh mekanisme exception handling. Keyword terakhir ini dirancang untuk mengatasi kemungkinan ini.

Pertanyaan Apa sajakah tipe pengubah akses di Java?

Jawaban

Publik: Apa pun yang dinyatakan sebagai publik dapat diakses dari mana saja.

Pribadi: Apa pun yang dideklarasikan sebagai pribadi tidak dapat dilihat di luar kelasnya.

Protected: Segala sesuatu yang dideklarasikan sebagai dilindungi dapat diakses oleh kelas-inner class paket dan subkelas yang sama di paket-paket lain.

Default Modifier: Hanya dapat diakses ke inner class paket yang sama.

Pertanyaan Berapa banyak cara argumen diteruskan ke subrutin?

Jawaban Sebuah argumen dapat disampaikan dengan dua cara. Mereka Pass by Value dan Passing by Reference.

Passing by value: Metode ini menyalin nilai argumen ke dalam parameter formal subrutin.

Passing by Reference: Dalam metode ini, referensi ke argumen (bukan nilai argumen) diteruskan ke parameter.

Pertanyaan Apa gunanya "bin" dan "lib" di JDK?

Jawaban "bin" berisi semua alat seperti javac, appletviewer, alat awt, dll. Sedangkan "lib" berisi API dan semua paket.

Pertanyaan Apa saja jenis Tingkat Isolasi Transaksi?

Jawaban Level isolasi menggambarkan sejauh mana data yang diperbarui dapat dilihat oleh transaksi lain. Ini penting ketika dua transaksi mencoba membaca baris yang sama dari sebuah tabel. Bayangkan dua transaksi: A dan B. Di sini tiga jenis ketidakkonsistenan dapat terjadi:

Dirty-read: A telah mengubah baris, tetapi belum melakukan perubahan. B membaca data yang tidak terikat tetapi pandangannya tentang datanya mungkin salah jika A memutar kembali perubahannya dan memperbarui perubahannya sendiri ke database.

*Non Repeatable Read: B melakukan pembacaan, tetapi A mengubah atau menghapus datanya nanti. Jika B membaca baris yang sama lagi, dia akan mendapatkan data yang berbeda.

*Phantom: A melakukan kueri pada sekumpulan baris untuk melakukan operasi. B memodifikasi tabel sedemikian rupa sehingga kueri A akan memberikan hasil yang berbeda. Tabel mungkin tidak konsisten.

Pertanyaan Apa yang Anda maksud dengan multiple inheritance dalam C++?

Jawaban Multiple inheritance adalah fitur dalam C++ yang dengannya satu kelas dapat terdiri dari tipe yang berbeda. Katakanlah class teachingAssistant diwarisi dari dua kelas katakanlah guru dan Murid.

Pertanyaan Apa yang Anda maksud dengan metode virtual? Metode Virtual Answer digunakan untuk menggunakan fitur polimorfisme di C++. Katakanlah kelas A diwarisi dari kelas B. Jika kita mendeklarasikan say function f() sebagai virtual di kelas B dan menimpa fungsi yang sama di kelas A maka pada saat runtime metode yang sesuai dari kelas tersebut akan dipanggil tergantung pada jenis objeknya. Pertanyaan Apa yang Anda maksud dengan metode statis?

Jawaban Dengan menggunakan metode statis tidak perlu membuat objek kelas itu untuk menggunakan metode itu. Kita bisa langsung memanggil metode itu di kelas itu. Misalnya, kelas A memiliki fungsi statis f(), maka kita dapat memanggil fungsi f() sebagai A.f(). Tidak perlu membuat objek kelas A.

Pertanyaan Apa keuntungan dari OOPL?

Jawaban Bahasa pemrograman berorientasi objek secara langsung mewakili objek kehidupan nyata. Ciri-ciri OOPL sebagai zat penghilang rasa sakit, polimorfisme, enkapsulasi membuatnya bertenaga.

Pertanyaan Apakah ada filter file standar untuk JFileChooser?

Jawaban Selain menerima semua filter, sampai Java 6, tidak ada filter yang telah ditetapkan. Java 6 memperkenalkan FileNameExtensionFilter, memungkinkan Anda untuk menentukan satu atau lebih tipe file untuk dipilih pengguna.

```
Filter FileFilter = FileNameExtensionFilter baru ("file JPEG", "jpg", "jpeg");
JFileChooser fileChooser =…;
fileChooser.addChoosableFileFilter (filter);
```

Pertanyaan Mengapa applet saya tidak dapat membaca atau menulis ke file?

Jawaban Applet dijalankan di bawah kendali browser web. Netscape dan Internet Explorer memberlakukan batasan keamanan, yang melarang akses ke sistem file lokal oleh applet. Meskipun hal ini dapat menyebabkan frustrasi bagi pengembang, ini adalah fitur keamanan penting bagi pengguna akhir. Tanpanya, applet akan bebas memodifikasi konten hard drive pengguna, atau membaca isinya dan mengirimkan informasi ini kembali melalui jaringan.

Applet yang ditandatangani secara digital dapat meminta izin untuk mengakses sistem file lokal, tetapi cara termudah mengatasi masalah ini adalah dengan membaca dan menulis ke file jarak jauh yang terletak di drive jaringan. Misalnya, dalam hubungannya dengan skrip atau servlet CGI, Anda dapat mengirim permintaan HTTP untuk menyimpan dan mengambil data.

Pertanyaan Bagaimana cara memanggil metode Java dari Javascript?

Jawaban Direct Web Remoting adalah pustaka Java open source yang memungkinkan integrasi AJAX dengan mudah ke situs web Anda. Ini memungkinkan Anda memanggil metode Java langsung dari Javascript.

Pertanyaan Bagaimana cara menemukan apakah ada parameter di objek permintaan?

Jawaban

boolean hasFoo = !(request.getParameter(”foo”) == null
|| request.getParameter(”foo”).equals(””));

Atau

boolean hasParameter =request.getParameterMap().contains(theParameter); //(yang mana yang dapat bekerja/berfungsi di Servlet 2.3+)

Pertanyaan Apa perbedaan antara RequestDispatcher dan sendRedirect?

Jawaban RequestDispatcher: pengalihan sisi server dengan objek permintaan dan respons. sendRedirect: Pengalihan sisi klien dengan objek permintaan dan respons baru.

PertanyaanApa perbedaan antara direktif include dan jsp include?

Jawaban<% @ include>: Digunakan untuk menyertakan sumber daya statis selama waktu terjemahan. JSP meliputi: Digunakan untuk menyertakan konten dinamis atau konten statis selama runtime.

Pertanyaan Apa perbedaan dalam menggunakan request.getRequestDispatcher() dan context.getRequestDispatcher()?

Jawaban request.getRequestDispatcher (path): Untuk membuatnya, kita perlu memberikan jalur relatif dari sumber daya, context.getRequestDispatcher (path): Untuk membuatnya, kita perlu memberikan jalur absolut dari sumber daya.

Pertanyaan Apa perbedaan antara ServletContext dan PageContext?

Jawaban ServletContext: Memberikan informasi tentang container. PageContext: Memberikan informasi tentang Request.

Pertanyaan Ada berapa jenis Driver JDBC dan apa saja?

Jawaban Ada 4 tipe JDBC Drivers:

JDBC-ODBC Bridge Driver.
Native API Partly Java Driver.
Network protocol Driver.
JDBC Net pure Java Driver.

Pertanyaan Apa kueri yang digunakan untuk menampilkan semua nama tabel di SQL Server (Query analyzer)?

Jawaban pilih * dari information_schema.tables

Pertanyaan Bagaimana Anda memanggil prosedur tersimpan dari JDBC?

Jawaban Langkah pertama adalah membuat objek CallableStatement. Seperti halnya objek Pernyataan dan PreparedStatement, ini dilakukan dengan objek onnection terbuka. Objek CallableStatement berisi panggilan ke prosedur tersimpan. CallableStatement cs = con.prepareCall (”{panggil SHOW_SUPPLIERS}”); ResultSet rs = cs.executeQuery();

Pertanyaan Bagaimana Anda bisa menggunakan PreparedStatement?

Jawaban Jenis pernyataan khusus ini berasal dari Pernyataan kelas. Jika Anda memerlukan objek Pernyataan untuk dieksekusi berkali-kali, biasanya masuk akal untuk menggunakan objek PreparedStatement sebagai gantinya.

Keuntungannya adalah bahwa dalam banyak kasus, pernyataan SQL ini akan dikirim ke DBMS segera, di mana ia akan dikompilasi. Akibatnya, objek PreparedStatement tidak hanya berisi pernyataan SQL, tetapi pernyataan SQL yang telah dikompilasi sebelumnya. Ini berarti bahwa ketika PreparedStatement dijalankan, DBMS hanya dapat menjalankan pernyataan SQL PreparedStatement tanpa harus mengkompilasinya terlebih dahulu.

PreparedStatement updateSales = con.prepareStatement (”UPDATE COFFEES SET SALES =? WHERE COF_NAME =

Pertanyaan Apa sajakah jenis Pernyataan?

Jawaban Pernyataan reguler (gunakan metode createStatement), pernyataan yang disiapkan (gunakan metode preparedStatement) dan pernyataan yang dapat dipanggil (gunakan preparedCall)

Pertanyaan Bagaimana cara mengambil data dari ResultSet?

Jawaban JDBC mengembalikan hasil dalam objek ResultSet, jadi kita perlu mendeklarasikan instance kelas ResultSet untuk menyimpan hasil kita. Kode berikut menunjukkan mendeklarasikan objek ResultSet rs.

ResultSet rs = stmt.executeQuery (”PILIH COF_NAME, HARGA DARI KOPI”);

String s = rs.getString (”COF_NAME”);

Metode getString dipanggil pada objek ResultSet rs, jadi getString() akan mengambil (mendapatkan) nilai yang disimpan di kolom COF_NAME di baris rs saat ini.

Pertanyaan Bagaimana Anda membuat pernyataan JDBC dan apa itu?

Jawaban Sebuah objek Pernyataan adalah apa yang mengirimkan pernyataan SQL Anda ke DBMS. Anda cukup membuat objek Pernyataan dan kemudian menjalankannya, menyediakan metode eksekusi yang sesuai dengan pernyataan SQL yang ingin Anda kirim.

Untuk pernyataan SELECT, metode yang digunakan adalah executeQuery. Untuk pernyataan yang membuat atau mengubah tabel, metode yang digunakan adalah executeUpdate.

Dibutuhkan contoh koneksi aktif untuk membuat objek Pernyataan. Dalam contoh berikut, kami menggunakan koneksi objek con kami untuk membuat objek Pernyataan Pernyataan stmt = con.createStatement();

Pertanyaan Apa yang akan dilakukan Class.forName saat memuat driver?

Jawaban Ini digunakan untuk membuat instance driver dan mendaftarkannya dengan DriverManager. Ketika kamu telah memuat driver, itu tersedia untuk membuat koneksi dengan DBMS.

Pertanyaan Bagaimana cara memuat driver?

Jawaban Memuat driver atau driver yang ingin Anda gunakan sangat sederhana dan hanya melibatkan satu baris kode. Jika, misalnya, Anda ingin menggunakan driver JDBC-ODBC Bridge, kode berikut akan memuatnya:

Class.forName (”sun.jdbc.odbc.JdbcOdbcDriver”);

Dokumentasi driver Anda akan memberi Anda nama kelas yang akan digunakan. Misalnya, jika nama kelasnya adalah jdbc.DriverXYZ, Anda akan memuat driver dengan baris kode berikut:

Class.forName (”jdbc.DriverXYZ”);

Pertanyaan Apa itu sumber data?

Jawaban Kelas DataSource menghadirkan tingkat abstraksi lain daripada secara langsung menggunakan objek koneksi.

Sumber data dapat dirujuk oleh JNDI. Sumber Data dapat mengarah ke RDBMS, Sistem file, DBMS apa pun, dll.

Pertanyaan Apa itu commit 2 fase?

Jawaban Komit 2-fase adalah algoritme yang digunakan untuk memastikan integritas transaksi yang melakukan.

Pada Tahap 1, koordinator transaksi menghubungi calon peserta dalam transaksi. Semua peserta setuju untuk menjadikan hasil transaksi permanen tetapi tidak segera dilakukan. Para peserta mencatat informasi ke disk untuk memastikan mereka dapat menyelesaikannya.

Pada fase 2 f semua peserta setuju untuk berkomitmen, koordinator mencatat kesepakatan itu dan hasilnya diputuskan. Pencatatan perjanjian ini di log berakhir di Tahap 2, koordinator memberi tahu setiap peserta tentang keputusan tersebut, dan mereka memperbarui sumber daya mereka secara permanen.

Pertanyaan Apa itu Metadata dan mengapa saya harus menggunakannya?

Jawaban Metadata ('data tentang data') adalah informasi tentang salah satu dari dua hal: Informasi database (java.sql.DatabaseMetaData), atau Informasi tentang ResultSet tertentu (java.sql.ResultSetMetaData).

Gunakan DatabaseMetaData untuk menemukan informasi tentang database Anda, seperti kapabilitas dan strukturnya. Gunakan ResultSetMetaData untuk menemukan informasi tentang hasil kueri SQL, seperti ukuran dan tipe kolom

Pertanyaan Apa itu “bacaan kotor”?

Jawaban Cukup sering dalam pemrosesan database, kita menemukan situasi di mana satu transaksi dapat mengubah nilai, dan transaksi kedua dapat membaca nilai ini sebelum perubahan asli dilakukan atau dibatalkan.

Ini dikenal sebagai skenario baca kotor karena selalu ada kemungkinan bahwa transaksi pertama dapat mengembalikan perubahan, mengakibatkan transaksi kedua membaca nilai yang tidak valid.

Meskipun Anda dapat dengan mudah memerintahkan database untuk melarang pembacaan kotor, hal ini biasanya menurunkan kinerja aplikasi Anda karena

peningkatan penguncian penguncian. Melarang pembacaan kotor juga menyebabkan penurunan konkurensi sistem.

Pertanyaan Apa keuntungan menggunakan PreparedStatement?

Jawaban Jika kita menggunakan PreparedStatement, waktu eksekusi akan lebih singkat. Objek PreparedStatement tidak hanya berisi pernyataan SQL, tetapi pernyataan SQL yang telah dikompilasi sebelumnya. Ini berarti bahwa ketika PreparedStatement dijalankan, RDBMS hanya dapat menjalankan pernyataan Sql PreparedStatement tanpa harus mengkompilasinya terlebih dahulu.

Pertanyaan Kapan kami akan mendenormalisasi data?

Jawaban Data denormalization adalah prosedur terbalik, dilakukan semata-mata untuk alasan peningkatan performa. Mungkin efisien untuk sistem throughput tinggi untuk mereplikasi data untuk data tertentu.

Pertanyaan Apa itu backup dingin, backup panas, pemulihan backup hangat?

Jawaban Cold backup (Semua file ini harus di-backup pada saat yang sama, sebelum database di-restart). Hot backup (nama resminya adalah 'backup online') adalah backup yang diambil dari setiap tablespace saat database sedang berjalan dan sedang diakses oleh pengguna.

Pertanyaan Apakah JDBC-ODBC Bridge multi-threaded?

Jawaban

Tidak, Jembatan JDBC-ODBC tidak mendukung akses bersamaan dari utas yang berbeda. Jembatan JDBC-ODBC menggunakan metode tersinkronisasi untuk membuat serial semua panggilan yang dibuat ke ODBC. Program Java multi-utas mungkin menggunakan Bridge, tetapi mereka tidak akan mendapatkan keuntungan dari multi-utas.

Pertanyaan Apa itu Applet? Haruskah applet memiliki konstruktor?

Jawaban Applet adalah program kecil yang ditransfer melalui Internet, secara otomatis diinstal dan dijalankan sebagai bagian dari browser web. Applets mengimplementasikan fungsionalitas klien.

Applet adalah program dinamis dan interaktif yang berjalan di dalam halaman Web yang ditampilkan oleh browser yang mendukung Java. Kami tidak memiliki konsep Konstruktor di Applet. Applet dapat dipanggil baik melalui browser atau melalui utilitas Appletviewer yang disediakan oleh JDK.

Pertanyaan Apakah metode utama dapat dinyatakan final?

Jawaban Ya, metode utama dapat dinyatakan final, selain statis publik.

Pertanyaan Bisakah kelas saya kelas publik didefinisikan dalam file sumber bernama YourClass.java?

Jawaban Tidak ada nama file sumber, jika berisi kelas publik, harus sama dengan nama kelas publik itu sendiri dengan ekstensi .java.

Pertanyaan Berapakah nilai default dari variabel lokal?

Jawaban Variabel lokal tidak diinisialisasi ke nilai default apa pun, baik primitif maupun referensi objek. Jika Anda mencoba menggunakan variabel ini tanpa menginisialisasinya secara eksplisit, compiler java tidak akan mengkompilasi kode. Ini akan mengeluh tentang varaible lokal yang tidak diinisiasi.

Pertanyaan Apa saja cakupan yang berbeda untuk variabel Java?

Jawaban Cakupan variabel Java ditentukan oleh konteks di mana variabel tersebut dideklarasikan. Jadi variabel java dapat memiliki salah satu dari tiga cakupan pada titik waktu tertentu.

Instansi: - Ini adalah variabel tingkat objek yang khas, mereka diinisialisasi ke nilai default pada saat pembuatan objek, dan tetap dapat diakses selama objek dapat diakses.

Lokal: - Ini adalah variabel yang ditentukan dalam suatu metode. Mereka tetap accessbile hanya selama eksekusi metode. Ketika metode selesai dieksekusi, variabel-variabel ini keluar dari ruang lingkup.

Statis: - Ini adalah variabel tingkat kelas. Mereka diinisialisasi saat kelas dimuat di JVM untuk pertama kalinya dan tetap di sana selama kelas tetap dimuat. Mereka tidak terikat pada contoh objek tertentu.

Pertanyaan Berapakah nilai awal dari referensi objek yang didefinisikan sebagai variabel instan?

Jawaban Referensi objek semua diinisialisasi ke null di java. Namun untuk melakukan sesuatu yang berguna dengan referensi ini, Anda harus menyetelnya ke objek yang valid, jika tidak, Anda akan mendapatkan NullPointerExceptions di mana pun Anda mencoba menggunakan referensi yang diinisialisasi default tersebut.

Pertanyaan Apa yang terjadi jika Anda tidak menginisialisasi variabel instance dari salah satu tipe primitif di Java?

Jawaban Java secara default menginisialisasinya ke nilai default untuk tipe primitif tersebut. Jadi int akan diinisialisasi ke 0, boolean akan diinisialisasi ke false.

Pertanyaan Apakah keyword utama, selanjutnya hapus, keluar di java?

Jawaban Tidak, itu bukan keyword di Java. delete bukanlah keyword di Java. Java tidak menggunakan destruktor eksplisit seperti yang dilakukan C++. Untuk keluar dari program secara eksplisit Anda menggunakan metode keluar di objek Sistem.

Pertanyaan Apakah String adalah tipe data primitif di Java?

Jawaban No String bukanlah tipe data primitif di Java, meskipun ini adalah salah satu objek yang paling banyak digunakan. String di Java adalah instance kelas String yang didefinisikan dalam paket java.lang.

Pertanyaan Bisakah file .java berisi lebih dari satu kelas java?

Jawaban Ya, file .java berisi lebih dari satu kelas java, asalkan salah satunya adalah kelas publik.

Pertanyaan Apakah file Empty.java adalah file sumber yang valid?

Jawaban Ya, file .java kosong adalah file sumber yang benar-benar valid.

Pertanyaan Apa yang dimaksud dengan metode atau kolom "statis"?

Jawaban Variabel dan metode statis dibuat hanya sekali per kelas. Dengan kata lain mereka adalah variabel kelas, bukan variabel instan.

Jika Anda mengubah nilai variabel statis di objek tertentu, nilai variabel itu berubah untuk semua instance kelas itu.

Metode statis dapat direferensikan dengan nama kelas daripada nama objek tertentu dari kelas tersebut (meskipun itu juga berfungsi). Begitulah cara metode pustaka seperti System.out.println() bekerja adalah bidang statis di kelas java.lang.System.

Pertanyaan Pengubah apa yang diizinkan untuk metode dalam Antarmuka?

Jawaban Hanya pengubah publik dan abstrak yang diperbolehkan untuk metode dalam antarmuka.

Pertanyaan Apa itu Dapat Dieksternalisasikan?

Jawaban Externalizable adalah Antarmuka yang memperluas Antarmuka Serializable. Dan mengirimkan data ke Aliran dalam Format Terkompresi. Ini memiliki dua metode, writeExternal (ObjectOuput out) dan readExternal (ObjectInput in)

Pertanyaan Metode apa yang harus diterapkan oleh semua utas?

Jawaban Semua tugas harus mengimplementasikan metode run(), apakah itu subkelas dari Thread atau mengimplementasikan antarmuka Runnable.

Pertanyaan Apa itu metode tersinkronisasi dan pernyataan tersinkronisasi?

Jawaban Metode yang disinkronkan adalah metode yang digunakan untuk mengontrol akses ke suatu objek. Sebuah thread hanya menjalankan metode tersinkronisasi setelah mendapatkan kunci untuk objek atau kelas metode tersebut.

Pernyataan tersinkronisasi mirip dengan metode tersinkronisasi. Pernyataan tersinkronisasi hanya dapat dijalankan setelah utas memperoleh kunci untuk objek atau kelas yang dirujuk dalam pernyataan tersinkronisasi.

Pertanyaan Bisakah objek yang tidak terjangkau menjadi dapat dijangkau lagi?

Jawaban Benda yang tidak terjangkau bisa jadi bisa dijangkau lagi. Ini bisa terjadi ketika metode finalize() objek dipanggil dan objek melakukan operasi yang membuatnya dapat diakses oleh objek yang dapat dijangkau.

Pertanyaan Bagaimana cara membuat serial objek ke file?

Jawaban Kelas yang instansinya akan diserialkan harus menerapkan antarmuka yang dapat diserialisasi. Kemudian Anda meneruskan instance ke ObjectOutputStream yang terhubung ke fileoutputstream. Ini akan menyimpan objek ke file.

Pertanyaan Metode antarmuka Serializable mana yang harus saya terapkan?

Jawaban Antarmuka yang dapat diserialisasi adalah antarmuka kosong, tidak berisi metode apa pun. Jadi kami tidak menerapkan metode apa pun.

Pertanyaan Bagaimana cara menyesuaikan proses seralisasi? yaitu bagaimana seseorang dapat mengontrol proses serialisasi?

Jawaban Ya adalah mungkin untuk memiliki kendali atas proses serialisasi. Kelas harus mengimplementasikan antarmuka Eksternal. Antarmuka ini berisi dua metode yaitu readExternal dan writeExternal. Anda harus menerapkan metode ini dan menulis logika untuk menyesuaikan proses serialisasi.

Pertanyaan Apa penggunaan umum serialisasi?

Jawaban Setiap kali suatu objek akan dikirim melalui jaringan, objek perlu diserialkan. Selain itu, jika status suatu objek ingin diselamatkan, objek perlu di-serilazed.

Pertanyaan Ketika Anda membuat serial sebuah objek, apa yang terjadi dengan referensi objek yang termasuk dalam objek?

Jawaban Mekanisme serialisasi menghasilkan grafik objek untuk serialisasi. Jadi itu menentukan apakah referensi objek yang disertakan berseri atau tidak. Ini adalah proses rekursif. Jadi ketika sebuah objek diserialkan, semua objek yang disertakan juga diserialkan bersama dengan objek aslinya.

Pertanyaan Apa yang harus diperhatikan saat membuat serialisasi objek?

Jawaban Seseorang harus memastikan bahwa semua objek yang disertakan juga dapat diserialkan. Jika salah satu objek tidak dapat diserialkan maka itu akan melontarkan NotSerializableException.

Pertanyaan Apa yang terjadi pada bidang statis kelas selama serialisasi?

Jawaban Ada tiga pengecualian di mana serialisasi tidak selalu membaca dan menulis ke aliran. Ini adalah:

Serialisasi mengabaikan bidang statis, karena bidang tersebut bukan bagian dari status keadaan tertentu.

Bidang kelas dasar hanya hendled jika kelas dasar itu sendiri dapat diserialkan

Bidang sementara.

Pertanyaan Apakah mengimpor paket juga mengimpor subpaket?

Jawaban Tidak, Anda harus mengimpor subpaket secara eksplisit.

Misalnya: Mengimpor com.MyTest. * Hanya akan mengimpor inner class paket MyTest. Ini tidak akan mengimpor kelas apa pun dalam subpaket mana pun.

Pertanyaan Apa urutan pemanggilan metode oleh AWT untuk applet?

Jawaban Saat applet dimulai, AWT memanggil metode berikut, dalam urutan ini:

Init()
Start()
Paint()
Ketika applet dihentikan, urutan pemanggilan metode berikut terjadi
Stop()
Destroy()

Pertanyaan Apa metode Siklus Hidup Applet? Jelaskan?

Jawaban Berikut ini adalah metode dalam siklus hidup Applet:

metode init() - dipanggil ketika applet pertama kali dimuat. Metode ini hanya dipanggil sekali dalam seluruh siklus applet. Metode ini biasanya menginisialisasi variabel yang akan digunakan di applet

metode start()- dipanggil setiap kali applet dimulai.

metode paint() - dipanggil saat applet diminimalkan atau disegarkan. Metode ini digunakan untuk menggambar string, gambar, dan gambar yang berbeda pada jendela applet.

metode stop() - dipanggil saat browser berpindah dari halaman applet.

metode Destroy()- dipanggil saat browser selesai dengan applet.

Pertanyaan Apa sajakah alternatif untuk inheritance?

Jawaban Delegasi adalah alternatif dari peinheritance. Delegasi berarti Anda menyertakan instance dari kelas lain sebagai variabel instance, dan meneruskan pesan ke instance tersebut.

Seringkali lebih aman daripada inheritance karena memaksa Anda untuk memikirkan setiap pesan yang Anda teruskan, karena instance dari kelas yang diketahui, bukan kelas baru, dan karena itu tidak memaksa Anda untuk menerima semua metode kelas super : Anda hanya dapat memberikan metode yang benar-benar masuk akal. Di sisi lain, ini membuat Anda menulis lebih banyak kode, dan lebih sulit untuk digunakan kembali (karena ini bukan subclass).

Pertanyaan Jika saya menulis return di akhir blok percobaan, apakah finally block masih akan dieksekusi?

Jawaban Ya meskipun Anda menulis return sebagai pernyataan terakhir di blok percobaan dan tidak ada pengecualian, blok akhirnya akan dieksekusi. Finally block akan dieksekusi dan kemudian kontrol kembali.

Pertanyaan Apakah perlu bahwa setiap blok percobaan harus diikuti oleh catch block?

Jawaban Tidak perlu setiap blok percobaan harus diikuti oleh catch block. Ini harus diikuti oleh catch blockan ATAU blok akhirnya. Dan pengecualian apa pun yang mungkin dilemparkan harus dideklarasikan dalam klausa lemparan metode.

Pertanyaan Apa sajakah cara untuk menangani pengecualian?

Jawaban Ada dua cara untuk menangani pengecualian,

Dengan membungkus kode yang diinginkan dalam blok percobaan diikuti dengan catch block untuk menangkap pengecualian. Dan

Buat daftar pengecualian yang diinginkan dalam klausa lemparan dari metode dan biarkan pemanggil metode tersebut menghadapi pengecualian tersebut.

Pertanyaan Bagaimana pengecualian menembus melalui kode?

Jawaban Sebuah pengecualian yang tidak tertangani bergerak ke atas tumpukan metode untuk mencari kecocokan Ketika sebuah pengecualian dilempar dari kode yang dibungkus dalam blok percobaan diikuti oleh satu atau lebih catch block, pencarian dilakukan untuk catch blockan yang cocok. Jika jenis yang cocok ditemukan maka blok itu akan dipanggil. Jika jenis yang cocok tidak ditemukan, pengecualian akan naik ke tumpukan metode dan mencapai metode pemanggil. Prosedur yang sama diulangi jika metode pemanggil termasuk dalam blok coba tangkap. Proses ini berlanjut hingga blok penangkap yang menangani jenis pengecualian yang sesuai ditemukan. Jika tidak menemukan blok seperti itu maka akhirnya program dihentikan.

Pertanyaan Apa itu pengecualian waktu proses?

Jawaban Runtime exception adalah pengecualian yang muncul pada saat runtime karena salah input data atau karena logika bisnis yang salah, dll. Ini tidak diperiksa oleh compiler pada waktu kompilasi.

Pertanyaan Apa yang dimaksud dengan pengecualian yang dicentang?

Jawaban Terkecuali yang dicentang adalah yang dipaksa oleh kompilator Java untuk Anda tangkap. misalnya IOException dicentang Pengecualian.

Pertanyaan Apakah Java menyediakan konstruksi untuk mengetahui ukuran suatu objek?

Jawaban Tidak ada ukuran operator di java. Jadi tidak ada cara langsung untuk menentukan ukuran suatu objek secara langsung di Java.

Pertanyaan Apa itu antarmuka yang Dapat Dieksternalisasikan?

Jawaban Externalizable adalah antarmuka yang berisi dua metode readExternal dan writeExternal. Metode ini memberi Anda kontrol atas mekanisme serialisasi. Jadi, jika kelas Anda mengimplementasikan antarmuka ini, Anda dapat menyesuaikan proses serialisasi dengan mengimplementasikan metode ini.

Pertanyaan Jenis parameter apa yang lewat yang didukung Java?

JawabanDi java, argumen selalu diteruskan oleh nilai.

Pertanyaan Bisakah kelas tingkat atas bersifat pribadi atau dilindungi?

Jawaban Tidak, kelas tingkat atas tidak boleh pribadi atau dilindungi. Ini dapat memiliki "publik" atau tanpa pengubah. Jika tidak memiliki pengubah, itu seharusnya memiliki akses default. Jika kelas tingkat atas dideklarasikan sebagai privat, kompilator akan mengeluh bahwa “pengubah privat tidak diizinkan di sini”. Ini berarti bahwa kelas tingkat atas tidak bisa menjadi pribadi. Sama halnya dengan dilindungi.

Pertanyaan Apa perbedaan antara mendeklarasikan variabel dan mendefinisikan variabel?

Jawaban Dalam deklarasi kita hanya menyebutkan jenis variabel dan namanya. Kami tidak memulainya. Tapi mendefinisikan berarti deklarasi + inisialisasi.

misal: String s; hanyalah sebuah deklarasi sementara String s = new String ("abcd"); Atau String s = “abcd”; keduanya adalah definisi.

Kelas anggota - Inner class anggota sama seperti metode anggota dan variabel anggota lainnya dan akses ke kelas anggota dibatasi, seperti metode dan variabel. Ini berarti kelas anggota publik bertindak serupa dengan kelas tingkat atas bersarang. Perbedaan utama antara kelas anggota dan kelas tingkat atas bersarang adalah bahwa kelas anggota memiliki akses ke instance tertentu dari kelas yang melingkupi.

Kelas lokal - Kelas lokal seperti variabel lokal, khusus untuk satu blok kode. Visibilitas mereka hanya dalam blok deklarasi mereka. Agar kelas bisa berguna di luar blok deklarasi, itu perlu mengimplementasikan antarmuka yang lebih tersedia untuk umum. Karena kelas lokal bukan anggota, pengubah public, protected, private, dan static tidak bisa digunakan.

Kelas anonim -Inner class anonim memperluas inner class lokal satu tingkat lebih jauh. Karena kelas anonim tidak memiliki nama, Anda tidak dapat menyediakan konstruktor.

Pertanyaan Apa yang Menggantikan?

Jawaban Ketika kelas mendefinisikan metode menggunakan nama yang sama, tipe kembalian, dan argumen sebagai metode dalam superclassnya, metode di kelas tersebut menimpa metode di superclass.

Ketika metode dipanggil untuk objek kelas, itu adalah definisi baru dari metode yang dipanggil, dan bukan definisi metode dari superclass. Metode dapat diganti menjadi lebih publik, bukan lebih pribadi.

Pertanyaan Apa itu Pengecualian yang Dicentang dan Tidak Dicentang?

Jawaban Sebuah pengecualian yang dicentang adalah beberapa subkelas dari Exception (atau Exception itu sendiri), tidak termasuk kelas RuntimeException dan subclassnya.

Membuat pengecualian dicentang akan memaksa pemrogram klien untuk menangani kemungkinan pengecualian akan dilemparkan. misalnya, IOException dilemparkan oleh metode read() java.io.FileInputStream ·

Pengecualian yang tidak dicentang adalah RuntimeException dan semua subkelasnya. Kesalahan Kelas dan subkelasnya juga tidak dicentang. Namun, dengan pengecualian yang tidak dicentang, kompilator tidak memaksa pemrogram klien untuk menangkap pengecualian atau mendeklarasikannya dalam klausa lemparan. Faktanya, programmer klien bahkan mungkin tidak tahu bahwa pengecualian dapat dilemparkan. misalnya, StringIndexOutOfBoundsException dilemparkan oleh metode charAt() String · Pengecualian yang diperiksa harus ditangkap pada waktu kompilasi. Pengecualian waktu proses tidak perlu. Kesalahan seringkali tidak bisa terjadi.

Pertanyaan Bagaimana cara membuktikan bahwa array tidak null tetapi kosong menggunakan satu baris kode?

Jawaban Cetak args.length. Ini akan mencetak 0. Artinya kosong. Tetapi jika nilainya null, maka NullPointerException akan muncul saat mencoba mencetak args.length.

Pertanyaan Apa argumen pertama dari array String dalam metode utama?

Jawaban Array String kosong. Itu tidak memiliki elemen apa pun. Ini tidak seperti C/C++ di mana elemen pertama secara default adalah nama program.

Pertanyaan Apa itu Iterator?

Jawaban Beberapa class collection menyediakan traversal kontennya melalui antarmuka java.util.Iterator. Antarmuka ini memungkinkan Anda untuk berjalan melalui sekumpulan objek, yang beroperasi pada setiap objek secara bergantian. Ingatlah saat menggunakan Iterator yang berisi snapshot dari koleksi pada saat Iterator diperoleh; umumnya tidak disarankan untuk mengubah koleksi itu sendiri saat melintasi Iterator.

Pertanyaan Perbedaan antara Vector dan ArrayList?

Jawaban Vector disinkronkan sedangkan arraylist tidak.

Pertanyaan Perbedaan antara HashMap dan HashTable?

Jawaban Kelas HashMap secara kasar setara dengan Hashtable, kecuali kelas itu tidak tersinkronisasi dan mengizinkan nulls. (HashMap memungkinkan nilai null sebagai kunci dan nilai sedangkan Hashtable tidak mengizinkan).

HashMap tidak menjamin bahwa urutan peta akan tetap konstan dari waktu ke waktu. HashMap tidak tersinkronisasi dan Hashtable disinkronkan.

Pertanyaan Apa itu HashMap dan Peta?

Jawaban Map adalah Interface dan Hashmap adalah kelas yang mengimplementasikannya.

Pertanyaan Apa itu pass by reference dan pass by value?

Jawaban Pass By Reference berarti meneruskan alamat itu sendiri daripada meneruskan nilainya. Pass by Value berarti meneruskan salinan nilai yang akan diteruskan.

Pertanyaan Jelaskan sinkronisasi sehubungan dengan multithreading.

Jawaban Sehubungan dengan multithreading, sinkronisasi adalah kemampuan untuk mengontrol akses beberapa utas ke sumber daya bersama. Tanpa sinkronisasi, satu utas dapat mengubah variabel bersama sementara utas lain sedang dalam proses menggunakan atau memperbarui variabel bersama yang sama. Ini biasanya menyebabkan kesalahan yang signifikan.

Pertanyaan Apa tujuan Garbage Collection di Java, dan kapan digunakan?

Jawaban Tujuan dari Garbage Collection adalah untuk mengidentifikasi dan membuang objek yang tidak lagi dibutuhkan oleh suatu program sehingga sumber dayanya dapat diperoleh kembali dan digunakan kembali. Objek Java tunduk pada Garbage Collection ketika menjadi tidak terjangkau oleh program yang digunakan.

Pertanyaan Apa perbedaan antara konstruktor dan metode?

Jawaban Konstruktor adalah fungsi anggota kelas yang digunakan untuk membuat objek kelas itu. Ini memiliki nama yang sama dengan kelas itu sendiri, tidak memiliki tipe kembalian, dan dipanggil menggunakan operator baru.

Metode adalah fungsi anggota biasa dari suatu kelas. Ini memiliki namanya sendiri, tipe kembali (yang mungkin kosong), dan dipanggil menggunakan operator titik.

Pertanyaan Apa yang disebut dengan final/akhir?

Jawaban Kelas terakhir tidak dapat diperpanjang, yaitu, kelas terakhir tidak boleh menjadi subkelas. Metode terakhir tidak dapat diganti ketika kelasnya diwarisi. Anda tidak dapat mengubah nilai variabel akhir (adalah konstanta).

Pertanyaan Apa itu statis di java?

Jawaban Statis berarti satu per kelas, bukan satu untuk setiap objek tidak peduli berapa banyak contoh kelas yang mungkin ada. Ini berarti Anda dapat menggunakannya tanpa membuat instance kelas. Metode statis bersifat final secara implisit, karena penggantian dilakukan berdasarkan jenis objek, dan metode statis dilampirkan ke kelas, bukan objek.

Metode statis dalam superkelas dapat dibayangi oleh metode statis lain dalam subkelas, selama metode asli tidak dinyatakan final. Namun, Anda tidak dapat mengganti metode statis dengan metode nonstatis. Dengan kata lain, Anda tidak dapat mengubah metode statis menjadi metode instance dalam subclass.

Pertanyaan Bagaimana jika metode utama dideklarasikan sebagai private?

Jawaban Program dikompilasi dengan benar tetapi pada waktu proses akan memberikan "Metode utama bukan publik." pesan.

Pertanyaan Apa perbedaan antara HttpServlet dan GenericServlet?

Jawaban Sebuah GenericServlet memiliki metode service() yang ditujukan untuk menangani permintaan. HttpServlet memperluas GenericServlet dan menambahkan dukungan untuk metode doGet(), doPost(), doHead() (HTTP 1.0) plus doPut(), doOptions(), doDelete(), doTrace() (HTTP 1.1).

Kedua kelas ini abstrak

Pertanyaan Apa itu prainisialisasi servlet?

JawabanSebuah kontainer tidak menginisialisasi servlet segera setelah dijalankan, ia menginisialisasi servlet ketika menerima permintaan servlet itu untuk pertama kalinya. Ini disebut pemuatan lambat. Spesifikasi servlet menentukan elemen, yang dapat ditentukan dalam deskriptor penerapan untuk membuat kontainer servlet memuat dan menginisialisasi servlet segera setelah dimulai. Proses memuat servlet sebelum permintaan apa pun masuk disebut preloading atau preinitializing servlet.

Pertanyaan Apa perbedaan antara ServletContext dan ServletConfig?

Jawaban

ServletContext: Mendefinisikan sekumpulan metode yang digunakan servlet untuk berkomunikasi dengan kontainer servletnya, misalnya, untuk mendapatkan tipe MIME file, mengirimkan permintaan, atau menulis ke file log. Objek ServletContext terkandung dalam objek ServletConfig, yang menyediakan server Web servlet saat servlet diinisialisasi

ServletConfig: Objek yang dibuat setelah servlet dibuat dan konstruktor defaultnya dibaca. Itu dibuat untuk meneruskan informasi inisialisasi ke servlet.

Pertanyaan Jelaskan ServletContext.

Jawaban ServletContext adalah jendela untuk servlet untuk melihat lingkungannya. Servlet dapat menggunakan antarmuka ini untuk mendapatkan informasi seperti parameter inisialisasi untuk aplikasi web atau versi penampung servlet.

Setiap aplikasi web memiliki satu dan hanya satu ServletContext dan dapat diakses oleh semua sumber daya aktif aplikasi itu.

Pertanyaan Apa perbedaan antara metode getRequestDispatcher (jalur String) dari antarmuka javax.servlet.ServletRequest dan antarmuka javax.servlet.ServletContext?

Jawaban Metode getRequestDispatcher (jalur String) dari antarmuka javax.servlet.ServletRequest menerima parameter jalur ke sumber daya yang akan disertakan atau diteruskan, yang dapat bersifat relatif terhadap permintaan servlet pemanggil. Jika jalur diawali dengan "/", ini diinterpretasikan sebagai relatif terhadap akar konteks saat ini.

Metode getRequestDispatcher (jalur String) dari antarmuka javax.servlet.ServletContext tidak dapat menerima jalur relatif. Semua jalur harus dimulai dengan "/" dan diinterpretasikan sebagai relatif terhadap akar konteks saat ini.

Pertanyaan Jelaskan metode siklus hidup seorang Servlet.

Jawaban Antarmuka javax.servlet.Servlet mendefinisikan tiga metode yang dikenal sebagai metode siklus hidup.

public void init (ServletConfig config) melempar ServletException

layanan public void (ServletRequest req, ServletResponse res) menampilkan ServletException, IOException
public void destroy()
Pertama servlet dibangun, kemudian diinisialisasi dengan metode init(). Setiap permintaan dari klien pada awalnya ditangani oleh metode service() sebelum mendelegasikan ke metode doXxx() dalam kasus HttpServlet.
Servlet dihapus dari layanan, dihancurkan dengan destruksi() methid, kemudian dibuang dan dikumpulkan dan diselesaikan.

Pertanyaan Jelaskan model keamanan java.

Jawaban Model keamanan Java adalah salah satu aspek bahasa yang paling menarik dan unik. Sebagian besar dipecah menjadi dua bagian: pengelola keamanan yang dapat disesuaikan pengguna yang memeriksa berbagai operasi API seperti akses file, dan pemverifikasi kode byte yang menegaskan validitas kode byte yang dikompilasi. kelas abstrak publik SecurityManager java.lang.SecurityManager adalah kelas abstrak yang subclass aplikasi berbeda untuk mengimplementasikan kebijakan keamanan tertentu. Ini memungkinkan aplikasi untuk menentukan apakah operasi tertentu akan menghasilkan pengecualian keamanan atau tidak.

Pertanyaan Apa itu daemon thread dan metode mana yang digunakan untuk membuat daemon thread?

Jawaban Daemon thread adalah thread berprioritas rendah yang berjalan sesekali di latar belakang melakukan operasi Garbage Collection untuk sistem runtime java. Metode setDaemon digunakan untuk membuat daemon thread.

Pertanyaan Apa itu Pengubah Transient dan Volatile?

Jawaban

Transient: Pengubah transien hanya berlaku untuk variabel dan tidak disimpan sebagai bagian dari status Persisten objeknya. Variabel transien tidak berseri.
Volatile: Pengubah volatile hanya berlaku untuk variabel dan memberitahu compiler bahwa variabel yang dimodifikasi oleh volatile dapat diubah secara tidak terduga oleh bagian lain dari program.

Pertanyaan Apa itu Penjadwalan Waktu Preemptive dan Non Preemptive?

Jawaban Preemptive: Menjalankan tugas diberi porsi kecil waktu untuk dieksekusi dengan menggunakan pembagian waktu.
Non-Preemptive: Satu tugas tidak memberikan tugas lain kesempatan untuk dijalankan hingga selesai atau biasanya menghabiskan waktunya.

Pertanyaan Apa itu runnable?

Jawaban Ini adalah Antarmuka di mana Java mengimplementasikan Threads. Kelas dapat diperluas dari kelas mana pun tetapi jika mengimplementasikan Runnable, Thread dapat digunakan dalam aplikasi tertentu.

PertanyaanApa perbedaan antara set dan list?

Jawaban Set menyimpan elemen dengan cara yang tidak berurutan tetapi tidak mengandung elemen duplikat, sedangkan list menyimpan elemen secara berurutan tetapi mungkin berisi elemen duplikat.

Pertanyaan Apa itu manajer tata letak?

Jawaban Manajer layout adalah sebuah obyek yang digunakan untuk mengatur komponen-komponen dalam sebuah wadah.

Pertanyaan Apa perbedaan antara kelas umum dan non-publik?

Jawaban Kelas publik dapat diakses di luar paketnya. Kelas non-publik tidak dapat diakses di luar paketnya.

Pertanyaan Apa perbedaan antara instanceof dan isInstance?

Jawaban instanceof digunakan untuk memeriksa apakah suatu objek dapat dilemparkan ke dalam tipe tertentu tanpa membuang classexception cor. isInstance() Menentukan apakah Objek yang ditentukan kompatibel dengan tugas dengan objek yang diwakili oleh Kelas ini. Metode ini adalah ekuivalen dinamis dari operator instanceof bahasa Java. Metode ini mengembalikan nilai true jika argumen Objek yang ditentukan adalah bukan nol dan dapat dilemparkan ke tipe referensi yang diwakili oleh objek Kelas ini tanpa memunculkan ClassCastException. Ia mengembalikan false jika tidak.

Pertanyaan Apa arti keyword "abstrak" di depan metode? Kelas?

Jawaban Abstract keyword mendeklarasikan metode atau kelas. Jika suatu metode memiliki keyword abstrak di depannya, itu disebut metode abstrak. Metode abstrak hs no body. Metode abstrak hanya memiliki argumen dan tipe kembalian. Metode abstrak bertindak sebagai metode placeholder yang diimplementasikan di subclass.

Kelas abstrak tidak dapat dibuat instance-nya Jika sebuah kelas dideklarasikan sebagai abstrak, tidak ada objek dari kelas tersebut yang dapat dibuat. Jika sebuah kelas berisi metode abstrak, ia harus dideklarasikan sebagai abstrak.

Pertanyaan Apa itu JDBC? Jelaskan langkah-langkah yang diperlukan untuk menjalankan kueri SQL menggunakan JDBC.

Jawaban JDBC adalah API Java murni yang digunakan untuk menjalankan pernyataan SQL. Ini menyediakan sekumpulan kelas dan antarmuka yang dapat digunakan oleh pengembang untuk menulis aplikasi database.

Langkah-langkah yang diperlukan untuk menjalankan kueri SQL menggunakan JDBC:

Buka koneksi ke database.
Jalankan pernyataan SQL.
Memproses hasil.
Tutup koneksi ke database

Pertanyaan Apa itu RMI?

Jawaban RMI adalah singkatan dari Remote Method Invocation. Pendekatan tradisional untuk mengeksekusi kode pada mesin lain di seluruh jaringan telah membingungkan serta membosankan dan rawan kesalahan untuk diterapkan. Cara terbaik untuk memikirkan masalah ini adalah bahwa beberapa objek kebetulan hidup di komputer lain, dan Anda dapat mengirim pesan ke objek jarak jauh dan mendapatkan hasil seolah-olah objek tersebut hidup di komputer lokal Anda. Penyederhanaan ini persis seperti yang dimungkinkan oleh Java Remote Method Invocation (RMI).

Pertanyaan Apakah metode asli itu? Bagaimana cara anda menggunakannya?

Jawaban Native method adalah metode yang didefinisikan sebagai metode statis publik dalam kelas java, tetapi implementasinya disediakan dalam bahasa pemrograman lain seperti C.

Pertanyaan Apa arti keyword "sinkronkan" di java. Kapan Anda menggunakannya?

Jawaban Synchronize digunakan ketika Anda ingin membuat thread metode Anda aman. Kerugian dari sinkronisasi adalah akan memperlambat program. Juga jika tidak ditangani dengan baik akan berakhir dengan dead lock.

Hanya gunakan (dan minimalkan penggunaannya) sinkronisasi saat menulis kode multithread karena ada biaya kecepatan (hingga lima hingga enam kali lebih lambat, tergantung pada waktu eksekusi metode tersinkronisasi/tidak tersinkronisasi) yang terkait dengan penggunaannya.

Dalam kasus pengubah metode tersinkronisasi, kode byte yang dihasilkan sama persis dengan metode tidak tersinkronisasi. Satu-satunya perbedaan adalah bahwa tanda yang disebut tanda properti ACC_SYNCRONIZED dalam struktur method_info metode disetel jika pengubah metode tersinkronisasi ada.

Selain itu, keyword yang disinkronkan dapat membuat ukuran kode lebih besar jika digunakan dalam komponen metode karena bytecode untuk monitorenter/monitorexit dihasilkan sebagai tambahan untuk exception handling apa pun.

Pertanyaan Apa perbedaan antara Vector dan Array. Diskusikan keuntungan dan kerugiannya

Jawaban Kelas wadah vektor menggeneralisasi konsep larik C biasa. Seperti array, vektor adalah struktur data yang diindeks, dengan nilai indeks yang berkisar dari 0 hingga satu kurang dari jumlah elemen yang ada dalam struktur.

Juga seperti larik, nilai paling sering ditetapkan dan diekstrak dari vektor menggunakan operator subskrip. Namun, vektor berbeda dari array dalam hal penting berikut ini. Ukuran vektor dapat berubah secara dinamis. Elemen baru dapat disisipkan di ujung vektor, atau di tengah. Penting untuk dicatat, bagaimanapun, bahwa meskipun kemampuan ini disediakan, penyisipan ke tengah vektor tidak seefisien penyisipan ke tengah daftar. Vektor memiliki lebih banyak "pengetahuan diri" daripada array biasa.

Secara khusus, vektor dapat ditanyai tentang ukurannya, tentang jumlah elemen yang berpotensi dapat ditampungnya (yang mungkin berbeda dari ukurannya saat ini), dan seterusnya. Vektor hanya dapat menampung referensi ke objek dan bukan tipe primitif.

Implementasi Vektor biasanya lebih lambat daripada array karena semua fungsionalitas yang menyertainya. Seperti yang diterapkan di Java, vektor adalah kelas yang aman untuk thread dan karenanya semua metode adalah metode sinkron, yang membuatnya sangat lambat.

Pertanyaan Java mengatakan "tulis sekali, jalankan di mana saja". Apa beberapa hal yang tidak sepenuhnya benar?

Jawaban Setiap kali Anda menggunakan panggilan sistem khusus untuk satu sistem operasi dan tidak membuat panggilan alternatif untuk sistem operasi lain, program

Anda tidak akan berfungsi dengan benar. Sistem Solaris dan sistem Intel mengurutkan bit integer secara berbeda. (Anda mungkin pernah mendengar tentang little endian vs. big endian)

Jika kode Anda menggunakan bit shifting, atau operator biner lainnya, mereka tidak akan berfungsi pada sistem yang memiliki endianisme berlawanan.

Pertanyaan Apa perbedaan antara Applet dan Application?

Jawaban

Applet dapat disematkan di halaman HTML dan diunduh melalui Internet sedangkan Aplikasi tidak memiliki dukungan khusus dalam HTML untuk disematkan atau diunduh.

Applet hanya dapat dijalankan di dalam wadah yang kompatibel dengan java, seperti browser atau appletviewer sedangkan Aplikasi dijalankan pada baris perintah oleh java.exe atau jview.exe.

Applet dijalankan di bawah batasan keamanan ketat yang melarang operasi tertentu (keamanan model kotak pasir) sedangkan Aplikasi tidak memiliki batasan keamanan yang melekat.

Applet tidak memiliki metode main() seperti dalam aplikasi. Sebaliknya mereka beroperasi pada mekanisme yang sama sekali berbeda di mana mereka diinisialisasi oleh init(), dimulai oleh start(), dihentikan oleh stop() atau dihancurkan oleh destruksi().

Pertanyaan Bagaimana Anda bisa memaksa semua kelas turunan untuk mengimplementasikan metode yang ada di kelas dasar?

Jawaban Membuat dan menerapkan antarmuka akan menjadi cara terbaik untuk situasi ini. Buat saja antarmuka dengan metode kosong yang memaksa programmer untuk mengimplementasikan semua metode yang ada di bawahnya.

Cara lain untuk mencapai tugas ini adalah dengan mendeklarasikan kelas sebagai abstrak dengan semua metodenya abstrak.

Pertanyaan Apa itu kelas abstrak, metode abstrak?

Jawaban Secara sederhana berbicara kelas atau metode yang memenuhi syarat dengan keyword "abstrak" adalah kelas abstrak atau metode abstrak.

Kamu membuat kelas abstrak saat kamu ingin memanipulasi sekumpulan kelas melalui antarmuka umum. Semua metode kelas turunan yang cocok dengan tanda tangan deklarasi kelas dasar akan dipanggil menggunakan mekanisme pengikatan dinamis. Metode abstrak adalah metode yang tidak lengkap. Ini hanya memiliki deklarasi dan tidak ada komponen metode. Berikut adalah sintaks untuk deklarasi metode abstrak: abstract void f();

Pertanyaan Apa perbedaan antara metode == dan sama?

Jawaban Metode sama dengan dapat dianggap melakukan perbandingan mendalam dari nilai suatu objek, sedangkan operator == melakukan perbandingan yang dangkal. Metode equals()membandingkan karakter di dalam objek string. == Operator membandingkan dua referensi objek untuk memeriksa apakah mereka merujuk ke instance yang sama atau tidak.

Pertanyaan Jelaskan, secara umum, bagaimana pengumpul sampah java bekerja?

Jawaban Lingkungan runtime Java menghapus objek ketika menentukan bahwa mereka tidak lagi digunakan.

Proses ini dikenal sebagai Garbage Collection. Lingkungan runtime Java mendukung pengumpul sampah yang secara berkala membebaskan memori yang digunakan oleh objek yang tidak lagi diperlukan.

Pengumpul sampah Java adalah pengumpul sampah mark-sweep yang memindai area memori dinamis Java untuk menemukan objek, menandai yang direferensikan. Setelah semua kemungkinan jalur ke objek diselidiki, objek yang tidak ditandai (yaitu tidak direferensikan) diketahui sebagai sampah dan dikumpulkan.

Pertanyaan Apa perbedaan antara kelas StringBuffer dan String?

Jawaban Sebuah string buffer mengimplementasikan urutan karakter yang bisa berubah. Buffer string seperti String, tetapi dapat dimodifikasi. Urutan berisi beberapa urutan karakter tertentu kapan saja, tetapi panjang dan konten urutan dapat diubah melalui panggilan metode tertentu. Kelas String mewakili string karakter. Semua string literal dalam program Java, seperti "abc" adalah konstan dan diimplementasikan sebagai instance kelas ini; nilainya tidak dapat diubah setelah dibuat.

Pertanyaan Bagaimana cara mencapai Inheritance Ganda di Java?

Jawaban Mekanisme antarmuka Java dapat digunakan untuk mengimplementasikan beberapa peinheritance, dengan satu perbedaan penting dari c++ cara melakukan MI: antarmuka yang diwariskan harus abstrak. Ini meniadakan kebutuhan untuk memilih di antara implementasi yang berbeda, karena dengan antarmuka tidak ada implementasi.

Pertanyaan Apa itu antarmuka?

Jawaban Interfaces menyediakan cara yang lebih canggih untuk mengatur dan mengontrol objek di sistem Anda. Keyword antarmuka membawa konsep abstrak selangkah lebih maju. Anda dapat menganggapnya sebagai kelas abstrak "murni". Ini memungkinkan pembuat untuk menetapkan formulir untuk kelas: nama metode, daftar argumen, dan tipe kembalian, tetapi tidak ada badan metode.

Antarmuka juga dapat berisi bidang, tetapi Keyword antarmuka membawa konsep abstrak selangkah lebih maju. Anda dapat menganggapnya sebagai kelas abstrak "murni". Ini memungkinkan pembuat untuk menetapkan formulir untuk kelas: nama metode, daftar argumen, dan tipe kembalian, tetapi tidak ada badan metode. Antarmuka juga dapat berisi kolom, tetapi Antarmuka mengatakan: "Seperti inilah tampilan semua kelas yang mengimplementasikan antarmuka khusus ini". Jadi, kode apa pun yang menggunakan antarmuka tertentu mengetahui metode apa yang mungkin dipanggil untuk antarmuka itu, dan itu saja. Jadi antarmuka digunakan untuk membuat "protokol" antar kelas.

Pertanyaan Bagaimana cara membuat aplikasi thread-safe?

Jawaban Anda harus menggunakan kata tersinkronisasi untuk menandai bagian penting dari kode. Anda juga dapat menggunakan metode sinkronisasi utas lainnya (lihat wait(), notify(), notifyAll() dll.

Pertanyaan Apa yang Anda ketahui tentang dukungan jaringan di Java?

Jawaban Java mendukung kelas "tingkat rendah" dan "tingkat tinggi". Kelas "tingkat rendah" menyediakan dukungan untuk pemrograman soket: kelas Socket, DatagramSocket, dan ServerSocket. Kelas "tingkat tinggi" menyediakan "Pemrograman web": kelas URL, URLEncoder, dan URLConnection.

Kelas pemrograman jaringan memudahkan pemrograman aplikasi jaringan, tetapi tidak menggantikan pengetahuan Anda tentang jaringan. Jaringan Java seperti apa pun di Java adalah platform-independen.

Pertanyaan Apa masalah yang dihadapi oleh programmer Java yang tidak menggunakan pengelola tata letak?

Jawaban Tanpa manajer tata letak, programmer Java dihadapkan pada penentuan bagaimana GUI mereka akan ditampilkan di beberapa sistem windowing dan menemukan ukuran dan pemosisian umum yang akan bekerja dalam batasan yang diberlakukan oleh setiap sistem windowing.

Pertanyaan Apa dua cara dasar di mana kelas yang dapat dijalankan sebagai utas dapat didefinisikan?

Jawaban Kelas thread dapat dideklarasikan sebagai subkelas dari Thread, atau mungkin menerapkan antarmuka Runnable.

Pertanyaan Apa tujuan blok pernyataan?

Jawaban Blok pernyataan digunakan untuk mengatur urutan pernyataan sebagai satu kelompok pernyataan.

Pertanyaan Apa tujuan Garbage Collection?

Jawaban Tujuan dari Garbage Collection adalah untuk mengidentifikasi dan membuang objek yang tidak lagi dibutuhkan oleh sebuah program sehingga sumber dayanya dapat diambil kembali dan digunakan kembali.

Pertanyaan Bagaimana ini dan super digunakan?

Jawaban Ini digunakan untuk merujuk ke instance objek saat ini. super digunakan untuk merujuk ke variabel dan metode kelas super dari instance objek saat ini.

Pertanyaan Apa itu filter I/O?

Jawaban Filter I/O adalah objek yang membaca dari satu aliran dan menulis ke aliran lain, biasanya mengubah data dengan cara tertentu saat diteruskan dari satu aliran ke aliran lainnya.

Pertanyaan Bagaimana elemen GridLayout diatur?

Jawaban Elemen-elemen tata letak GridBad memiliki ukuran yang sama dan ditata menggunakan kotak-kotak dari sebuah grid.

Pertanyaan Bagaimana this() dan super() digunakan dengan konstruktor?

Jawaban this() digunakan untuk memanggil konstruktor dari kelas yang sama. super() digunakan untuk menjalankan konstruktor superclass.

Pertanyaan Bagaimana kelas Kotak centang digunakan untuk membuat tombol radio?

Jawaban Dengan mengaitkan objek Checkbox dengan CheckboxGroup

Pertanyaan Jika suatu metode dinyatakan sebagai dilindungi, di manakah metode itu dapat diakses?

Jawaban Metode yang dilindungi hanya dapat diakses oleh kelas atau antarmuka dari paket yang sama atau dengan subkelas kelas di mana ia dideklarasikan.

Pertanyaan Kapan kompilator menyediakan konstruktor default untuk kelas?

Jawaban Compiler menyediakan konstruktor default untuk kelas jika tidak ada konstruktor lain yang disediakan.

Pertanyaan Apa kelas acara tingkat tertinggi dari model delegasi acara?

Jawaban Kelas java.util.EventObject adalah kelas tingkat tertinggi dalam hierarki kelas eventdelegation.

Pertanyaan Pembatasan apa yang ditempatkan pada nilai setiap kasus pernyataan switch?

Jawaban Selama kompilasi, nilai dari setiap kasus pernyataan switch harus dievaluasi ke nilai yang dapat dipromosikan ke nilai int.

Pertanyaan Bagaimana elemen CardLayout diatur?

Jawaban Elemen CardLayout ditumpuk, satu di atas yang lain, seperti setumpuk kartu.

Pertanyaan Untuk pernyataan mana yang masuk akal menggunakan label?

Jawaban Satu-satunya pernyataan yang masuk akal untuk menggunakan label adalah pernyataan yang dapat menyertakan pernyataan istirahat atau lanjutkan.

Pertanyaan Bagaimana pembulatan dilakukan di bawah pembagian integer?

Jawaban Bagian pecahan dari hasil tersebut dipotong. Ini dikenal sebagai pembulatan menuju nol.

Pertanyaan Bisakah sebuah objek dikumpulkan sampah saat masih bisa dijangkau?

Jawaban Objek yang dapat dijangkau tidak dapat dikumpulkan dari sampah. Hanya objek yang tidak terjangkau yang dapat dikumpulkan.

Pertanyaan Apa perbedaan antara Jendela dan Bingkai?

Jawaban Kelas Frame memperluas Window untuk menentukan jendela aplikasi utama yang dapat memiliki menu bar.

Pertanyaan Kapan referensi objek dapat dilemparkan ke referensi antarmuka?

Jawaban Referensi objek dikirim ke referensi antarmuka ketika objek mengimplementasikan antarmuka yang direferensikan.

PertanyaanApakah kelas Dictionary itu?

Jawaban Kelas Dictionary menyediakan kemampuan untuk menyimpan pasangan nilai-kunci.

Pertanyaan Apa status utas tingkat tinggi?

Jawaban Status thread tingkat tinggi siap, berjalan, menunggu, dan mati.

Pertanyaan Apa tipe argumen dari metode main() program?

Jawaban Metode main() program mengambil argumen dari tipe String [].

Pertanyaan Apa yang memanggil metode run() thread?

Jawaban Setelah utas dimulai, melalui metode start() atau kelas Thread, JVM memanggil metode run() utas saat utas pertama kali dijalankan.

Pertanyaan Apa urutan prioritas dan asosiativitas, dan bagaimana penggunaannya?

Jawaban Urutan prioritas menentukan urutan di mana operator dievaluasi dalam ekspresi. Keterkaitan menentukan apakah ekspresi dievaluasi dari kiri ke kanan atau kanan ke kiri.

Pertanyaan Apa itu clipping?

Jawaban Clipping adalah proses membatasi operasi cat pada area atau bentuk tertentu. Karakter mana yang dapat digunakan sebagai karakter kedua dari sebuah pengenal, tapi tidak sebagai yang pertama Digit 0 sampai 9 tidak dapat digunakan sebagai karakter pertama dari sebuah pengenal tetapi mereka dapat digunakan setelah karakter pertama dari sebuah pengenal

Pertanyaan Apa itu CTS (Common Type System)?

Jawaban Ini mendefinisikan tentang bagaimana Objek harus dideklarasikan, didefinisikan dan digunakan dalam .NET. CLS adalah bagian dari CTS.

Pertanyaan Apa itu kebijakan ?.

Jawaban Ini adalah kelas abstrak untuk mewakili kebijakan keamanan sistem untuk lingkungan aplikasi Java (menentukan izin yang tersedia untuk kode dari berbagai sumber). File properti keamanan Java berada di direktori /lib/security/java.security. Nilai "policy.provider" harus diubah.

Pertanyaan Apa batasan yang diberlakukan oleh Manajer Keamanan pada Applet ?.

Jawaban

Tidak dapat membaca atau menulis file di host yang menjalankannya.
Tidak dapat memuat pustaka atau menentukan metode asli.
Tidak dapat membuat koneksi jaringan kecuali ke host asalnya
Tidak dapat memulai program apa pun di host yang sedang menjalankannya.
Tidak dapat membaca properti sistem tertentu.
Jendela yang ditampilkan applet berbeda dengan jendela yang ditampilkan aplikasi.

Pertanyaan Lingkup dan masa pakai variabel?

Jawaban Lingkup variabel hanya untuk blokwaktu tertentu sampai blok berakhir. Variabel yang dideklarasikan di atas blok dalam kelas juga berlaku untuk blok dalam itu.

Pertanyaan Apa itu kelas Object dan java.lang?

Jawaban Kelas Objek adalah superclass dari semua kelas dan berarti bahwa variabel referensi dari objek tipe dapat merujuk ke objek dari kelas lain. dan juga mendefinisikan metode seperti finalize, wait.java.lang berisi semua fungsi bahasa dasar dan diimpor ke semua program secara implisit.

Pertanyaan Apa itu paket dan mengapa? bagaimana 2 menjalankan program dalam sebuah paket?

Jawaban Package adalah sekumpulan kelas, yang dapat diakses sendiri dan tidak dapat diakses di luar paket. dan dapat didefinisikan sebagai paket. Nama paket dan nama direktori harus sama. Dan eksekusi program dalam paket dilakukan dengan: java mypack.account dimana mypack adalah nama direktori dan akun adalah nama program.

Pertanyaan Apa itu StringTokenizer?

Jawaban String Tokenizer menyediakan proses penguraian di mana ia mengidentifikasi pembatas yang disediakan oleh pengguna, secara default pembatas adalah spasi, tab, baris baru, dll. Dan memisahkannya dari token. Token adalah yang dipisahkan oleh pembatas.

Pertanyaan Apa itu kelas bertingkat?

Jawaban Ada dua jenis - statis dan non-statis. kelas statis berarti anggota dalam kelas penutupnya (inner class kelas) dapat diakses dengan membuat objek dan tidak dapat diakses secara langsung tanpa membuat objek. kelas non-statis berarti inner class dan dapat diakses langsung dengan objek yang dibuat untuk kelas luar tidak perlu membuat lagi objek seperti kelas statis.

Pertanyaan Apa itu metode dan bagaimana mereka didefinisikan?

Jawaban Metode adalah fungsi yang beroperasi pada contoh kelas di mana mereka didefinisikan. Objek dapat berkomunikasi satu sama lain menggunakan metode dan dapat memanggil metode di kelas lain. Definisi metode memiliki empat bagian.

Mereka adalah nama metode, tipe objek atau tipe primitif yang dikembalikan metode, daftar parameter dan isi metode. Tanda tangan metode adalah kombinasi dari tiga bagian pertama yang disebutkan di atas.

Pertanyaan Apa itu Garbage Collection dan bagaimana menyebutnya secara eksplisit?

Jawaban Ketika sebuah objek tidak lagi dirujuk oleh variabel apapun, java secara otomatis mengambil kembali memori yang digunakan oleh objek tersebut. Ini dikenal sebagai metode Garbage Collection.System.gc() dapat digunakan untuk memanggilnya secara eksplisit.

Pertanyaan Apa itu antarmuka dan penggunaannya?

JawabanInterface mirip dengan kelas yang mungkin hanya berisi tanda tangan metode tetapi bukan badan dan merupakan sekumpulan metode formal dan deklarasi konstan yang harus ditentukan oleh kelas yang mengimplementasikannya. Antarmuka berguna untuk:

Mendeklarasikan metode yang satu atau lebih kelas diharapkan untuk mengimplementasikan Menangkap kesamaan antara kelas yang tidak terkait tanpa memaksa hubungan kelas. Menentukan antarmuka pemrograman objek tanpa mengungkapkan isi kelas yang sebenarnya.

Pertanyaan Apa perbedaan antara kelas abstrak dan antarmuka?

Jawaban

Semua metode yang dideklarasikan di dalam antarmuka bersifat abstrak sedangkan kelas abstrak harus memiliki setidaknya satu metode abstrak dan yang lainnya mungkin konkret atau abstrak.

Di kelas abstrak, keyword abstrak harus digunakan untuk metode. Sedangkan antarmuka kita tidak perlu menggunakan keyword itu untuk metode. c) Kelas abstrak harus memiliki subkelas sedangkan antarmuka tidak boleh memiliki subkelas.

Pertanyaan Apa perbedaan antara pengecualian dan kesalahan?

Jawaban Kelas pengecualian mendefinisikan kondisi kesalahan ringan yang ditemui program Anda.

Contoh: Pengecualian Aritmatika, Pengecualian FilenotFound dapat terjadi saat mencoba membuka file, yang tidak ada, koneksi jaringan terganggu, operan yang dimanipulasi berada di luar kisaran yang ditentukan, file kelas yang ingin Anda muat hilang Kelas kesalahan mendefinisikan kondisi kesalahan serius yang tidak boleh Anda coba pulihkan. Dalam kebanyakan kasus, disarankan untuk membiarkan program berhenti ketika kesalahan seperti itu ditemui. Contoh: Kesalahan kehabisan memori, kesalahan Stack overflow.

Pertanyaan Apa perbedaan antara aplikasi dan applet?

Jawaban

Aplikasi harus dijalankan di mesin lokal sedangkan applet tidak memerlukan instalasi eksplisit di mesin lokal.

Aplikasi harus dijalankan secara eksplisit dalam mesin virtual yang kompatibel dengan java sementara applet memuat dan berjalan sendiri secara otomatis di browser yang mendukung java.

Aplikasi memulai eksekusi dengan metode utamanya sedangkan applet memulai eksekusi dengan metode initnya.

Aplikasi dapat berjalan dengan atau tanpa antarmuka pengguna grafis sedangkan applet harus berjalan dalam antarmuka pengguna grafis.

Pertanyaan Apakah peristiwa itu dan apa model yang tersedia untuk penanganan peristiwa?

Jawaban Suatu peristiwa adalah suatu objek peristiwa yang menggambarkan suatu keadaan perubahan dalam suatu sumber. Dengan kata lain, peristiwa terjadi ketika suatu tindakan dibuat, seperti menekan tombol, mengklik mouse, memilih daftar, dll. Ada dua jenis model untuk menangani peristiwa dan mereka adalah: a) model peinheritance peristiwa dan b) pendelegasian peristiwa model

Pertanyaan Apa itu kelas adaptor?

Jawaban Kelas adaptor menyediakan implementasi kosong dari semua metode dalam antarmuka pendengar peristiwa. Kelas adaptor berguna saat Anda ingin menerima dan memproses hanya beberapa kejadian yang ditangani oleh antarmuka pendengar kejadian tertentu. Anda dapat menentukan kelas baru untuk bertindak listener dengan memperluas salah satu kelas adaptor dan hanya mengimplementasikan kejadian yang Anda minati.

Pertanyaan Apa yang dimaksud dengan kontrol dan apa saja jenis kontrol yang berbeda di AWT?

Jawaban Kontrol adalah komponen yang memungkinkan pengguna untuk berinteraksi dengan aplikasi Anda dan AWT mendukung jenis kontrol berikut: Label, Tombol Tekan, Kotak Centang, Daftar Pilihan, Daftar, Scrollbar, Komponen Teks. Kontrol ini adalah subclass dari Komponen.

Pertanyaan Apa perbedaan antara pilihan dan daftar?

Jawaban Pilihan A ditampilkan dalam bentuk kompak yang mengharuskan Anda untuk menariknya ke bawah untuk melihat daftar pilihan yang tersedia dan hanya satu item dapat dipilih dari pilihan. Daftar dapat ditampilkan sedemikian rupa sehingga beberapa item daftar terlihat dan ini mendukung pemilihan satu atau lebih item daftar.

Pertanyaan Apa itu paket Java dan bagaimana cara menggunakannya?

Jawaban Paket Java adalah konteks penamaan untuk kelas dan antarmuka. Paket digunakan untuk membuat ruang nama terpisah untuk grup kelas dan antarmuka. Paket juga digunakan untuk mengatur kelas dan antarmuka terkait ke dalam satu unit API dan untuk mengontrol aksesibilitas ke kelas dan antarmuka ini.

Pertanyaan Apa perbedaan antara bentuk prefiks dan postfix pada operator++?

Jawaban Formulir awalan melakukan operasi kenaikan dan mengembalikan nilai operasi kenaikan. Formulir postfix mengembalikan nilai saat ini semua ekspresi dan kemudian melakukan operasi penambahan pada nilai itu.

Pertanyaan Apa itu promosi numerik?

Jawaban Numeric promotion adalah konversi dari jenis numerik yang lebih kecil ke jenis numerik yang lebih besar, sehingga operasi integer dan floating-point dapat berlangsung. Dalam promosi numerik, nilai byte, char, dan short diubah menjadi nilai int. Nilai int juga diubah menjadi nilai panjang, jika perlu. Nilai long dan float diubah menjadi nilai ganda, sesuai kebutuhan.

Pertanyaan Apa tiga cara di mana utas dapat memasuki status menunggu?

Jawaban Sebuah thread bisa memasuki kondisi menunggu dengan memanggil metode sleep(), dengan memblokir di I/O, dengan tidak berhasil mencoba mendapatkan kunci objek, atau dengan memanggil metode wait() objek. Itu juga bisa memasuki keadaan menunggu dengan memanggil metode suspend() nya (deprecated).

Pertanyaan Apa yang terjadi jika pengecualian tidak tertangkap?

Jawaban Hasil pengecualian yang tidak tertangkap dalam metode uncaughtException() dari ThreadGroup utas sedang dipanggil, yang pada akhirnya mengakibatkan penghentian program tempat ia dilempar.

Pertanyaan Batasan apa yang diterapkan pada pengabaian metode?

JawabanMetode yang diganti harus memiliki nama yang sama, daftar argumen, dan tipe kembalian. Metode penggantian mungkin tidak membatasi akses metode yang ditimpanya. Metode penggantian tidak boleh menampilkan pengecualian apa pun yang mungkin tidak muncul oleh metode yang diganti.

Pertanyaan Apa pengkodean karakter default platform Anda?

Jawaban Jika Anda menjalankan Java pada platform Windows Inggris, kemungkinan Cp1252. Jika Anda menjalankan Java pada platform Solaris bahasa Inggris, kemungkinan besar adalah 8859_1.

Pertanyaan Apa perbedaan antara kelas File dan RandomAccessFile?

Jawaban Kelas File mengenkapsulasi file dan direktori dari sistem file lokal. Kelas RandomAccessFile menyediakan metode yang diperlukan untuk mengakses data secara langsung yang terdapat di bagian mana pun dari file.

Pertanyaan Apa tujuan dari metode enableEvents()?

Jawaban Metode enableEvents() digunakan untuk mengaktifkan acara untuk objek tertentu. Biasanya, acara diaktifkan ketika pendengar ditambahkan ke objek untuk acara tertentu. Metode enableEvents() digunakan oleh objek yang menangani peristiwa dengan menimpa metode pengiriman peristiwa mereka.

Pertanyaan Apa perbedaan antara model peristiwa JDK 1.02 dan delegasi peristiwa?

Jawaban Model peristiwa JDK 1.02 menggunakan peinheritance peristiwa atau pendekatan menggelegak. Dalam model ini, komponen diperlukan untuk menangani kejadiannya sendiri. Jika mereka tidak menangani acara tertentu, acara tersebut diwarisi oleh (atau dialihkan ke) wadah komponen. Kontainer kemudian akan menangani acara tersebut atau itu digelembungkan ke wadahnya dan seterusnya, sampai wadah tingkat tertinggi telah dicoba. Dalam model event-delegation, objek tertentu ditunjuk sebagai event handler untuk komponen GUI. Objek-objek ini mengimplementasikan antarmuka event-listener. Model event-delegation lebih efisien daripada model event-inheritance karena menghilangkan pemrosesan yang diperlukan untuk mendukung penggelembungan acara yang tidak tertangani.

Pertanyaan Apa perbedaan antara Choice dan List?

Jawaban A Choice ditampilkan dalam bentuk ringkas yang mengharuskan Anda menariknya ke bawah untuk melihat daftar pilihan yang tersedia. Hanya satu item yang dapat dipilih dari sebuah Pilihan. Daftar dapat ditampilkan sedemikian rupa sehingga beberapa item Daftar terlihat. Daftar mendukung pemilihan satu atau lebih item Daftar.

Pertanyaan Apa itu casting?

Jawaban Ada dua jenis pengecoran, pengecoran antara tipe numerik primitif dan pengecoran antar referensi objek. Transmisi antar tipe numerik digunakan untuk mengonversi nilai yang lebih besar, seperti nilai ganda, menjadi nilai yang lebih kecil, seperti nilai byte. Transmisi antar referensi objek digunakan untuk merujuk ke objek dengan kelas yang kompatibel, antarmuka, atau referensi tipe array.

Pertanyaan Kapan klausa akhirnya dari pernyataan coba-tangkap-akhirnya dieksekusi?

Jawaban Klausa akhirnya dari pernyataan coba-tangkap-akhirnya selalu dijalankan kecuali utas eksekusi dihentikan atau pengecualian terjadi dalam eksekusi klausa akhirnya.

Pertanyaan Bagaimana multithreading terjadi di komputer dengan satu CPU?

Jawaban Penjadwal tugas sistem operasi mengalokasikan waktu eksekusi untuk beberapa tugas. Dengan beralih cepat di antara menjalankan tugas, ini menciptakan kesan bahwa tugas dijalankan secara berurutan.

Pertanyaan Apa perbedaan antara variabel statis dan non-statis?

Jawaban Variabel statis diasosiasikan dengan kelas secara keseluruhan daripada dengan instance kelas tertentu. Variabel non-statis mengambil nilai unik dengan setiap instance objek.

Pertanyaan Apa keuntungan yang diberikan pengelola tata letak Java dibandingkan sistem jendela tradisional?

Jawaban Java menggunakan manajer tata letak untuk meletakkan komponen secara konsisten di semua platform windowing. Karena pengelola tata letak Java tidak terikat dengan ukuran dan pemosisian absolut, mereka dapat mengakomodasi perbedaan khusus platform di antara sistem windowing.

Pertanyaan Bagaimana elemen GridBagLayout diatur?

Jawaban Elemen GridBagLayout diatur menurut grid. Namun, elemen memiliki ukuran berbeda dan dapat menempati lebih dari satu baris atau kolom kisi. Selain itu, baris dan kolom mungkin memiliki ukuran yang berbeda.

Pertanyaan Apa perbedaan antara pernyataan while dan pernyataan do?

Jawaban memeriksa di awal pengulangan untuk melihat apakah pengulangan berikutnya harus terjadi. Pernyataan do memeriksa di akhir pengulangan untuk melihat apakah pengulangan berikutnya dari pengulangan harus terjadi. Pernyataan do akan selalu mengeksekusi badan loop setidaknya sekali.

Pertanyaan Apa hubungan antara antarmuka pendengar peristiwa dan adaptor peristiwa?

Jawaban Antarmuka event-listener mendefinisikan metode yang harus diimplementasikan oleh event handler untuk jenis event tertentu. Adaptor peristiwa menyediakan implementasi default dari antarmuka pemroses peristiwa.

Pertanyaan Jika sebuah kelas dideklarasikan tanpa pengubah akses, di mana kelas tersebut dapat diakses?

JawabanSebuah kelas yang dideklarasikan tanpa pengubah akses dikatakan memiliki akses paket. Ini berarti bahwa kelas tersebut hanya dapat diakses oleh kelas dan antarmuka lain yang ditentukan dalam paket yang sama.

Pertanyaan Kelas pengecualian apa yang mungkin ditangkap oleh klausa catch?

Jawaban Sebuah klausa catch dapat menangkap pengecualian apapun yang mungkin ditugaskan ke tipe Throwable. Ini termasuk jenis Kesalahan dan Pengecualian.

Pertanyaan Apa yang terjadi jika utas tidak bisa mendapatkan kunci pada suatu objek?

Jawaban Jika utas mencoba untuk menjalankan metode tersinkronisasi atau pernyataan tersinkronisasi dan tidak dapat memperoleh kunci objek, itu memasuki status menunggu sampai kunci tersedia.

Pertanyaan Apa perbedaan antara kelas Font dan FontMetrics?

Jawaban Kelas FontMetrics digunakan untuk mendefinisikan properti khusus implementasi, seperti naik dan turun, dari objek Font.

Pertanyaan Apa perbedaan antara Jendela dan Bingkai?

Jawaban Kelas Frame memperluas Window untuk menentukan jendela aplikasi utama yang dapat memiliki menu bar.

Pertanyaan Apa operator% itu?

Jawaban Ini disebut sebagai modulo atau operator sisa. Ini mengembalikan sisa pembagian operan pertama dengan operan kedua.

Pertanyaan Apakah kunci sebuah benda dan benda mana yang memiliki kunci?

Jawaban Kunci objek adalah mekanisme yang digunakan oleh beberapa utas untuk mendapatkan akses yang disinkronkan ke objek. Sebuah utas dapat menjalankan metode tersinkronisasi dari sebuah objek hanya setelah ia memperoleh kunci objek tersebut. Semua objek dan kelas memiliki kunci. Kunci kelas diperoleh pada objek Kelas kelas.

Pertanyaan Apa perbedaan antara inner class statis dan non-statis?

JawabanInner class non-statis mungkin memiliki contoh objek yang terkait dengan contoh kelas luar kelas. Inner class statis tidak memiliki instance objek apa pun.

Pertanyaan Bagaimana nama file kode sumber Java?

Jawaban File kode sumber Java menggunakan nama kelas atau antarmuka publik yang ditentukan di dalam file. File kode sumber dapat berisi paling banyak satu kelas atau antarmuka publik.

Jika kelas atau antarmuka publik didefinisikan dalam file kode sumber, maka file kode sumber harus menggunakan nama kelas atau antarmuka publik. Jika tidak ada kelas atau antarmuka publik yang ditentukan dalam file kode sumber, maka file tersebut harus menggunakan nama yang berbeda dari kelas dan antarmukanya. File kode sumber menggunakan ekstensi .java.

Pertanyaan Apa tujuan dari metode wait(), notify(), dan notifyAll()?

Jawaban Metode wait(), notify(), dan notifyAll() digunakan untuk menyediakan cara yang efisien bagi thread untuk menunggu sumber daya bersama. Ketika sebuah thread menjalankan metode wait() objek, ia memasuki status menunggu. Ini hanya memasuki status siap setelah thread lain memanggil metode notify() atau notifyAll() dari objek.

Pertanyaan Apa keuntungan dari model event-delegation dibandingkan model eventinheritance sebelumnya?

Jawaban Model event-delegation memiliki dua keunggulan dibandingkan model event-inheritance. Pertama, ini memungkinkan penanganan peristiwa untuk ditangani oleh objek selain yang menghasilkan peristiwa (atau penampungnya). Ini memungkinkan pemisahan yang bersih antara desain komponen dan penggunaannya. Keuntungan lain dari model delegasi-peristiwa adalah model ini bekerja jauh lebih baik dalam aplikasi di mana banyak peristiwa dihasilkan. Peningkatan kinerja ini disebabkan oleh fakta bahwa model delegasi peristiwa tidak harus berulang kali memproses peristiwa yang tidak ditangani, seperti kasus model peinheritance peristiwa.

Pertanyaan Apa yang harus dilakukan kelas untuk mengimplementasikan antarmuka?

Jawaban Itu harus menyediakan semua metode dalam antarmuka dan mengidentifikasi antarmuka dalam klausul implementasinya.

Pertanyaan Apa perbedaan antara pernyataan break dan pernyataan lanjutan?

Jawaban Sebuah pernyataan istirahat menghasilkan penghentian pernyataan yang berlaku (switch, for, do, or while). Pernyataan lanjutkan digunakan untuk mengakhiri iterasi pengulangan saat ini dan mengembalikan kontrol ke pernyataan pengulangan.

Pertanyaan Apa itu kelas Lokal?

Jawaban Kelas Lokal digunakan untuk menyesuaikan keluaran program dengan konvensi wilayah geografis, politik, atau budaya tertentu.

Pertanyaan Apa tujuan dari klausa akhirnya dari pernyataan coba-tangkap-akhirnya?

Jawaban Klausa akhirnya digunakan untuk memberikan kemampuan untuk mengeksekusi kode tidak peduli apakah pengecualian dilempar atau ditangkap atau tidak.

Pertanyaan Apa tujuan dari kelas Runtime?

Jawaban Tujuan dari kelas Runtime adalah untuk menyediakan akses ke sistem runtime Java.

Pertanyaan Apa perbedaan antara Boolean & operator dan operator &&?

Jawaban Jika ekspresi yang melibatkan operator Boolean & dievaluasi, kedua operan dievaluasi. Kemudian operator & diterapkan ke operan. Saat ekspresi yang melibatkan operator && dievaluasi, operan pertama dievaluasi. Jika operan pertama mengembalikan nilai true maka operan kedua dievaluasi.

Operator && kemudian diterapkan ke operan pertama dan kedua. Jika operan pertama bernilai false, evaluasi operan kedua akan dilewati.

Pertanyaan Apa tujuan finalisasi?

Jawaban Tujuan finalisasi adalah untuk memberikan kesempatan kepada objek yang tidak terjangkau untuk melakukan pemrosesan pembersihan sebelum objek tersebut dikumpulkan sampahnya.

Pertanyaan Di paket manakah sebagian besar acara AWT yang mendukung delegasi acara?

Jawaban Sebagian besar peristiwa terkait AWT model delegasi peristiwa ditentukan dalam paket java.awt.event. Kelas AWTEvent didefinisikan dalam paket java.awt.

Pertanyaan Bisakah kelas anonim dideklarasikan sebagai menerapkan antarmuka dan memperluas kelas?

Jawaban Kelas anonim dapat mengimplementasikan antarmuka atau memperluas superclass, tetapi tidak dapat dideklarasikan untuk melakukan keduanya.

Pertanyaan Kelas apa yang merupakan puncak hierarki peristiwa AWT?

JawabanKelas java.awt.AWTEvent adalah kelas tingkat tertinggi dalam hierarki kelas acara AWT

Pertanyaan Apa itu prioritas tugas dan bagaimana itu digunakan dalam penjadwalan?

Jawaban Prioritas tugas adalah nilai integer yang mengidentifikasi urutan relatif yang harus dijalankan sehubungan dengan tugas lain. Penjadwal mencoba menjadwalkan tugas dengan prioritas lebih tinggi sebelum tugas dengan prioritas lebih rendah.

Pertanyaan Apa aturan catch atau decare untuk deklarasi metode?

Jawaban Jika pengecualian yang dicentang dapat dilemparkan ke dalam komponen metode, metode tersebut harus menangkap pengecualian tersebut atau mendeklarasikannya dalam klausa lemparannya.

Pertanyaan Apa urutan prioritas dan asosiativitas, dan bagaimana penggunaannya?

Jawaban Urutan prioritas menentukan urutan di mana operator dievaluasi dalam ekspresi. Keterkaitan menentukan apakah ekspresi dievaluasi dari kiri ke kanan atau kanan ke kiri.

Pertanyaan Apa perbedaan antara penjadwalan preemptive dan pembagian waktu?

Jawaban Di bawah penjadwalan preemptive, tugas prioritas tertinggi dijalankan hingga memasuki status menunggu atau mati atau tugas dengan prioritas lebih tinggi muncul. Di bawah pembagian waktu, tugas dijalankan untuk bagian waktu yang telah ditentukan dan kemudian memasukkan kembali kumpulan tugas yang siap. Penjadwal kemudian menentukan tugas mana yang harus dijalankan selanjutnya, berdasarkan prioritas dan faktor lainnya.

Pertanyaan Apakah Garbage Collection menjamin bahwa program tidak akan kehabisan memori?

Jawaban Garbage collection tidak menjamin bahwa program tidak akan kehabisan memori. Ada kemungkinan program menggunakan sumber daya memori lebih cepat daripada sampah yang dikumpulkan. Mungkin juga program membuat objek yang tidak tunduk pada Garbage Collection.

Pertanyaan Kelas dan antarmuka java.util manakah yang mendukung penanganan acara?

Jawaban Kelas EventObject dan pemrosesan acara antarmuka dukungan EventListener.

Pertanyaan Apa perbedaan antara menyerah dan tidur?

Jawaban Ketika sebuah tugas memanggil metode yield(), ia kembali ke status siap. Ketika sebuah tugas memanggil metode sleep(), ia kembali ke status menunggu.

Pertanyaan Berapa banyak bit yang digunakan untuk merepresentasikan karakter Unicode, ASCII, UTF-16, dan UTF-8?

Jawaban Unicode membutuhkan 16 bit dan ASCII membutuhkan 7 bit. Meskipun himpunan karakter ASCII hanya menggunakan 7 bit, biasanya direpresentasikan sebagai 8 bit. UTF-8 mewakili karakter yang menggunakan pola 8, 16, dan 18 bit. UTF-16 menggunakan pola bit 16-bit dan yang lebih besar.

Pertanyaan Apa itu kelas Vector?

Jawaban Kelas Vector menyediakan kemampuan untuk mengimplementasikan array objek yang dapat dikembangkan.

Pertanyaan Apa itu antarmuka Daftar?

Jawaban Antarmuka Daftar menyediakan dukungan untuk koleksi objek yang dipesan.

Pertanyaan Apa itu Collections API?

Jawaban The Collections API adalah sekumpulan kelas dan antarmuka yang mendukung operasi pada koleksi objek.

Pertanyaan Berapa ukuran komponen yang disukai?

Jawaban Ukuran yang disukai dari sebuah komponen adalah ukuran komponen minimum yang memungkinkan komponen untuk ditampilkan secara normal.

Pertanyaan Bisakah kunci diperoleh di kelas?

Jawaban Ya, kunci dapat diperoleh di kelas. Kunci ini diperoleh pada objek Kelas kelas

Pertanyaan Apa itu sinkronisasi dan mengapa itu penting?

Jawaban Sehubungan dengan multithreading, sinkronisasi adalah kemampuan untuk mengontrol akses beberapa utas ke sumber daya bersama. Tanpa sinkronisasi, satu utas dimungkinkan untuk memodifikasi objek bersama sementara utas lain sedang dalam proses menggunakan atau memperbarui nilai objek itu. Ini sering kali menyebabkan kesalahan yang signifikan.

Pertanyaan Bagaimana Observer dan Observable digunakan?

Jawaban Objek yang merupakan subclass dari kelas Observable memelihara daftar pengamat. Ketika objek Observable diperbarui, ia memanggil metode update() dari setiap pengamatnya untuk memberi tahu pengamat bahwa statusnya telah berubah. Antarmuka Observer diimplementasikan oleh objek yang mengamati objek Observable.

Pertanyaan apa itu variabel transien?

Jawaban Variabel transien adalah variabel yang tidak dapat diserialkan.

BAB 15
PERTANYAAN DAN JAWABAN WAWANCARA II

Advise Drill - Jika anda berpikir bahwa anda lemah dalam Pemrograman Java, Cobalah buku Java ini "Core Java Professional" Penulis Oleh Harry. H. Chaudhary.

Beli Edisi Paperback dari Amazon, Edisi PDF Digital (7,99 USD) dari Google Books& Google Play. (800-1000 Halaman Deep Core Java)

Pertanyaan :Bisakah ada kelas abstrak tanpa metode abstrak di dalamnya?

Jawaban : Ya

Pertanyaan : Bisakah sebuah antarmuka menjadi final? (Core Java)

Jawaban : Tidak

Pertanyaan : Bisakah sebuah antarmuka memiliki inner class? (Core Java)

Jawaban : Ya.

```
public interface abc {
static int i=0;
void dd();
class a1 {
a1() {
int j;
System.out.println("in interfia"); };
public static void main(String a1[]) {
System.out.println("in interfia"); } } }
```

Pertanyaan : Bisakah kita menetapkan pengubah privat dan perlindungan untuk variabel dalam antarmuka? (Core Java)

Jawaban : Tidak.

Pertanyaan : Apa kueri yang digunakan untuk menampilkan semua nama tabel di SQL Server (Query analyzer)? (JDBC)

Jawaban : pilih * dari information_schema.tables

Pertanyaan : Apa itu *Externalizable*? (Core Java)

Jawaban : Externalizable adalah antarmuka yang memperluas antarmuka Serializable. Dan mengirimkan data ke Aliran dalam Format Terkompresi. Ini memiliki dua metode, writeExternal (ObjectOuput out) dan readExternal (ObjectInput in)

Pertanyaan : Pengubah apa yang diizinkan untuk metode dalam Antarmuka?

Jawaban : Hanya pengubah publik dan abstrak yang diperbolehkan untuk metode antarmuka.

Pertanyaan : Apa itu variabel lokal, anggota dan kelas? (Core Java)

Jawaban : Variabel yang dideklarasikan dalam suatu metode adalah variabel "lokal".

Variabel yang dideklarasikan dalam kelas yaitu tidak dalam metode apa pun adalah variabel "anggota" (variabel global).Variabel yang dideklarasikan dalam kelas yaitu tidak dalam metode apa pun dan didefinisikan sebagai "statis" adalah variabel kelas.

Pertanyaan : Ada berapa jenis Driver JDBC dan apa saja?

Jawaban : Ada 4 tipe JDBC Drivers

Tipe 1 : Driver Bridge JDBC-ODBC.

Tipe 2 : Native API Partly Java Driver.

Jenis 3 : Driver protokol jaringan.

Tipe 4 : JDBC Net pure Java Driver.

Pertanyaan : Bisakah kita menerapkan antarmuka dalam JSP? (JSP)

Jawaban : Tidak

Pertanyaan : Apa perbedaan antara ServletContext dan PageContext?

Jawaban : ServletContext: Memberikan informasi tentang container PageContext: Memberikan informasi tentang Request

Pertanyaan : Apa perbedaan dalam menggunakan request.getRequestDispatcher() dan context.getRequestDispatcher()? (JSP)

Jawaban : request.getRequestDispatcher(path): Untuk membuatnya, kita perlu memberikan jalur relatif dari konteks sumber daya. GetRequestDispatcher(path): Untuk membuatnya, kita perlu memberikan jalur absolut sumber daya.

Pertanyaan : Bagaimana cara menyampaikan informasi dari JSP ke included JSP? (JSP)

Jawaban : Menggunakan <% jsp: param>tag.

Pertanyaan : Apa perbedaan antara directive include dan jsp include?

Jawaban : <% @ include>: Digunakan untuk menyertakan sumber daya statis selama waktu terjemahan. :Digunakan untuk memasukkan konten dinamis atau konten statis selama runtime.

Pertanyaan : Apa perbedaan antara RequestDispatcher dan sendRedirect?

Jawaban : RequestDispatcher: pengalihan sisi server dengan objek permintaan dan respons. sendRedirect: Pengalihan sisi klien dengan objek permintaan dan respons baru.

Pertanyaan : Bagaimana JSP menangani pengecualian waktu proses? (JSP)

Jawaban : Menggunakan atribut errorPage dari halaman direktif dan juga kita perlu menentukan isErrorPage=true jika halaman saat ini dimaksudkan untuk pengalihan URL dari JSP.

Pertanyaan : Bagaimana anda menghapus Cookie dalam JSP? (JSP)

Jawaban :

```
Cookie mycook = new Cookie("name", "value");
response.addCookie(mycook);
Cookie killmycook = new Cookie("mycook", "value");
killmycook.setMaxAge(0);
killmycook.setPath("/");
killmycook.addCookie(killmycook);
```

Pertanyaan : Bagaimana cara menggabungkan JSP dan SSI #include? (JSP)

Jawaban : Jika anda hanya memasukkan HTML mentah, gunakan perintah #include seperti biasa di dalam file .jsp anda.

<! - # include file = "data.inc" ->

Tetapi ini sedikit lebih rumit jika anda ingin server mengevaluasi kode JSP apa pun yang ada

di dalam file yang disertakan. Ronel Sumibcay

(ronel@LIVESOFTWARE.COM) mengatakan: Jika file data.inc berisi kode jsp, anda harus menggunakan –

<% @ vinclude = "data.inc"%>

<! - # include file = "data.inc" -> digunakan untuk menyertakan file non-JSP.

Pertanyaan : Saya membuat kelas saya Dapat Dikloning tetapi masih 'Tidak dapat mengakses metode pelidungklon. Mengapa? (Core Java)

Jawaban : Ya, beberapa buku Java, khususnya "Bahasa Pemrograman Java", menyiratkan bahwa hal yang harus anda lakukan agar kelas anda mendukung metode clone() adalah mengimplementasikan Cloneable interface.

Catatan : Bukan demikian, Itu hanya berlaku di beberapa titik, tapi saat ini cara kerjanya tidak seperti itu. Seperti yang sudah ada, anda harus menerapkan metode clone() publik anda sendiri, bahkan jika itu tidak melakukan sesuatu yang khusus dan hanya memanggil super.clone().

Pertanyaan : Mengapa XML merupakan perkembangan penting? (XML)

Jawaban : Karena menghilangkan dua kendala yang menghambat perkembangan Web:

Ketergantungan pada satu jenis dokumen yang tidak fleksibel (HTML) yang sering disalahgunakan untuk tugas-tugas yang tidak pernah dirancang untuk itu;

Kompleksitas SGML penuh, yang memungkinkan banyak sintaks yang kuat

Tetapi opsi yang sulit-diprogram. XML memungkinkan pengembangan fleksibelitas jenis dokumen yang ditentukan pengguna. Ini menyediakan format file yang kuat, tidak berpemilik, persisten, dan dapat diverifikasi untuk penyimpanan dan transmisi teks dan data baik di dalam maupun di luar Web; dan menghapus opsi SGML yang lebih kompleks, membuatnya lebih mudah untuk diprogram.

Pertanyaan : Apakah Enterprise beans diizinkan menggunakan Thread.sleep()? (EJB)

Jawaban : Enterprise beans memanfaatkan layanan yang disediakan oleh EJB container, seperti manajemen siklus hidup. Untuk menghindari konflik dengan layanan ini, Enterprise beans dilarang melakukan operasi tertentu: Mengelola atau menyinkronkan utas.

Pertanyaan : Apakah mungkin untuk menulis dua EJB yang berbagi interface Remote & Home yang sama, dan memiliki bean classes yang berbeda? Jika ya, apa keuntungan/ kerugiannya? (EJB)

Jawaban : Itu sangat mungkin. Faktanya, ada contoh yang dikirimkan dengan interface Pelayanan Akun Aplikasi Inprise dengan implementasi terpisah untuk CheckingAccount dan SavingsAccount, salah satunya adalah CMP dan salah satunya adalah BMP.

Pertanyaan : Apakah mungkin untuk menentukan beberapa nama JNDI saat menerapkan EJB?

Jawaban : Tidak. Untuk mencapai ini, anda harus menerapkan EJB beberapa kali dengan menentukan nama masing-masing JNDI yang berbeda.

Pertanyaan : Apakah ada cara untuk memaksa Entity Bean menyimpan dirinya sendiri ke db? Saya tidak ingin menunggu kontainer memperbarui db, saya ingin melakukannya SEKARANG! Apa itu mungkin? (EJB)

Jawaban : Tentukan atribut transaksi bean sebagai Requirement. Kemudian sesuai bagian 11.6.2.4 dari container EJB spek EJB v 1.1 secara otomatis memulai transaksi baru sebelum pemanggilan metode. Container juga melakukan protokol komit sebelum hasil metode dikirim ke klien.

Pertanyaan : Saya sedang mengembangkan BMP Entity bean. Saya mencatat bahwa setiap kali metode create dipanggil, metode ejbLoad() dan ejbStore() juga terpanggil. Saya merasa, setelah penyisipan database selesai, saya harus melakukan pemilihan dan perbaruan kueri SQL. Apakah inikebiasaan dari semua penyimpanan EJB? Apakah ada cara untuk menekan kebiasaan ini? (EJB)

Jawaban : Ini adalah perilaku default untuk EJB. Spesifikasi menyatakan bahwa ejbLoad() akan dipanggil sebelum tiap transaksi dan ejbStore() setelah tiap transaksi. Setiap Vendor memiliki pengoptimalan, yang merupakan hak milik untuk skenario ini.

Pertanyaan : Bisakah EJB mengirim notifikasi asynchronous ke kliennya? (EJB)

Jawaban : Asynchronous notification adalah lubang yang diketahui di versi pertama spesifikasi EJB. Solusi yang disarankan untuk hal ini adalah dengan menggunakan JMS, yang tersedia di server yang mendukung J2EE. Pilihan lainnya, tentu saja, adalah menggunakan thread sisi klien dan polling.Ini bukan solusi ideal, tetapi bisa diterapkan untuk banyak skenario.

Pertanyaan : Bagaimana cara mengakses EJB dari ASP? (EJB)

Jawaban : Anda dapat menggunakan Platform Java 2, Layanan Akses Klien Edisi Perusahaan(J2EETM CAS) COM Bridge 1.0, saat ini dapat diunduh dari http://developer.java.sun.com/developer/earlyAccess/j2eecas/

Pertanyaan : Apakah ada jaminan keunikan untuk entity beans? (EJB)

Jawaban : Tidak ada jaminan seperti itu. Server dapat membuat instance dari Entity Bean dasar yang sama (dengan PK yang sama) sebanyak yang diinginkannya. Namun, setiap instance dijamin memiliki nilai data terbaru, dan konsisten secara transaksional, sehingga keunikan tidak diperlukan. Hal ini memungkinkan server untuk menskalakan sistem untuk mendukung banyak utas, beberapa permintaan bersamaan, dan banyak host.

Pertanyaan : Bagaimana enam atribut transaksi dipetakan ke tingkat isolasi seperti "dirty read"? Apakah atribut seperti “Utility" mengunci pembaca lain sampai saya selesai memperbarui? (EJB)

Jawaban : Atribut Transaksi di EJB tidak dipetakan ke level Isolasi Transaksi yang digunakan di JDBC. Ini adalah kesalahpahaman yang umum. Atribut Transaksi menentukan penampung saat Transaksi harus dimulai, ditangguhkan (dijeda), dan

dilakukan di antara pemanggilan metode di Enterprise JavaBeans. Untuk detail lebih lanjut dan ringkasan Atribut Transaksi, lihat bagian 11.6 dari spesifikasi EJB 1.1.

Pertanyaan : Saya telah membuat referensi jarak jauh ke EJB di FirstServlet. Dapatkah saya meletakkan referensi dalam sesi servlet dan menggunakannya di SecondServlet? (EJB)

Jawaban : Ya. Klien EJB (dalam hal ini servlet anda) memperoleh referensi jarak jauh ke EJB dari Interface Home; referensi tersebut dapat diserialisasi dan dapat diteruskan dari servlet ke servlet. Jika ini adalah sesi bean, maka server EJB akan menganggap sesi servlet klien web anda sesuai dengan satu sesi EJB, yang biasanya (tetapi tidak selalu) yang anda inginkan.

Pertanyaan : Bisakah kunci utama dalam entity bean menjadi tipe primitif Java seperti int? (EJB)

Jawaban : Kunci utama tidak boleh tipe primitif - gunakan wrapper classes primitif sebagai gantinya. Misalnya, anda dapat menggunakan java.lang.Integer sebagai kelas kunci utama, tetapi tidak int (harus kelas, bukan primitif)

Pertanyaan : Apa yang baru dalam spesifikasi EJB 2.0? (EJB)

Jawaban : Berikut ini adalah fitur-fitur utama yang didukung pada EJB 2.0 * Integrasi EJB denganJMS * Message Driven Beans * Menerapkan metode Bisnis tambahan di Interface Home yang tidak spesifik untuk contoh bean. * EJB QL.

Pertanyaan : Berapa banyak jenis implementasi protokol yang dimiliki RMI? (RMI)

Jawaban : RMI memiliki setidaknya tiga implementasi protokol: Java Remote Method Protocol (JRMP), Internet Inter ORB Protocol (IIOP), dan Jini Extensible Remote Invocation (JERI).

Ini adalah alternatif, bukan bagian dari hal yang sama, Ketiganya memang merupakan protokol lapisan 6 bagi mereka yang masih menggunakan model referensi OSI.

Pertanyaan : Apa saja status pengenal yang berbeda dari sebuah Thread? (Core java)

Jawaban : Pengidentifikasi berbeda dari sebuah Thread adalah:

R - Thread yang berjalan atau dapat dijalankan.

S - Thread yang ditangguhkan.

CW - Thread menunggu pada variabel kondisi.

MW - Thread menunggu di kunci monitor.

MS - Thread ditangguhkan menunggu di kunci monitor.

Pertanyaan : Apa kebutuhan dari Remote dan Interface Home. Kenapa tidak bisa jadi satu? (EJB)

Jawaban :

Singkatnya, saya akan mengatakan bahwa alasan utamanya adalah karena ada pembagian peran dan tanggung jawab yang jelas antara dua Interface. Interface beranda adalah cara anda untuk berkomunikasi dengan Container, yaitu siapa yang bertanggung jawab untuk membuat, menemukan bahkan menghapus satu atau lebih bean.

Interface jarak jauh adalah tautan anda ke bean, yang memungkinkan anda mengakses semua metode dan anggotanya dari jarak jauh. Seperti yang anda tau, ada dua elemen

berbeda (container dan bean) dan anda memerlukan dua Interface berbeda untuk mengakses keduanya.

Pertanyaan : Apa perbedaan antara Java Beans dan EJB? (EJB)

Jawaban : Java Beans adalah objek sisi klien dan EJB adalah objek sisi server, dan keduanya memiliki tujuan pengembangan, siklus hidup, dan tujuan yang sama sekali berbeda.

Pertanyaan : Pertanyaan Berkenaan dengan Entity Beans, apa yang terjadi jika Server EJB dan Database saya mogok, apa yang akan terjadi pada perubahan yang belum disimpan? Apakah ada file log transaksi yang digunakan? (EJB)

Jawaban : Sebenarnya, jika server EJB anda crash, anda bahkan tidak akan dapat membuat koneksi ke server untuk melakukan pencarian bean, karena server tidak akan lagi mendengarkan pada port untuk permintaan pencarian JNDI yang masuk. Anda akan kehilangan semua data yang tidak disimpan sebelum crash. Di sinilah anda harus mulai mencari pengelompokan server EJB.

JawabanLain–

Setiap perubahan yang belum disimpan akan hilang saat Server EJB anda mogok. Jika database anda juga macet, maka semua perubahan yang disimpan juga hilang kecuali anda memiliki beberapa cadangan atau mekanisme pemulihan untuk pengambilan data. Jadi pertimbangkan replikasi database dan EJB Clustering untuk skenario seperti itu, meskipun kejadian seperti itu sangat jarang terjadi. Semua database memiliki konsep file log (misalnya oracle memiliki konsep redo log files). Jadi jika basis data macet maka mereka mulai mencari file log untuk melakukan semua pekerjaan yang tertunda. Tapi jika EJB macet, Ini tergantung pada penampung seberapa sering ia pasif atau seberapa sering menyegarkan data dengan Database.

Pertanyaan : Bisakah anda mengontrol kapan passivasi terjadi? (EJB)

Jawaban : Developer, menurut spesifikasinya, tidak bisa langsung mengontrol kapan passivasi terjadi. Meskipun untuk Sesi Stateful bean, container tidak dapat pasif secara instan dalam transaksi. Jadi menggunakan transaksi bisa menjadi strategi untuk mengontrol passivasi. Metode ejbPassivate() dipanggil selama passivasi, sehingga pengembang memiliki kendali atas apa yang harus dilakukan selama latihan ini dan dapat mengimplementasikan logika yang membutuhkan optimasi. Beberapa penampung EJB, seperti BEA WebLogic, memberikan kemampuan untuk menyesuaikan penampung untuk meminimalkan panggilan pasif. Diambil dari WebLogic 6.0

DTD -

"Passivation-strategy dapat berupa" default "atau" transaction ". Dengan pengaturan default, container akan mencoba untuk menyimpan sekumpulan bean yang berfungsi di cache. Dengan pengaturan" transaksi ", container akan mem-passivasi bean setelah setiap transaksi (atau panggilan metode untuk permintaan non-transaksional). "

Pertanyaan : Apakah RMI-IIOP mendukung pengunduhan kelas secara dinamis? (RMI)

Jawaban : Tidak, RMI-IIOP tidak mendukung pengunduhan dinamis kelas seperti yang dilakukan dengan CORBA di DII (Dynamic Interface Invocation). Sebenarnya RMI-IIOP menggabungkan kegunaan Java Remote Method Invocation (RMI) dengan interoperabilitas Internet InterORB Protocol (IIOP). Jadi untuk mencapai interoperabilitas antara RMI dan CORBA, memerlukan beberapa fitur yang didukung oleh RMI tetapi bukan CORBA dan sebaliknya dihilangkan dari spesifikasi RMI-IIOP.

Pertanyaan : Apakah dukungan EJB 1.1 mengamanatkan dukungan untuk RMI-IIOP? Apa yang dimaksud dengan "API klien harus mendukung model pemrograman Java RMI-IIOP agar mudah dibawa, tetapi protokol yang mendasarinya dapat berupa apa saja"? (EJB)

Jawaban : EJB1.1 tidak mengamanatkan dukungan dari RMI-IIOP. OKE, untuk menjawab pertanyaan yang kedua:

Ada 2 jenis implementasi yang mungkin disediakan oleh Server EJB: Server EJB berbasis CORBA dan Server EJB Proprietri. Keduanya mendukung API RMI-IIOP tetapi bagaimana API tersebut diimplementasikan adalah cerita yang berbeda. (NB: Yang kami maksud dengan API adalah interface yang disediakan untuk klien oleh rintisan atau proxy). Server EJB berbasis CORBA benar-benar mengimplementasikan Objek EJB sebagai Objek CORBA (oleh karena itu menggabungkan ORB dan ini berarti bahwa EJB dapat dihubungi oleh klien CORBA (serta klien RMI-IIOP)

Kepemilikan EJB masih mengimplementasikan API RMI-IIOP (dalam rintisan klien) tetapi protokol yang mendasari dapat berupa apa saja. Oleh karena itu EJB anda TIDAK DAPAT dihubungi oleh klien CORBA. Perbedaannya adalah bahwa dalam kedua kasus, klien anda melihat API yang sama (karenanya, portabilitas klien anda) TETAPI bagaimana stub berkomunikasi dengan serverberbeda.

Pertanyaan :Spesifikasi EJB mengatakan bahwa kami tidak dapat menggunakan Transaksi yang Dikelola Bean di Entity Beans. Mengapa? (EJB)

Jawaban : Jawaban singkat dan praktis adalah ... karena membuat entity Bean tidak berfungsi sebagai komponen yang dapat digunakan kembali. Selain itu, manajemen transaksi sebaiknya diserahkan ke server aplikasi - untuk itulah mereka ada di sana. Ini semua tentang operasi atom pada data anda. Jika sebuah operasi memperbarui lebih dari satu entitas maka semuanya berhasil atau gagal, tidak ada di antaranya. Jika anda meletakkan komit di entity bean maka sangat sulit untuk melakukan rollback jika kesalahan terjadi di beberapa titik di akhir operasi.

Pertanyaan : Dapatkah saya menjalankan Runtime.gc() di EJB? (EJB)

Jawaban : Seharusnya tidak. Apa yang akan terjadi bergantung pada implementasinya, tetapi panggilan tersebut kemungkinan besar akan diabaikan. Anda harus membiarkan pengelolaan tingkat sistem seperti Garbage Collection untuk ditangani oleh penampung. Bagaimanapun, itu adalah bagian dari manfaat penggunaan EJB, anda tidak perlu mengelola sumber daya sendiri.

Pertanyaan : Apa itu clustering? Apa saja algoritme berbeda yang digunakan untuk clustering? (EJB)

Jawaban : Clustering adalah pengelompokan mesin bersama-sama secara transparan dengan menyediakan layanan perusahaan. Klien sekarang tidak membedakan antara mendekati satu server atau mendekati sekelompok server. Cluster memberikan dua manfaat: skalabilitas dan ketersediaan tinggi. Informasi lebih lanjut dapat ditemukan di artikel JavaWorld J2EE Clustering.

Pertanyaan : Apa keuntungan menggunakan Entity Bean untuk operasi database, dibandingkan secara langsung menggunakan JDBC API untuk melakukan operasi database? Kapan saya akan menggunakannya? (EJB)

Jawaban : Entity Beans sebenarnya merepresentasikan data dalam database. Entity Beans tidak menggantikan JDBC API. Ada dua jenis Entity Beans Container Managed dan Bean Mananged. Dalam Container Managed Entity Bean - Setiap kali instance bean dibuat, container secara otomatis mengambil data dari penyimpanan DB/Persistance dan menetapkan ke variabel objek dalam bean agar pengguna dapat memanipulasi atau menggunakannya.

Untuk ini, pengembang perlu memetakan bidang dalam database ke variabel dalam file deskriptor penerapan (yang bervariasi untuk setiap vendor).

Di Bean Managed Entity Bean - Pengembang harus secara khusus membuat koneksi,mengambil nilai, menetapkannya ke objek di ejbLoad() yang akan dipanggil oleh container saat instatiate objek bean. Demikian pula di ejbStore() container menyimpan nilai objek kembali ke penyimpanan persistensi.

ejbLoad dan ejbStore adalah metode callback dan hanya bisa dipanggil oleh container.

Selain itu, saat anda menggunakan entity bean, anda tidak perlu khawatir tentang penanganan transaksi database, penyatuan koneksi database, dll.yang diurus oleh container ejb. Tetapi dalam kasus JDBC anda harus secara eksplisit melakukan fitur-fitur di atas. tentu saja, ini berada di bawah transisi database, tetapi saya ingin menambahkan ini. Hal yang hebat tentang entity bean dari pengelolaan container, setiap kali koneksi gagal selama pemrosesan transaksi, konsistensi database dijaga secara otomatis.

Penampung menulis data yang disimpan di penyimpanan persisten dari entity bean ke database lagi untuk memberikan konsistensi database. sedangkan di jdbc api, kami, pengembang harus melakukannya secara manual.

Pertanyaan : Apa peran serialisasi di EJB? (EJB)

Jawaban : Sebagian besar EJB adalah kerangka kerja untuk RMI yang mendasari: pemanggilan metode jarak jauh. Anda menjalankan metode dari jarak jauh dari ruang JVM 'A' pada objek yang berada di ruang JVM 'B' - mungkin berjalan di komputer lain di jaringan.

Untuk mewujudkan hal ini, semua argumen dari setiap panggilan metode harus memiliki status saat ini yang dicabut dari memori JVM 'A', diratakan menjadi aliran byte yang dapat dikirim melalui koneksi jaringan TCP/IP, dan kemudian dideserialisasi untuk dihidupkan kembali di sisi lain. diakhiri dengan JVM 'B' di mana pemanggilan

metode sebenarnya dilakukan. Jika metode memiliki value back, itu serial untuk streaming kembali ke JVM A. Jadi persyaratan bahwa semua argumen metode EJB dan value back harus serializable. Cara termudah untuk melakukannya adalah dengan memastikan semua kelas anda mengimplementasikan java.io.Serializable.

Pertanyaan : Apa itu EJB QL? (EJB)

Jawaban : EJB QL adalah Bahasa Kueri yang disediakan untuk navigasi di seluruh jaringan enterprise bean dan objek dependen yang ditentukan melalui persistensi yang dikelola container. EJB QL diperkenalkan dalam spesifikasi EJB 2.0.

Bahasa kueri QL EJB mendefinisikan metode finder untuk entity bean dengan persistensi yang dikelola container dan portabel di seluruh container dan manajer persistensi. EJB QL digunakan untuk query dari dua jenis metode finder: metode Finder yang didefinisikan di interface entity home bean dan yang mengembalikan objek entitas.

Pilih metode, yang tidak diekspos ke klien, tetapi yang digunakan oleh Penyedia Bean untuk memilih nilai persisten pada objek entitas yang terkait dengan entity bean di mana kueri ditentukan.

Pertanyaan : Apa jenis driver JDBC tercepat? (JDBC)

Jawaban : Kinerja driver JDBC akan bergantung pada sejumlah masalah:

(a) Kualitas kode driver, (b) ukuran kode driver,

(c) Server database dan bebannya, (d) topologi jaringan,

(e) Berapa kali permintaan anda diterjemahkan ke API yang berbeda. Secara umum, semuanya dianggap sama, anda dapat berasumsi bahwa semakin banyak permintaan dan respons berpindah tangan, semakin lambat jadinya. Ini berarti driver Tipe 1 dan Tipe 3 akan lebih lambat daripada driver Tipe 2 (panggilan database membuat setidaknya tiga terjemahan versus dua), dan driver Tipe 4 adalah yang tercepat (hanya satu terjemahan).

Pertanyaan : Parameter permintaan. Bagaimana menemukan apakah ada parameter dalam objek permintaan? (Client Service)

Jawaban :

1.boolean hasFoo =! (Request.getParameter ("foo") ==
null || request.getParameter ("foo"). sama dengan (""));
2. boolean hasParameter =
request.getParameterMap(). contains (theParameter);
(yang bekerja di Servlet 2.3+)

Pertanyaan : Bagaimana saya dapat mengirim informasi otentikasi pengguna saat membuatURLConnection? (Client Service)

Jawaban : Anda akan ingin menggunakan HttpURLConnection.setRequestProperty dan menyetel semua header yang sesuai ke otorisasi HTTP.

Pertanyaan : Apa sajakah alternatif untuk inheritance ? (Core Java)

Jawaban : Delegasi adalah alternatif inheritance. Delegasi berarti anda menyertakan instance dari kelas lain sebagai variabel instance, dan meneruskan pesan ke instance tersebut.

Seringkali lebih aman daripada inheritance karena memaksa anda untuk memikirkan setiap pesan yang anda teruskan, karena instance dari kelas yang diketahui, bukan kelas baru, dan karena itu tidak memaksa untuk menerima semua metode kelas super : anda hanya dapat memberikan metode yang benar-benar masuk akal. Di sisi lain, ini membuat anda menulis lebih banyak kode, dan lebih sulit untuk digunakan kembali (karena ini bukan subclass).

Pertanyaan : Mengapa tidak ada operator yang kelebihan beban? (Core Java)

Jawaban : Karena C++ telah membuktikan dengan contoh bahwa kelebihan beban operator membuat kode hampir tidak mungkin untuk dipertahankan. Nyatanya, hampir tidak ada metode overloading di Java, tetapi dianggap terlalu berguna untuk beberapa metode yang sangat dasar seperti print(). Perhatikan bahwa beberapa class seperti DataOutputStream memiliki metode unoverload seperti writeInt() dan writeByte().

Pertanyaan : Apa yang dimaksud dengan metode atau kolom "statis"? (Core Java)

Jawaban : Variabel dan metode statis dibuat hanya sekali per kelas. Dengan kata lain mereka adalah variabel kelas, bukan variabel instan. Jika anda mengubah nilai variabel statis di objek tertentu, nilai variabel itu berubah untuk semua instance kelas itu. Metode statis dapat direferensikan dengan nama kelas daripada nama objek tertentu dari kelas tersebut (meskipun itu juga berfungsi). Begitulah cara metode perpustakaan seperti System.out.println() bekerja. Itulah bidang statis di kelas java.lang.System.

Pertanyaan : Bagaimana cara mengubah alamat IP numerik seperti 192.18.97.39 menjadi nama host seperti java.sun.com? (Network)

Jawaban : String hostname = InetAddress.getByName ("192.18.97.39"). GetHostName();

Pertanyaan : Perbedaan antara JRE dan JVM AND JDK (Core Java)

Jawaban : Baca Di Atas pada Pendahuluan di Awal bab ini.

Pertanyaan : Mengapa utas memblokir I/O? (Core Java)

Jawaban : Thread blok pada i/o (yang memasuki status menunggu) sehingga thread lain dapat dieksekusi saat Operasi i/o dilakukan.

Pertanyaan : Apa itu sinkronisasi dan mengapa itu penting? (Core Java)

Jawaban : Sehubungan dengan multithreading, sinkronisasi adalah kemampuan untuk mengontrol akses beberapa utas ke sumber daya bersama. Tanpa sinkronisasi, ada kemungkinan satu utas memodifikasi objek bersama sementara utas lain sedang dalam proses menggunakan atau memperbarui nilai objek itu. Ini sering kali menyebabkan kesalahan yang signifikan.

Pertanyaan : Apakah null sebuah keyword? (Core Java)

Jawaban : Nilai null bukanlah keyword.

Pertanyaan : Karakter mana yang dapat digunakan sebagai karakter kedua dari sebuah identifier, tapi tidak sebagai karakter pertama dari sebuah identifier? (Core Java)

Jawaban : Angka 0 sampai 9 tidak boleh digunakan sebagai karakter pertama identifier, tetapi bisa digunakan setelah karakter pertama identifier.

Pertanyaan : Pengubah apa yang dapat digunakan dengan inner class yang merupakan anggota kelas luar? (Core Java)

Jawaban : Inner class (non-lokal) dapat dideklarasikan sebagai publik, dilindungi, pribadi, statis, final, atau abstrak.

Pertanyaan : Berapa banyak bit yang digunakan untuk merepresentasikan karakter Unicode, ASCII, UTF-16, dan UTF-8? (Core Java)

Jawaban : Unicode membutuhkan 16 bit dan ASCII membutuhkan 7 bit. Meskipun himpunan karakter ASCII hanya menggunakan 7 bit, biasanya direpresentasikan sebagai 8 bit. UTF-8 mewakili karakter yang menggunakan pola 8, 16, dan 18 bit. UTF-16 menggunakan pola bit 16-bit dan yang lebih besar.

Pertanyaan : Apa ituwrapped class? (Core Java)

Jawaban : Wrapped class adalah kelas yang memungkinkan tipe primitif untuk diakses sebagai objek.

Pertanyaan : Pembatasan apa yang ditempatkan di lokasi pernyataan paket dalam file source code? (Core Java)

Jawaban : Pernyataan paket harus muncul sebagai baris pertama dalam file source code (tidak termasuk baris kosong dan komentar).

Pertanyaan : Apa perbedaan antara preemptive scheduling dantime slicing? (Core Java)

Jawaban : Di bawah penjadwalan preemptive scheduling, tugas prioritas tertinggi dijalankan hingga memasuki status wait atau off atau tugas dengan prioritas lebih tinggi muncul.

Di bawah time slicing, tugas dijalankan untuk bagian waktu yang telah ditentukan dan kemudian memasukkan kembali kumpulan tugas yang siap. Penjadwal kemudian menentukan tugas mana yang harus dijalankan selanjutnya, berdasarkan prioritas dan faktor lainnya.

Pertanyaan : Apa metode asli? (Core Java)

Jawaban : Metode asli adalah metode yang diimplementasikan dalam bahasa selain Java.

Pertanyaan : Apa urutan prioritas dan asosiatif, & bagaimana penggunaannya?

Jawaban : Urutan prioritas menentukan urutan di mana operator dievaluasi dalam ekspresi. Keterkaitan menentukan apakah ekspresi dievaluasi dari kiri ke kanan atau kanan ke kiri.

Pertanyaan : Apa aturan tangkap atau deklarasikan untuk deklarasi metode? (Core Java)

Jawaban : Jika pengecualian yang dicentang dapat dilemparkan ke dalam komponen metode, metode tersebut harus menangkap pengecualian atau mendeklarasikannya dalam klausul lemparannya.

Pertanyaan : Bisakah kelas anonim dideklarasikan dengan penerapkan interface dan memperluas kelas? (Core Java)

Jawaban : Kelas anonim dapat mengimplementasikan interface atau memperluas superclass, tetapi mungkin tidak dideklarasikan untuk melakukan keduanya.

Pertanyaan : Berapakah kisaran tipe karakter? (Core java)

Jawaban : Kisaran tipe karakter adalah 0 hingga 2 ^ 16 - 1.

Pertanyaan : Apa tujuan finalisasi? (Core Java)

Jawaban : Tujuan finalisasi adalah memberikan kesempatan kepada objek yang tidak terjangkau untuk melakukan pemrosesan pembersihan apa pun sebelum objek tersebut dikumpulkan sampahnya.

Pertanyaan : Apa perbedaan antara Operator Boolean &dan Operator &&? (Core Java)

Jawaban : Jika ekspresi yang melibatkan operator Boolean & dievaluasi, kedua operan dievaluasi. Kemudian operator & diterapkan ke operan. Saat ekspresi yang melibatkan operator && dievaluasi, operan pertama dievaluasi. Jika operan pertama mengembalikan nilai true maka operan kedua dievaluasi. Operator && kemudian diterapkan ke operan pertama dan kedua. Jika operan pertama bernilai false, evaluasi operan kedua akan dilewati.

Pertanyaan : Berapa kali metode finalize() objek dipanggil oleh pengumpul sampah? (Core Java)

Jawaban : Metode finalize() objek hanya dapat dipanggil sekali oleh pengumpul sampah.

Pertanyaan : Apa tujuan dari finally clause dari pernyataan try-catchfinally? (Core Java)

Jawaban : Finally clause digunakan untuk memberikan kemampuan untuk mengeksekusi kode, tidak peduli apakah pengecualian dilempar, ditangkap atau tidak.

Pertanyaan : Apa tipe argumen dari metode main() program? (Core)

Jawaban : Metode main() program mengambil argumen dari tipe String [].

Pertanyaan : Operator Java manakah yang benar asosiatif? (Core Java)

Jawaban : Operator = adalah asosiatif yang benar.

Pertanyaan : Bisakah nilai ganda diubah menjadi satu byte? (Core Java)

Jawaban : Ya, nilai ganda dapat diubah menjadi satu byte.

Pertanyaan : Apa perbedaan antara pernyataan break dan pernyataan lanjutan? (Core Java)

Jawaban : Hasil pernyataan break dalam penghentian pernyataan yang berlaku (switch, for, do, or while). Pernyataan lanjutkan digunakan untuk mengakhiri literasi pengulangan saat ini dan mengembalikan kontrol ke pernyataan pengulangan.

Pertanyaan : Apa yang harus dilakukan kelas untuk mengimplementasikan interface? (Core Java)

Jawaban : Yaitu harus menyediakan semua metode dalam interface dan mengidentifikasi interface dalam klausul implementasinya.

Pertanyaan : Apa keuntungan dari model event-delegation dibandingkan model eventinheritance sebelumnya? (Core Java)

Jawaban : Model event-delegation memiliki dua keunggulan dibandingkan model event-inheritance. Pertama, ini memungkinkan penanganan peristiwa untuk ditangani oleh objek selain yang menghasilkan peristiwa (atau penampungnya).

Hal ini memungkinkan pemisahan yang bersih antara desain komponen dan penggunaannya. Keuntungan lain dari model eventdelegation adalah performanya jauh lebih baik dalam aplikasi di mana banyak event dibuat.

Peningkatan kinerja ini disebabkan oleh fakta bahwa model delegasi peristiwa tidak harus berulang kali memproses peristiwa yang tidak ditangani, seperti kasus model peinheritance peristiwa.

Pertanyaan : Bagaimana koma digunakan dalam bagian inisialisasi dan iterasi dari pernyataan for? (Core Java)

Jawaban : Koma digunakan untuk memisahkan beberapa pernyataan dalam inisialisasi danbagian iterasi dari pernyataan for.

Pertanyaan : Apa itu metode abstrak? (Core Java)

Jawaban : Metode abstrak adalah metode yang implementasinya ditangguhkan ke subkelas.

Pertanyaan : Nilai apa yang dikembalikan read() ketika telah mencapai akhir file? (Core Java)

Jawaban : Metode read() mengembalikan -1 ketika telah mencapai akhir file.

Pertanyaan : Bisakah objek Byte dilemparkan ke nilai ganda? (Core Java)

Jawaban : Tidak, sebuah objek tidak bisa dilemparkan ke nilai primitif.

Pertanyaan : Apa perbedaan antara inner class statis dan non-statis? (Core Jawa)

Jawaban : Inner class non-statis mungkin memiliki contoh objek yang terkait dengan contoh kelas luar. Inner class statis tidak memiliki instance objek apa pun.

Pertanyaan : Jika sebuah variabel dideklarasikan sebagai private, di manakah variabel tersebut dapat diakses? (Core Java)

Jawaban : Variabel privat hanya dapat diakses di dalam kelas yang dideklarasikan.

Pertanyaan : Apa kunci sebuah objek dan objek mana yang memiliki kunci? (Core Java)

Jawaban : Kunci objek adalah mekanisme yang digunakan oleh beberapa utas untuk mendapatkan akses yang disinkronkan ke objek. Sebuah utas dapat menjalankan metode tersinkronisasi dari suatu objek hanya setelah ia memperoleh kunci objek tersebut. Semua objek dan kelas memiliki kunci. Kunci kelas diperoleh pada objek Kelas.

Pertanyaan : Apa operator% itu? (Core Java)

Jawaban : Ini disebut sebagai modulo atau operator sisa. Ini mengembalikan sisa pembagian operan pertama dengan operan kedua.

Pertanyaan : Kapan referensi objek dapat dikirim ke referensi interface?

Jawaban : Referensi objek dilemparkan ke referensi Interface ketika objek mengimplementasikan interface yang direferensikan.

Pertanyaan : Kelas mana yang diperluas oleh semua kelas lainnya? (Core Java)

Jawaban : Kelas Object diperluas oleh semua kelas lainnya.

Pertanyaan : Bisakah suatu objek dikumpulkan sampah saat masih dapat dijangkau?

Jawaban : Objek yang dapat dijangkau tidak dapat dikumpulkan dari sampah. Hanya objek yang tidak terjangkau yang dapat dikumpulkan.

Pertanyaan : Apakah operator terner tertulis x: y? z atau x? y: z? (Core Java)

Jawaban : Ada tertulis x? y: z.

Pertanyaan : Bagaimana pembulatan dilakukan di bawah pembagian integer? (Core Java)

Jawaban : Bagian pecahan dari hasil tersebut dipotong. Ini dikenal sebagai pembulatan menuju nol.

Pertanyaan : Apa perbedaan antara hierarki kelas Reader/Writer danHierarki kelas InputStream/OutputStream? (Core Java)

Jawaban : Hierarki kelas Reader/Writer berorientasi pada karakter, dan hierarki kelas InputStream/OutputStream berorientasi pada byte.

Pertanyaan : Kelas pengecualian apa yang dapat ditangkap oleh klausa catch?

Jawaban : Klausa catch dapat menangkap pengecualian apa pun yang mungkin ditetapkan ke tipe Throwable. Ini termasuk jenis Kesalahan dan Pengecualian.

Pertanyaan : Jika sebuah kelas dideklarasikan tanpa pengubah akses, di mana kelas tersebut dapat diakses? (Core Java)

Jawaban : Kelas yang dideklarasikan tanpa pengubah akses disebut memiliki akses paket. Ini berarti bahwa kelas tersebut hanya dapat diakses oleh kelas dan interface lain yang ditentukan dalam paket yang sama.

Pertanyaan : Apakah kelas mewarisi konstruktor superclass-nya? (Core Java)

Jawaban : Kelas tidak mewarisi konstruktor dari salah satu kelas supernya.

Pertanyaan : Apa tujuan system class? (Core Java)

Jawaban : Tujuan system class adalah untuk menyediakan akses ke sumber daya sistem.

Pertanyaan : Sebutkan delapan tipe Java primitif. (Core Java)

Jawaban : Delapan tipe primitif adalah byte, char, short, int, long, float, double, dan boolean.

Pertanyaan : Kelas mana yang harus anda gunakan untuk memperoleh informasi desain tentang suatu objek? (Core Java)

Jawaban : Kelas yang digunakan untuk memperoleh informasi tentang desain objek.

Pertanyaan : Apakah "abc" adalah nilai primitif? (Core Java)

Jawaban : String literal "abc" bukanlah nilai primitif. Ini adalah objek String.

Pertanyaan : Batasan apa yang ditempatkan pada nilai setiap kasus pernyataan switch? (Core Java)

Jawaban : Selama kompilasi, nilai dari setiap kasus pernyataan sakelar harus mengevaluasi ke nilai yang dapat dipromosikan ke nilai int.

Pertanyaan : Pengubah apa yang dapat digunakan dengan deklarasi interface?

Jawaban : Interface dapat dideklarasikan sebagai publik atau abstrak.

Pertanyaan : Apakah sebuah kelas merupakan subkelasnya sendiri? (Core Java)

Jawaban : Kelas adalah subkelas itu sendiri.

Pertanyaan : Apa perbedaan antara pernyataan while dan pernyataan do?

Jawaban : Pernyataan while memeriksa awal pengulangan untuk melihat apakah pengulangan berikutnya harus terjadi.

Pernyataan : do memeriksa di akhir pengulangan untuk melihat apakah pengulangan berikutnya dari pengulangan harus terjadi. Pernyataan do akan selalu mengeksekusi badan loop setidaknya sekali.

Pertanyaan : Pengubah apa yang dapat digunakan dengan inner class lokal? (Core Java)

Jawaban : Inner class lokal mungkin final atau abstrak.

Pertanyaan : Apa tujuan dari kelas File? (Core Java)

Jawaban : Kelas File digunakan untuk membuat objek yang menyediakan akses ke file dan direktori sistem file lokal.

Pertanyaan : Bisakah pengecualian dicabut kembali? (Core Java)

Jawaban : Ya, pengecualian dapat dicabut kembali.

Pertanyaan : Kapan kompilator menyediakan konstruktor default untuk kelas?

Jawaban : Kompilator menyediakan konstruktor default untuk kelas jika tidak ada konstruktor lain yang disediakan.

Pertanyaan : Jika suatu metode dideklarasikan sebagai dilindungi, di manakah metode tersebut dapat diakses?

Jawaban : Metode yang dilindungi hanya dapat diakses oleh kelas atau interface dari paket yang sama atau oleh subkelas kelas tempat ia dideklarasikan.

Pertanyaan : Karakter huruf non-Unicode mana yang dapat digunakan sebagai karakter pertama pengenal? (Core Java)

Jawaban : Karakter huruf non-Unicode $ dan _ mungkin muncul sebagai karakter pengenal pertama.

Pertanyaan : Batasan apa yang diterapkan pada metode overloading? (Core Java)

Jawaban : Dua metode mungkin tidak memiliki nama dan daftar argumen yang sama tetapi tipe kembalian yang berbeda.

Pertanyaan : Apa itu casting? (Core Java)

Jawaban : Ada dua jenis casting, casting antara tipe numerik primitif dan casting di antara referensi objek. Transmisi antar tipe numerik digunakan untuk mengonversi nilai yang lebih besar, seperti nilai ganda, menjadi nilai yang lebih kecil, seperti nilai

byte. Transmisi antar referensi objek digunakan untuk merujuk ke objek dengan kelas yang kompatibel, interface, atau referensi tipe array.

Pertanyaan : Apa tipe kembalian dari metode main() program? (Core Java)

Jawaban : Metode main() program memiliki tipe pengembalian kosong.

Pertanyaan : Kelas pengecualian apa yang dihasilkan oleh sistem runtime Java?

Jawaban : Sistem runtime Java menghasilkan pengecualian RuntimeException dan Error.

Pertanyaan : Kelas apa yang memungkinkan anda membaca objek langsung dari aliran?

Jawaban : Kelas ObjectInputStream mendukung pembacaan objek dari aliran input.

Pertanyaan : Apa perbedaan antara variabel bidang dan variabel lokal?

Jawaban : Variabel bidang adalah variabel yang dideklarasikan sebagai anggota kelas. Variabel lokal adalah variabel yang dideklarasikan secara lokal ke suatu metode.

Pertanyaan : Bagaimana this() dan super() digunakan dengan konstruktor? (Core Java)

Jawaban : this() digunakan untuk memanggil konstruktor dari kelas yang sama. super() digunakan untuk menjalankan konstruktor superclass.

Pertanyaan : Apa hubungan antara klausa lemparan metode dan pengecualian yang dapat dilemparkan selama eksekusi metode? (Core Java)

Jawaban : Klausa lemparan metode harus mendeklarasikan pengecualian apa pun yang dicentang yang tidak ada di dalam komponen metode.

Pertanyaan : Mengapa metode kelas Matematika bersifat statis? (Core Java)

Jawaban : Jadi mereka bisa dipanggil seolah-olah mereka adalah pustaka kode matematika.

Pertanyaan : Apa operand legal dari operator instanceof? (Core Java)

Jawaban : Operan kiri adalah referensi objek atau nilai null dan operan kanan adalah kelas, interface, atau tipe array.

Pertanyaan : Apa itu filter I/O? (Core Java)

Jawaban : Filter I/O adalah objek yang membaca dari satu aliran dan menulis ke aliran lain, biasanya mengubah data dengan cara tertentu saat diteruskan dari satu aliran ke aliran lainnya.

Pertanyaan : Jika suatu objek dikumpulkan sampah, apakah dapat dijangkau lagi?

Jawaban : Setelah sebuah objek dikumpulkan sampah, ia tidak ada lagi. Itu tidak bisa dijangkau lagi.

Pertanyaan : Apa itu E dan PI? (Core Java)

Jawaban : E adalah basis dari logaritma natural dan PI adalah nilai matematika pi.

Pertanyaan : Apakah keyword benar dan salah? (Core Java)

Jawaban : Nilai benar dan salah bukanlah keyword.

Pertanyaan : Apa perbedaan antara kelas File dan RandomAccessFile? (Core Java)

Jawaban : Kelas File merangkum file dan direktori dari sistem file lokal.Kelas RandomAccessFile menyediakan metode yang diperlukan untuk mengakses data secara langsung yang terdapat di bagian mana pun dari file.

Pertanyaan : Apa yang terjadi jika anda menambahkan nilai ganda ke String? (Core Java)

Jawaban : Hasilnya adalah objek String.

Pertanyaan : Apa pengkodean karakter default platform anda? (Core Java)

Jawaban : Jika anda menjalankan Java pada platform Windows Inggris, mungkin Cp1252. Jika anda menjalankan Java pada platform Solaris bahasa Inggris, kemungkinan besar adalah 8859_1.

Pertanyaan : Paket mana yang selalu diimpor secara default? (Core Java)

Jawaban : Paket java.lang selalu diimpor secara default.

Pertanyaan : Interface apa yang harus diterapkan objek sebelum dapat ditulis ke aliran sebagai objek? (Core Java)

Jawaban : Sebuah objek harus mengimplementasikan Interface Serializable atau Externalizable sebelum dapat ditulis ke aliran sebagai objek.

Pertanyaan : Bagaimana aplikasi saya mengetahui saat HttpSession dihapus?

Jawaban : Definisikan Class HttpSessionNotifier yang mengimplementasikan HttpSessionBindingListener dan mengimplementasikan fungsionalitas yang anda perlukan dalam metode valueUnbound(). Buat instance dari kelas itu dan letakkan instance itu di HttpSession.

Pertanyaan : Apa perbedaan antara notify() dan notifyAll()? (Core Java)

Jawaban : notify() digunakan untuk membuka blokir satu thread yang menunggu; notifyAll() digunakan untuk membuka semua blokir. Menggunakan notify() lebih disukai (untuk efisiensi) ketika hanya satu utas yang diblokir yang dapat memanfaatkan perubahan (misalnya, saat membebaskan kembali buffer ke dalam kelompok). notifyAll() diperlukan (untuk kebenaran) jika beberapa utas harus dilanjutkan (misalnya, saat melepaskan kunci "writer" pada file mungkin mengizinkan semua "readers" untuk melanjutkan).

Pertanyaan : Mengapa saya tidak bisa hanya mengatakan abs() atau sin() melainkan Math.abs() dan Math.sin()? (Core java)

Jawaban : Pernyataan import tidak membawa metode ke dalam ruang nama lokal anda. Ini memungkinkan anda menyingkat nama kelas, tetapi tidak menghapusnya sama sekali. Begitulah cara kerjanya, anda akan terbiasa dengannya. Jauh lebih aman dengan cara ini.
 Namun, sebenarnya ada sedikit trik yang dapat anda gunakan dalam beberapa kasus yang membuat anda mendapatkan apa yang anda inginkan.

Jika kelas level teratas tidak perlu mewarisi dari yang lain, buatlah itu mewarisi dari java.lang.Math. Itu * tidak * membawa semua metode ke dalam ruang nama lokal. Tetapi anda tidak dapat menggunakan trik ini di applet, karena anda harus mewarisi dari java.awt.Applet. Dan sebenarnya, anda tidak dapat menggunakannya sama sekali

di java.lang.Math, karena Matematika adalah kelas "terakhir" yang artinya tidak bisa diperpanjang.

Pertanyaan : Apakah mungkin bagi klien EJB untuk menyusun objek kelas java.lang.Class ke EJB? (EJB)

Jawaban : Secara teknis ya, spek. patuh TIDAK! – Enterprise Bean tidak boleh untuk meminta kelas untuk mendapatkan informasi tentang anggota yang dideklarasikan yang tidak dapat diakses oleh Enterprise bean karena aturan keamanan bahasa Java.

Pertanyaan : Apakah legal untuk memiliki blok penginisialisasi statis di EJB? (EJB)

Jawaban : Meskipun secara teknis legal, blok penginisialisasi statis digunakan untuk mengeksekusi beberapa bagian kode sebelum menjalankan konstruktor atau metode apa pun saat membuat instance kelas.

Blok penginisialisasi statis juga biasanya digunakan untuk menginisialisasi bidang statis - yang mungkin ilegal di EJB jika dibaca/ditulis - Di EJB hal ini dapat dicapai dengan menyertakan kode baik dalam metode ejbCreate(), setSessionContext() atau setEntityContext() .

Pertanyaan : Bagaimana cara menerapkan halaman JSP yang aman untuk thread? (JSP)

Jawaban : Anda dapat membuat JSP aman untuk thread dengan meminta mereka mengimplementasikan Interface SingleThreadModel. Ini dilakukan dengan menambahkan perintah <% @ page isThreadSafe = "false"%> di dalam halaman JSP.

Pertanyaan : Apakah mungkin untuk menghentikan eksekusi metode sebelum penyelesaian di SessionBean? (EJB)

Jawaban : Menghentikan eksekusi metode di dalam Session Bean tidak dimungkinkan tanpa menulis kode di dalam Session Bean. Ini karena nda tidak diizinkan mengakses Thread di dalam EJB.

Pertanyaan : Apa atribut transaksi default untuk EJB? (EJB)

Jawaban : Tidak ada atribut transaksi default untuk EJB. Bagian 11.5 dari spesifikasi EJB v1.1 mengatakan bahwa penerapan harus menentukan nilai atribut transaksi untuk metode tersebut yang memiliki transaksi yang dikelola container. Dalam weblogic, atribut transaksi default untuk EJB adalah SUPPORTS.

Pertanyaan : Apa perbedaan antara session dan entity beans? Kapan saya harus menggunakan salah satunya? (EJB)

Jawaban : Entity beans mewakili data global yang persisten dari database; session bean mewakili data khusus pengguna sementara yang akan mati saat pengguna terputus (mengakhiri sesinya). Umumnya, session bean menerapkan metode bisnis (misalnya Bank.transferFunds) yang memanggil entity bean (misalnya Account.deposit, Account.withdraw)

Pertanyaan : Apakah ada sistem manajemen cache default dengan entity bean? Dengan kata lain apakah cache data di database akan dipertahankan di EJB?

Jawaban : Cache data dari database di dalam Application Server digunakan untuk Entity EJB. Metode ejbLoad() dan ejbStore() digunakan untuk menyinkronkan status Entity Bean dengan penyimpanan persisten (database).

Transaksi juga memainkan peran penting dalam skenario ini. Jika data dihapus dari database, melalui aplikasi eksternal - Entity Bean anda masih bisa "menghidupkan" container EJB. Ketika transaksi dilakukan, ejbStore() dipanggil dan baris tidak akan ditemukan, dan transaksi dibatalkan.

Pertanyaan : Mengapa ejbFindByPrimaryKey wajib? (EJB)

Jawaban : Entity Bean merepresentasikan data persisten yang disimpan di luar EJB Container/Server. EjbFindByPrimaryKey adalah metode yang digunakan untuk mencari dan memuat Entity Bean ke dalam container, mirip dengan pernyataan SELECT di SQL. Dengan membuat metode ini wajib, programmer klien dapat diyakinkan bahwa jika mereka memiliki kunci utama Entity Bean, maka mereka dapat mengambil bean tanpa harus membuat bean baru setiap kali - yang berarti membuat duplikasi data persisten dan memecah integritas EJB.

Pertanyaan : Mengapa kami memiliki metode penghapusan di EJBHome dan EJBObject?

Jawaban : Dengan versi penghapusan EJBHome, anda dapat menghapus entity bean tanpa terlebih dahulu membuatnya (anda dapat memberikan objek PrimaryKey sebagai parameter untuk metode hapus). Versi beranda hanya berfungsi untuk entity bean. Di sisi lain, versi interface jarak jauh bekerja pada entity bean yang telah anda temukan. Selain itu, versi jarak jauh juga bekerja pada session bean (stateless dan statefull) untuk menginformasikan container hilangnya minat anda pada bean ini.

Pertanyaan : Bagaimana cara memanggil satu EJB dari dalam EJB lain? (EJB)

Jawaban : EJB dapat menjadi klien EJB lain. Ini berhasil. Gunakan JNDI untuk menemukan Interface home bean lain, lalu dapatkan referensi contoh, dan seterusnya.

Pertanyaan : Apa perbedaan antara Server, Container, dan Connector? (EJB)

Jawaban : Server EJB adalah aplikasi, biasanya produk seperti BEA WebLogic, yang menyediakan (atau harus menyediakan) untuk koneksi klien bersamaan dan mengelola sumber daya sistem seperti utas, proses, memori, koneksi database, koneksi jaringan, dll. Container EJB berjalan di dalam (atau di dalam) server EJB, dan menyediakan bean EJB yang diterapkan dengan manajemen transaksi dan keamanan, dll.

Container EJB mengisolasi bean EJB dari spesifikasi server EJB yang mendasarinya dengan menyediakan API standar yang sederhana antara bean EJB dan container. Konektor menyediakan kemampuan untuk Sistem Informasi Perusahaan (EIS) apa pun untuk dicolokkan ke server EJB mana pun yang mendukung arsitektur Konektor. Lihat http://java.sun.com/j2ee/connector/ untuk informasi lebih mendalam tentang Konektor.

Pertanyaan : Bagaimana persistensi diimplementasikan dalam enterprise beans? (EJB)

Jawaban : Persistensi di EJB dilakukan dengan dua cara, bergantung pada bagaimana anda mengimplementasikan bean: container managed persistence (CMP) atau bean managed persistence (BMP) Untuk CMP, container EJB tempat bean anda dijalankan menjaga persistensi bidang yang telah anda nyatakan tetap ada dengan database - deklarasi ini ada dalam deskriptor penerapan.

Jadi, setiap kali anda mengubah bidang dalam bean CMP, segera setelah metode yang anda jalankan selesai, data baru disimpan ke database oleh penampung. Untuk BMP, pengembang bean EJB bertanggung jawab untuk mendefinisikan rutinitas persistensi di tempat yang tepat di dalam bean, misalnya, metode ejbCreate(), ejbStore(), ejbRemove() akan dikembangkan oleh pengembang bean untuk membuat panggilan ke database.

Container bertanggung jawab, dalam BMP, untuk memanggil metode yang sesuai pada bean. Jadi, jika bean sedang dicari, ketika metode create() dipanggil pada interface Home, maka container bertanggung jawab untuk memanggil metode ejbCreate() di dalam bean, yang seharusnya memiliki fungsionalitas di dalamnya untuk pergi ke database dan mencaridata.

Pertanyaan : Apa itu Konteks EJB? (EJB)

Jawaban : EJBContext adalah interface yang diimplementasikan oleh container, dan juga merupakan bagian dari kontrak bean-container. Entity bean menggunakan subclass EJBContext yang disebut EntityContext. Session bean menggunakan subclass yang disebut SessionContext. Objek EJBContext ini menyediakan kelas bean dengan informasi tentang container, klien menggunakan bean dan bean itu sendiri. Mereka juga menyediakan fungsi lain. Lihat dokumen API dan spesifikasi untuk lebih jelasnya.

Pertanyaan : Apakah metode overloading diperbolehkan di EJB? (EJB)

Jawaban : Ya, anda dapat membebani metode

Pertanyaan : Haruskah sinkronisasi primitif digunakan pada metode bean? (EJB)

Jawaban : Tidak. Spesifikasi EJB secara spesifik menyatakan bahwa enterprise bean tidak diperbolehkan menggunakan primitif thread. Container bertanggung jawab untuk mengelola akses bersamaan ke bean pada saat runtime.

Pertanyaan : Apakah kita diperbolehkan mengubah properti isolasi transaksi di tengah transaksi? (EJB)

Jawaban : Tidak. Anda tidak dapat mengubah tingkat isolasi transaksi di tengah transaksi.

Pertanyaan : Untuk Entity Beans, apa yang terjadi pada bidang instance yang tidak dipetakan ke penyimpanan persisten apa pun, saat bean dipasivasi? (EJB)

Jawaban : Spesifikasi menyimpulkan bahwa container tidak pernah membuat serial sebuah instance dari Entity bean (tidak seperti status session bean). Jadi passivasi hanya melibatkan pemindahan bean dari tempat "ready" ke tempat "pooled".

Jadi apa yang terjadi pada konten variabel instance dikontrol oleh programmer. Ingat bahwa ketika entity bean dipasivasi, instance akan dipisahkan secara logis dari objek jarak jauhnya.

Berhati-hatilah di sini, karena fungsi passivation/activation untuk Stateless Session, Stateful Session dan Entity beans sangat berbeda. Untuk entity bean, metode ejbPassivate memberi tahu entity bean bahwa ia sedang dipisahkan dengan entitas tertentu sebelum digunakan kembali atau untuk dereferenc.

Pertanyaan : Apa itu Message Driven Bean, Fungsi apa yang dimiliki message driven bean dan bagaimana cara kerjanya dalam kolaborasi dengan JMS? (EJB)

Jawaban : Message Driver Bean adalah tambahan terbaru untuk keluarga jenis komponen bean yang ditentukan oleh spesifikasi EJB.

Jenis bean asli termasuk session bean, yang berisi logika bisnis dan mempertahankan status yang terkait dengan sesi klien, dan entity bean, yang memetakan objek ke data yang persisten. Bean yang digerakkan pesan akan menyediakan sinkronisasi untuk aplikasi berbasis EJB dengan bertindak sebagai konsumen pesan JMS.

Message Bean dikaitkan dengan topik atau antrian JMS dan menerima pesan JMS yang dikirim oleh klien EJB atau bean lainnya. Tidak seperti entity bean dan session bean, message bean tidak memiliki interface home atau jarak jauh. Sebagai gantinya, bean yang didorong pesan dibuat oleh container seperti yang diperlukan.

Seperti session bean tanpa status, message bean tidak mempertahankan status khusus klien, memungkinkan container untuk mengelola kumpulan instance message bean secara optimal. Klien mengirim pesan JMS ke message beans dengan cara yang persis sama seperti mereka mengirim pesan ke tujuan JMS lainnya.

Kesamaan ini adalah tujuan desain fundamental dari kapabilitas JMS dari spesifikasi baru. Untuk menerima pesan JMS, bean berbasis pesan mengimplementasikan interface javax.jms.MessageListener, yang mendefinisikan metode "onMessage()" tunggal.

Ketika sebuah pesan tiba, container memastikan bahwa message bean yang sesuai dengan topik/antrian pesan ada (membuat instance jika perlu), dan memanggil metode onMessage-nya yang meneruskan pesan klien sebagai argumen tunggal. Implementasi message bean dari metode ini berisi logika bisnis yang diperlukan untuk memproses pesan.

Perhatikan bahwa session bean dan entity bean tidak diizinkan berfungsi sebagai message bean.

Pertanyaan : Apakah RMI-IIOP mendukung pengunduhan kode untuk objek Java yang dikirim oleh nilai melalui koneksi IIOP dengan cara yang sama seperti yang dilakukan RMI pada koneksi JRMP? (RMI)

Jawaban : Iya. JDK 1.2 mendukung pemuatan kelas dinamis.

Pertanyaan : Container EJB mengimplementasikan kelas EJBHome dan EJBObject. Untuk setiap permintaan dari klien unik, apakah container membuat instance terpisah dari kelas EJBHome dan EJBObject yang dihasilkan? (EJB)

Jawaban : Kontainer EJB mempertahankan kumpulan instance. Penampung menggunakan instance ini untuk referensi Beranda EJB terlepas dari permintaan klien. saat merujuk kelas Objek EJB, wadah membuat instance terpisah untuk setiap permintaan klien.

Jawaban Lain

Pemeliharaan kumpulan instance tergantung pada penerapan penampung. Jika penampung menyediakannya, itu tersedia jika tidak, penyedia tidak wajib mengimplementasikannya.

Karena itu, ya, sebagian besar penyedia penampung mengimplementasikan fungsionalitas penggabungan untuk meningkatkan kinerja server aplikasi. Bagaimana itu diterapkan, sekali lagi terserah pelaksana.

Pertanyaan : Apa keuntungan dari menempatkan instance Entity Bean dari "Ready State" menjadi "Pooled State"? (EJB)

Jawaban : Ide dari "Pooled State" adalah untuk memungkinkan sebuah container untuk memelihara kumpulan entity bean yang telah dibuat, tetapi belum "disinkronkan" atau ditugaskan ke sebuah EJBObject. Ini berarti bahwa instance tersebut merepresentasikan entity bean, tetapi hanya dapat digunakan untuk melayani metode Home (create atau findBy), karena metode tersebut tidak menyampaikan nilai spesifik bean tersebut.

Semua contoh ini, pada kenyataannya, persis sama, jadi, mereka tidak memiliki status yang berarti. Jon Thorarinsson juga menambahkan:

Hal ini dapat dilihat dengan cara: Jika tidak ada klien yang menggunakan entity bean dari jenis tertentu, tidak perlu membuat cache (data disimpan dalam database). Oleh karena itu, dalam kasus seperti itu, container akan, setelah beberapa waktu, memindahkan entity bean dari "Ready State" ke "Pooled State" untuk menghemat memori.

Kemudian, untuk menyimpan memori tambahan, container dapat mulai memindahkan entity bean dari "Pooled State" ke "Does Not Exixt State", karena meskipun cache bean telah dihapus, Bean masih menggunakan beberapa memori hanya dengan berada di "Pooled State".

Pertanyaan : Dapatkah Session Bean didefinisikan tanpa metode ejbCreate()? (EJB)

Jawaban : Metode ejbCreate() adalah bagian dari siklus hidup bean, jadi kompilator tidak akan mengembalikan kesalahan karena tidak ada metode ejbCreate(). Namun, spesifikasi J2EE eksplisit: interface beranda Stateless Session Bean harus memiliki metode create() tunggal tanpa argumen, sedangkan kelas session bean harus berisi tepat satu metode ejbCreate(), juga tanpa argumen.

Stateful Session Bean dapat memiliki argumen (lebih dari satu metode pembuatan) stateful bean dapat berisi beberapa ejbCreate() asalkan cocok dengan definisi home interface, anda memerlukan referensi ke EJBObject untuk memulai. Untuk itu Sun bersikeras meletakkan metode untuk membuat referensi itu (membuat metode di home interface). EJBObject penting di sini. Bukan bean yang sebenarnya.

Pertanyaan : Apakah mungkin untuk berbagi HttpSession antara JSP dan EJB? Apa yang terjadi jika saya mengubah nilai di HttpSession dari dalam EJB? (EJB)

Jawaban : Anda dapat meneruskan HttpSession sebagai parameter ke metode EJB, hanya jika semua objek dalam sesi dapat diserialkan. Ini harus dianggap sebagai "nilai yang diteruskan", yang berarti bahwa itu hanya dapat dibaca di EJB. Jika ada yang diubah dari dalam EJB, itu tidak akan direfleksikan kembali ke HttpSession dari Servlet Container. "Pass-byreference" dapat digunakan antara EJBs Remote Interfaces, karena mereka adalah referensi jarak jauh.

Meskipun IS mungkin untuk meneruskan HttpSession sebagai parameter ke objek EJB, ini dianggap sebagai "bad practice (1)" dalam hal desain berorientasi objek. Ini karena anda membuat penggabungan yang tidak perlu antara objek back-end (ejbs) dan objek frontend (HttpSession).

Buat tingkat abstraksi yang lebih tinggi untuk api ejb anda. Daripada meneruskan keseluruhan, HttpSession (yang membawa sekumpulan semantik http), buat kelas yang bertindak sebagai objek nilai (atau struktur) yang menyimpan semua data yang anda butuhkan untuk diteruskan bolak-balik antara front-end/back-end.

Pertimbangkan kasus di mana ejb anda perlu mendukung klien berbasis non-http. Tingkat abstraksi yang lebih tinggi ini akan cukup fleksibel untuk mendukungnya. (1) Core J2EE design patterns (2001).

Pertanyaan : Apakah ada cara untuk membaca nilai dari entity bean tanpa menguncinya untuk sisa transaksi (misalnya transaksi read-only)? Kami memiliki peta key-value yang menemui jalan buntu selama beberapa pembacaan bersamaan. Tingkat isolasi tampaknya mempengaruhidatabase saja, dan kita perlu bekerja dalam transaksi. (EJB)

Jawaban : Satu-satunya hal yang terlintas dalam pikiran (saya) adalah bahwa anda dapat menulis 'group accessor' - metode yang mengembalikan satu objek yang berisi semua atribut entity bean anda (atau semua atribut yang menarik). Metode ini kemudian dapat ditempatkan dalam transaksi 'Requires New'.

Dengan cara ini, transaksi saat ini akan ditangguhkan selama durasi panggilan ke entity bean dan siklus fetch/operate/commit dari entity bean akan berada dalam transaksi terpisah dan kunci apa pun harus segera dilepaskan. Bergantung pada perincian apa yang perlu anda tarik dari peta, group accessor mungkin berlebihan.

Pertanyaan : Apa perbedaan antara Entity Bean "Coarse Grained" dan Entity Bean "Fine Grained"? (EJB)

Jawaban : Entity Bean 'fine grained' cukup banyak secara langsung dipetakan ke satu tabel relasional, dalam bentuk normal ketiga. Entity bean 'coarse grained' lebih besar dan lebih kompleks, baik karena atributnya menyertakan nilai atau daftar dari tabel lain, atau karena ia 'memiliki' satu atau lebih set objek dependen.

Perhatikan bahwa coarse grained bean mungkin dipetakan ke satu tabel atau file datar, tetapi tabel tunggal itu akan menjadi sangat jelek, dengan data disalin dari tabel lain, grup bidang berulang, kolom yang bergantung pada bidang non-kunci, dll.

Fine Grained Entity umumnya dianggap sebagai liabilitas dalam sistem besar karena mereka akan cenderung meningkatkan beban pada beberapa subsistem server EJB (akan ada lebih banyak objek yang diekspor melalui lapisan distribusi, lebih banyak objek yang berpartisipasi dalam transaksi, lebih banyak kerangka dalam memori, lebih banyak Objek EJB dalam memori, dll.)

Sisi lain dari koin ini adalah bahwa spesifikasi 1.1 tidak mengamanatkan CMP Error! Tidak ada entri indeks yang ditemukan. Dukungan untuk objek dependen (atau bahkan menunjukkan bagaimana mereka harus didukung), yang membuatnya lebih sulit untuk melakukan objek coarse grained dengan CMP. Spesifikasi EJB 2.0 meningkatkan hal ini dengan cara yang sangat besar.

Pertanyaan : Apa itu EJBDoclet? (EJB)

Jawaban : EJBDoclet adalah dokumen JavaDoc open source yang menghasilkan banyak file sumber terkait EJB dari tag komentar JavaDoc kustom yang disematkan di file sumber EJB.

Pertanyaan : Apa keluaran dari System.out.println ("Hello" + null);

Jawaban : Hellonull

Pertanyaan : Bisakah kita menggunakan konstruktor, sebagai ganti init(), untuk menginisialisasi servlet? (Servlets)

Jawaban : Ya, tentu saja anda dapat menggunakan konstruktor daripada init(). Tidak ada yang menghentikan anda. Tapi sebaiknya jangan. Alasan asli penggunaan init() adalah bahwa versi kuno Java tidak dapat secara dinamis memanggil konstruktor dengan argumen, jadi tidak ada cara untuk memberikan konstruktor ServletConfig. Itu tidak lagi berlaku, tetapi container servlet masih hanya akan memanggil konstruktor no-arg anda. Jadi, anda tidak akan memiliki akses ke ServletConfig atau ServletContext.

Pertanyaan : Bagaimana servlet me-refresh secara otomatis jika beberapa data baru telah masuk ke database? (Servlets)

Jawaban : Anda dapat menggunakan Refresh sisi klien atau Server Push.

Pertanyaan : Kode dalam klausa finally tidak akan pernah gagal untuk dieksekusi, bukan? (Servlets)

Jawaban : Menggunakan System.exit(1); di blok percobaan tidak akan memungkinkan finally kode untuk dieksekusi.

Pertanyaan : Mengapa tidak ada variabel global di Java? (Core Java)

Jawaban : Variabel global dianggap bentuk yang buruk karena berbagai alasan:

Menambahkan variabel status merusak transparansi referensial (anda tidak lagi dapat memahami pernyataan atau ekspresi itu sendiri: anda perlu memahaminya dalam konteks pengaturan variabel global). Variabel status mengurangi kohesi program: anda perlu tahu lebih banyak untuk memahami cara kerja sesuatu. Poin utama dari pemrograman ObjectOriented adalah memecah status global menjadi kumpulan negara lokal yang lebih mudah dipahami. Ketika anda menambahkan satu variabel, anda membatasi penggunaan program menjadi satu contoh. Apa yang anda pikir bersifat global, orang lain mungkin menganggapnya lokal: mereka mungkin ingin

menjalankan dua salinan program sekaligus. Karena alasan ini, Java memutuskan untuk melarang variabel global.

Pertanyaan : Apa artinya kelas atau anggota sudah final? (Core Java)

Jawaban : Kelas final tidak dapat lagi dijadikan subkelas. Sebagian besar ini dilakukan untuk alasan keamanan dengan kelas dasar seperti String dan Integer. Ini juga memungkinkan kompiler untuk membuat beberapa pengoptimalan, dan membuat keamanan thread sedikit lebih mudah untuk dicapai. Metode juga dapat dinyatakan final. Ini berarti mereka tidak dapat diganti dalam subclass.

Fields juga bisa dinyatakan final. Namun, ini memiliki arti yang sangat berbeda. Fields terakhir tidak dapat diubah setelah diinisialisasi, dan harus menyertakan pernyataan penginisialisasi di tempat yang dideklarasikan. Misalnya, double final public c = 2.998;

Ini juga memungkinkan untuk membuat field statis final untuk mendapatkan efek pernyataan const C++ atau beberapa penggunaan #define C, mis. publik akhir statis ganda c = 2.998;

Pertanyaan : Apa artinya metode atau kelas itu abstrak? (Core Java)

Jawaban : Kelas abstrak tidak dapat dibuat instance-nya. Hanya subkelasnya yang dapat dibuat instance-nya.

Anda menunjukkan bahwa kelas abstrak dengan keyword abstrak seperti ini: kelas abstrak publik Container extends Komponen {Kelas abstrak mungkin berisi metode abstrak. Metode yang dideklarasikan abstrak sebenarnya tidak diterapkan di kelas saat ini. Itu ada hanya untuk diganti dalam subclass. Ia tidak memiliki tubuh. Misalnya, harga float abstrak publik();

Metode abstrak hanya dapat dimasukkan dalam kelas abstrak. Namun, kelas abstrak tidak diharuskan memiliki metode abstrak, meskipun kebanyakan dari mereka melakukannya. Setiap subkelas dari kelas abstrak harus mengganti metode abstrak dari kelas supernya atau dirinya sendiri dinyatakan abstrak.

Pertanyaan : apa itu variabel transien? (Core Java)

Jawaban : Variabel transien adalah variabel yang tidak dapat diserialkan.

Pertanyaan : Bagaimana Observer dan Observable digunakan? (Core Java)

Jawaban : Objek yang merupakan subclass dari kelas Observable menyimpan daftar pengamat. Saat objek Observable diperbarui, ia memanggil metode update() dari setiap pengamatnya untuk memberi tahu pengamat bahwa statusnya telah berubah. Interface Observer diimplementasikan oleh objek yang mengamati objek Observable.

Pertanyaan : Bisakah kunci diperoleh di kelas? (Core Java)

Jawaban : Ya, kunci dapat diperoleh di kelas. Kunci ini diperoleh pada objek Kelas kelas.

Pertanyaan : Status apa yang dimasuki thread saat menghentikan pemrosesannya?

Jawaban : Saat utas menghentikan pemrosesannya, utas memasuki kondisi mati.

Pertanyaan : Bagaimana Java menangani overflows dan underflows integer?

Jawaban : Ini menggunakan byte urutan rendah dari hasil yang dapat sesuai dengan ukuran jenis yang diizinkan oleh operasi.

Pertanyaan : Apa perbedaan antara operator >> dan >>>?

Jawaban : Operator >> membawa sedikit tanda saat menggeser ke kanan. Bit >>> pengisian nol yang telah digeser keluar.

Pertanyaan : Apakah sizeof sebuah keyword? (Core Java)

Jawaban : Ukuran operator bukanlah keyword.

Pertanyaan : Apakah Garbage Collection menjamin bahwa program tidak akan kehabisan memori? (Core Java)

Jawaban : Garbage Collection tidak menjamin bahwa program tidak akan kehabisan memori. Ada kemungkinan program menggunakan sumber daya memori lebih cepat daripada sampah yang dikumpulkan. Mungkin juga program membuat objek yang tidak tunduk pada Garbage Collection.

Pertanyaan : Bisakah metode finalize() objek yang dipanggil saat itu bisa dijangkau?

Jawaban : Metode finalize() objek tidak bisa dipanggil oleh pengumpul sampah saat objek masih bisa dijangkau. Namun, metode finalize() objek mungkin dipanggil oleh objek lain.

Pertanyaan : Nilai apa yang dikembalikan readLine() ketika telah mencapai akhir file? (Core Java)

Jawaban : Metode readLine() mengembalikan null ketika telah mencapai akhir file.

Pertanyaan : Bisakah pernyataan for loop tanpa batas? (Core Java)

Jawaban : Ya, pernyataan for bisa berulang tanpa batas. Misalnya, pertimbangkan hal berikut: for (;;);

Pertanyaan : Untuk nilai apa variabel tipe String diinisialisasi secara otomatis? (Core Java)

Jawaban : Nilai default dari tipe String adalah null.

Pertanyaan : Apa prioritas tugas dan bagaimana itu digunakan dalam penjadwalan?

Jawaban : Prioritas tugas adalah nilai integer yang mengidentifikasi urutan relatif yang harus dieksekusi sehubungan dengan tugas lain. Penjadwal mencoba menjadwalkan tugas dengan prioritas lebih tinggi sebelum tugas dengan prioritas lebih rendah.

Pertanyaan : Berapa kisaran tipe pendek? (Core Java)

Jawaban : Kisaran tipe pendek adalah - (2 ^ 15) hingga 2 ^ 15 - 1.

Pertanyaan : Apa tujuan Garbage Collection? (Core Java)

Jawaban : Tujuan Garbage Collection adalah untuk mengidentifikasi dan membuang objek yang tidak lagi dibutuhkan oleh program sehingga sumber dayanya dapat diambil kembali dan digunakan kembali.

Pertanyaan : Bagaimana model perpesanan yang disediakan JMS dan apa sajakah itu?

Jawaban : JMS menyediakan dua model perpesanan, publish-andsubscribe dan point-toppoint antrian.

Pertanyaan : Informasi apa yang diperlukan untuk membuat Soket TCP? (Networking)

Jawaban : Alamat IP dan Nomor Port Sistem Lokal. Dan IPAddress dan Nomor Port Sistem Jarak Jauh.

Pertanyaan : Apa yang akan dilakukan Class.forName saat memuat driver? (JDBC)

Jawaban : Ini digunakan untuk membuat instance driver dan mendaftarkannya dengan DriverManager. Ketika anda telah memuat driver, itu tersedia untuk membuat koneksi dengan DBMS.

Pertanyaan : Bagaimana cara Retrieve Warning? (JDBC)

Jawaban : Objek SQLWarning adalah subclass dari SQLException yang berhubungan dengan warnings akses database. Warnings tidak menghentikan eksekusi aplikasi, seperti halnya pengecualian; mereka hanya memberi tahu pengguna bahwa sesuatu tidak terjadi seperti yang direncanakan.

Warning dapat dilaporkan pada objek Connection, objek Pernyataan (termasuk PreparedStatement dan objek CallableStatement), atau objek ResultSet. Masing-masing kelas ini memiliki metode getWarnings, yang harus anda panggil untuk melihat peringatan pertama yang dilaporkan pada objek pemanggil Mis.

```
1       Peringatan SQLWarning = stmt.getWarnings();
2       if (warning! = null) {
3       while (warning! = null) {
4       System.out.println ("Message:" +
5       warning.getMessage());
6       System.out.println ("SQLState:" +
7       warning.getSQLState());
8       System.out.print ("Vendor error code:");
9       System.out.println (warning.getErrorCode());
10      warning = warning.getNextWarning();
11      }
12      }
```

Pertanyaan : Ada berapa elemen skrip JSP dan apa saja? (JSP)

Jawaban : Ada tiga elemen bahasa skrip:

deklarasi

scriptlets

ekspresi

Pertanyaan : Dalam spesifikasi Servlet 2.4, SingleThreadModel sudah tidak digunakan lagi, mengapa? (JSP)

Jawaban : Karena tidak praktis memiliki model seperti itu. Biarpun anda menyetel ThreadSafe ke true atau false, anda harus menangani permintaan klien secara bersamaan ke halaman JSP dengan menyinkronkan akses ke objek bersama yang ditentukan di tingkat halaman.

Pertanyaan : Apa stored procedures? Bagaimana fungsinya? (JDBC)

Jawaban : Stored procedures adalah sekumpulan pernyataan/perintah yang berada dalam database. Prosedur yang disimpan telah dikompilasi sebelumnya dan menghemat upaya parsing dan kompilasi pernyataan sql database setiap kali kueri dijalankan.

Setiap Database memiliki bahasa prosedur tersimpannya sendiri, biasanya varian C dengan preproceesor SQL. Versi yang lebih baru dari dukungan db menulis procs yang tersimpan di Java dan Perl juga.

Sebelum munculnya arsitektur 3-tier/n-tier, cukup umum bagi procs yang disimpan untuk mengimplementasikan logika bisnis (Banyak sistem masih melakukannya). Keuntungan terbesar tentu saja kecepatan. Juga jenis manipulasi data tertentu tidak dicapai dalam SQL.

Procs tersimpan menyediakan mekanisme untuk melakukan manipulasi ini. Procs yang disimpan juga berguna ketika anda ingin melakukan pembaruan Batch/ekspor/ jenis houseKeeping di db. Overhead Sambungan JDBC mungkin signifikan dalam kasus ini.

Pertanyaan : Apa yang anda pahami tentang private, protected, dan public?

Jawaban : Ini adalah pengubah aksesibilitas. Private adalah yang paling dibatasi, sedangkan public adalah yang paling tidak dibatasi. Tidak ada perbedaan nyata antara protected dan tipe default (juga dikenal sebagai package protected) dalam konteks package yang sama, namun keyword protected memungkinkan visibilitas ke kelas turunan dalam package yang berbeda.

Pertanyaan : Apa itu Downcasting? (Core Java)

Jawaban : Downcasting adalah casting dari tipe umum ke tipe yang lebih spesifik, yaitu menurunkan hierarki

Pertanyaan : Dapatkah sebuah metode kelebihan beban berdasarkan tipe kembalian yang berbeda tetapi tipe argumen yang sama? (Core Java)

Jawaban : Tidak, karena metode dapat dipanggil tanpa menggunakan tipe kembaliannya dalam hal ini terdapat ambiguitas untuk kompilator.

Pertanyaan : Apa yang terjadi pada var statis yang didefinisikan dalam metode kelas? (Core Java)

Jawaban : Tidak bisa digunakan. Anda akan mendapatkan kesalahan kompilasi.

Pertanyaan : Berapa banyak init statis yang dapat anda miliki? (Core Java)

Jawaban : Sebanyak yang anda inginkan, tetapi penginisialisasi statis dan penginisialisasi variabel kelas dijalankan dalam urutan tekstual dan tidak boleh merujuk ke variabel kelas yang dideklarasikan di kelas yang deklarasinya muncul secara tekstual setelah penggunaan, meskipun variabel kelas ini berada dalam cakupan.

Pertanyaan : Apa perbedaan antara JVMSpec, JVM Implementation, JVM Runtime? (Core Java)

Jawaban : JVM Spec adalah cetak biru untuk JVM yang dibuat dan dimiliki oleh Sun. JVM Implementation adalah implementasi sebenarnya dari spesifikasi oleh vendor dan JVM runtime adalah contoh aktual yang berjalan dari JVM Implementation.

Pertanyaan : Jelaskan apa yang terjadi ketika sebuah objek dibuat di Java?

Jawaban : Beberapa hal terjadi dalam urutan tertentu untuk memastikan objek dibangun dengan benar:

Memori dialokasikan dari heap untuk menampung semua variabel instan dan data spesifik implementasi dari objek dan superclass-nya. Data khusus implementasi mencakup petunjuk ke data kelas dan metode.

Variabel instance dari objek diinisialisasi ke nilai defaultnya.

Konstruktor untuk kelas yang paling diturunkan dipanggil. Hal pertama yang dilakukan konstruktor adalah memanggil consctructor untuk superclass-nya. Proses ini berlanjut hingga konstruktor untuk java.lang.Object dipanggil, karena java.lang.Object adalah kelas dasar untuk semua objek di java.

Sebelum isi konstruktor dijalankan, semua penginisialisasi variabel instance dan blok inisialisasi dijalankan. Kemudian tubuh konstruktor dijalankan. Jadi, konstruktor untuk kelas dasar diselesaikan terlebih dahulu dan konstruktor untuk kelas yang paling diturunkan diselesaikan terakhir.

Pertanyaan : Apa arti keyword "final" di depan variabel? Metode? Kelas? (Core Java)

Jawaban : FINAL untuk variabel : nilai konstan.

FINAL untuk metode : tidak bisa diganti.

FINAL untuk kelas : tidak dapat diturunkan.

Pertanyaan : Apa perbedaan antara instanceof dan isInstance?

Jawaban : Instanceof digunakan untuk memeriksa apakah sebuah objek dapat di-cast ke dalam tipe tertentu tanpa mengeluarkan pengecualian kelas cast. isInstance() menentukan apakah Objek yang ditentukan kompatibel dengan tugas dengan objek yang diwakili oleh Kelas ini. Metode ini adalah ekuivalen dinamis dari operator instanceof bahasa Java. Metode ini mengembalikan nilai true jika argumen Objek yang ditentukan adalah bukan nol dan dapat dilemparkan ke tipe referensi yang diwakili oleh objek Kelas ini tanpa memunculkan ClassCastException. Ia mengembalikan false jika tidak.

Pertanyaan : Mengapa perlu begitu banyak waktu untuk mengakses Applet yang memiliki Komponen Swing untuk pertama kali? (Swing)

Jawaban : Karena di balik setiap komponen Swing terdapat banyak objek dan sumber daya Java. Hal ini membutuhkan waktu untuk dibuat dalam memori. JDK 1.3 dari Sun memiliki beberapa perbaikan yang dapat mempercepat eksekusi aplikasi Swing.

Pertanyaan : Bagaimana cara menyertakan file statis dalam halaman JSP? (JSP)

Jawaban : Sumber daya statis harus selalu disertakan menggunakan direktif JSP include.

Dengan cara ini, penyertaan dilakukan hanya sekali selama fase penerjemahan. Contoh berikut menunjukkan sintaks: Perhatikan bahwa anda harus selalu memberikan URL relatif untuk atribut file. Meskipun anda juga dapat menyertakan sumber daya statis menggunakan tindakan tersebut, hal ini tidak disarankan karena penyertaan kemudian dilakukan untuk setiap permintaan.

Pertanyaan : Mengapa JComponent memiliki metode add() dan remove() tetapi Komponen tidak? (Swing)

Jawaban : Karena JComponent adalah subclass dari Container, dan dapat berisi komponen lain dan jcomponents.

Pertanyaan : Bagaimana anda akan membuat tombol dengan tepi membulat? (Swing)

Jawaban : Ada 2 cara. Hal pertama adalah mengetahui bahwa tepi JButton digambar oleh Border. sehingga anda dapat mengganti metode tombol paintComponent (Grafik) dan menggambar lingkaran atau persegi panjang bulat (apa saja), dan mematikan border. Atau anda dapat membuat batas khusus dengan menggambar lingkaran atau persegi panjang bulat di sekitar komponen apa pun dan mengatur batas tombol ke sana.

Pertanyaan : Bagaimana anda mendeteksi penekanan tombol di JComboBox? (Swing)

Jawaban : Tambahkan KeyListener ke komponen editor JComboBox daripada menambahkan KeyListener ke JComboBox itu sendiri.

Pertanyaan : Mengapa implementasi callback Swing (seperti listener) harus dijalankan dengan cepat? (Swing)

Jawaban : Karena callback dipanggil oleh thread pengiriman peristiwa yang akan diblokir memproses peristiwa lain selama metode anda membutuhkan waktu untuk mengeksekusinya.

Pertanyaan : Mengapa anda menggunakan SwingUtilities.invokeAndWait atau SwingUtilities.invokeLater? (Swing)

Jawaban : Saya ingin memperbarui komponen Swing tetapi saya tidak sedang callback. Jika saya ingin update segera terjadi (mungkin untuk komponen progress bar) maka saya akan menggunakan invokeAndWait. Jika saya tidak peduli saat update terjadi, saya akan menggunakan invokeLater.

Pertanyaan : Jika UI anda tampak macet secara berkala, apa kemungkinan penyebabnya?

Jawaban : Implementasi callback seperti ActionListener.actionPerformed atau MouseListener.mouseClicked membutuhkan waktu lama untuk dieksekusi sehingga memblokir thread pengiriman peristiwa agar tidak memproses peristiwa UI lainnya.

Pertanyaan : Metode Swing mana yang aman untuk thread? (Swing)

Jawaban : Satu-satunya metode thread-safe adalah repaint(), revalidate(), dan invalidate()

Pertanyaan : Mengapa JVM tidak berhenti ketika saya menutup semua jendela aplikasi? (Swing)

Jawaban : Operator AWT thread bukanlah daemon thread. Anda harus secara eksplisit memanggil System.exit untuk menghentikan JVM.

Pertanyaan : JFrame adalah komponen berbobot berat. Karena ini memperpanjang Frame awt, apakah Thread Safe? (Swing)

Jawaban : JFrame itu sendiri, karena pada dasarnya JFrame hanyalah java.awt.Frame, tetapi panel root/bukan panel konten, sehingga secara efektif mengikuti aturan yang sama untuk container Swing dan tidak dianggap thread safe.

Pertanyaan : Apa perbedaan antara invokeAndWait() dan invokeLater()? (Swing)

Jawaban : invokeAndWait() blok hingga tugas Runnable selesai; itu sinkron. invokeLater() memposting peristiwa aksi ke antrian acara dan segera kembali; itu asynchronous.

Pertanyaan : Mengapa tidak disarankan untuk memiliki contoh variabel di Interface.

Jawaban : Secara Default, Semua data dan metode dalam Interface bersifat publik. Memiliki variabel publik di kelas yang akan diimplementasikan akan melanggar prinsip Enkapsulasi. Saya harap itu cukup baik.

Pertanyaan : Apa perbedaan antara inner class dan nested class?

Jawaban : Ketika sebuah kelas didefinisikan dalam satu ruang lingkup dari kelas lain, maka itu menjadi Inner class. Jika pengubah akses inner class adalah statis, maka kelas itu menjadi nested class.

Pertanyaan : Apa itu compilation unit? (Core Java)

Jawaban : Compilation unit adalah file kode sumber Java.

Pertanyaan : Batasan apa yang diterapkan pada penggantian metode? (Core Java)

Jawaban : Metode yang diganti harus memiliki nama, daftar argumen, dan tipe kembalian yang sama.

Metode overriding mungkin tidak membatasi akses metode yang ditimpanya. Metode overriding tidak boleh menampilkan pengecualian apa pun yang mungkin tidak muncul oleh metode yang diganti.

Pertanyaan : Bagaimana thread yang mati dapat dimulai ulang? (Core Java)

Jawaban : Thread yang mati tidak dapat dimulai ulang.

Pertanyaan : Apa yang terjadi jika exception tidak tertangkap? (Core Java)

Jawaban : Hasil exception yang tidak tertangkap dalam metode uncaughtException() dari ThreadGroup yang sedang dilibatkan, yang pada akhirnya mengakibatkan penghentian program tempat ia dilempar.

Pertanyaan : Operasi aritmatika mana yang dapat menghasilkan ArithmeticException? (Core Java)

Jawaban : Integer/dan% dapat menghasilkan ArithmeticException

Pertanyaan : Bisakah kelas abstrak menjadi final? (Core Java)

Jawaban : Kelas abstrak tidak boleh dideklarasikan sebagai final

Pertanyaan : Apa yang terjadi jika pernyataan try-catch-finally tidak memiliki klausa catch untuk menangani exception yang dilemparkan ke dalam try statement? (Core Java)

Jawaban : Exception menyebar ke pernyataan trycatch tingkat lebih tinggi berikutnya (jika ada) atau menghasilkan penghentian program.

Pertanyaan : Apa itu numeric promotion? (Core Java)

Jawaban : Numeric promotion adalah konversi jenis numerik yang lebih kecil ke jenis numerik yang lebih besar, sehingga operasi integer dan floatingpoint dapat berlangsung. Dalam numeric promotion, nilai byte, char, dan short diubah menjadi nilai int. Nilai int juga diubah menjadi nilai panjang, jika perlu. Nilai long dan float diubah menjadi nilai ganda, sesuai kebutuhan.

Pertanyaan : Apa perbedaan antara kelas publik dan non-publik?

Jawaban : Kelas publik dapat diakses di luar paketnya. Kelas non-publik tidak dapat diakses di luar paketnya.

Pertanyaan : Untuk apa nilai variabel tipe boolean diinisialisasi secara otomatis? (Core Java)

Jawaban : Nilai default dari tipe boolean adalah salah.

Pertanyaan : Bisakah percobaan statement bersarang? (Core Java)

Jawaban : Ya

Pertanyaan : Apa perbedaan antara bentuk prefix dan postfix dari operator++? (Core Java)

Jawaban : Bentuk awalan melakukan operasi kenaikan dan mengembalikan nilai operasi kenaikan. Formulir postfix mengembalikan nilai saat ini ke semua ekspresi dan kemudian melakukan operasi penambahan pada nilai itu.

Pertanyaan : Apa tujuan dri statement block? (Core Java)

Jawaban : Statement block digunakan untuk mengatur urutan pernyataan sebagai satu grup pernyataan.

Pertanyaan : Apa itu Java package dan bagaimana cara menggunakannya? (Core Java)

Jawaban : Java package adalah konteks penamaan untuk kelas dan interface. Sebuah paket digunakan untuk membuat ruang nama terpisah untuk grup kelas dan interface. Paket juga digunakan untuk mengatur kelas dan interface terkait ke dalam satu unit API dan untuk mengontrol aksesibilitas ke kelas dan interface ini.

Pertanyaan : Pengubah apa yang dapat digunakan dengan kelas tingkat atas? (Core Java)

Jawaban : Kelas tingkat atas mungkin publik, abstrak, atau final.

Pertanyaan : Untuk apa kelas Object dan Class digunakan? (Core Java)

Jawaban : Kelas Object adalah kelas tingkat tertinggi dalam hierarki kelas Java. Kelas Class digunakan untuk mewakili kelas dan interface yang dimuat oleh program Java.

Pertanyaan : Bagaimana try statement menentukan klausa catch mana yang harus digunakan untuk menangani exception? (Core Java)

Jawaban : Ketika exception dilemparkan ke dalam komponen try statement, catch clauses dari try statement diperiksa dalam urutan kemunculannya. Klausa catch pertama yang mampu menangani exception akan dieksekusi. Klausa catch yang tersisa diabaikan.

Pertanyaan : Apa itu synchronized methods dan synchronized statement?

Jawaban : Synchronized methods adalah metode yang digunakan untuk mengontrol akses ke suatu objek.

Sebuah thread hanya menjalankan synchronized methods setelah mendapatkan kunci untuk objek atau kelas metode tersebut. Synchronized statement mirip dengan Synchronized methods.

Synchronized statement hanya dapat dijalankan setelah thread memperoleh kunci untuk objek atau kelas yang dirujuk dalam synchronized statement.

Pertanyaan : Apa perbedaan antara if statement dan switch statement? (Core Java)

Jawaban : If statement digunakan untuk memilih di antara dua alternatif. Ini menggunakan ekspresi boolean untuk memutuskan alternatif mana yang harus dieksekusi. Switch statement digunakan untuk memilih di antara beberapa alternatif. Ini menggunakan ekspresi int untuk menentukan alternatif mana yang harus dieksekusi.

Pertanyaan : Container mana yang menggunakan Border Layout sebagai tata letak defaultnya? (AWT)

Jawaban : Kelas window, Frame dan Dialog menggunakan border layout sebagai layput default mereka.

Pertanyaan : Berapa ukuran komponen yang disukai? (AWT)

Jawaban : Ukuran komponen yang disukai adalah ukuran komponen minimum yang memungkinkan komponen ditampilkan secara normal.

Pertanyaan : Penampung mana yang menggunakan FlowLayout sebagai tata letak defaultnya? (AWT)

Jawaban : Kelas Panel dan Applet menggunakan FlowLayout sebagai tata letak defaultnya.

Pertanyaan : Apa itu immediate superclass dari kelas Applet? (AWT)

Jawaban : Panel.

Pertanyaan : Sebutkan tiga subclass Komponen yang mendukung pengecatan (AWT)

Jawaban : Kelas Canvas, Frame, Panel, dan Applet pendukung lukisan.

Pertanyaan : Apa itu immediate superclass dari kelas Dialog? (AWT)

Jawaban : Window.

Pertanyaan : Apa itu kliping? (AWT)

Jawaban : Clipping adalah proses pembtasan operasi cat pada area atau bentuk terbatas.

Pertanyaan : Apa perbedaan antara MenuItem dan CheckboxMenuItem? (AWT)

Jawaban : Kelas CheckboxMenuItem memperluas kelas MenuItem untuk mendukung item menu yang mungkin dicentang atau tidak dicentang.

Pertanyaan : Kelas apa yang berada di AWT event hierarchy? (AWT)

Jawaban : Kelas java.awt.AWTEvent adalah kelas tingkat tertinggi dalam AWT event hierarchy.

Pertanyaan : Di paket mana ditentukan sebagian besar peristiwa AWT yang mendukung model delegasi peristiwa? (AWT)

Jawaban : Sebagian besar peristiwa terkait AWT, model delegasi peristiwa ditentukan dalam paket java.awt.event. Kelas AWTEvent didefinisikan dalam paket java.awt.

Pertanyaan : Kelas mana yang merupakan superclass langsung dari kelas MenuComponent (AWT)

Jawaban : Obyek.

Pertanyaan : Wadah mana yang mungkin memiliki MenuBar? (AWT)

Jawaban : Bingkai.

Pertanyaan : Apa hubungan antara kelas Canvas dan kelas Graphics? (AWT)

Jawaban : Objek Canvas menyediakan akses ke objek Graphics melalui metode paint().

Pertanyaan : Bagaimana elemen-elemen BorderLayout diatur? (AWT)

Jawaban : Elemen BorderLayout disusun di perbatasan (Utara, Selatan, Timur, dan Barat) dan di tengah container.

Pertanyaan : Apa perbedaan antara Jendela dan Bingkai? (AWT)

Jawaban : Kelas Frame memperluas Window untuk menentukan jendela aplikasi utama yang dapat memiliki menu bar.

Pertanyaan : Apa perbedaan antara kelas Font dan FontMetrics?

Jawaban : Kelas FontMetrics digunakan untuk mendefinisikan properti implementasi khusus, seperti naik dan turun, dari objek Font.

Pertanyaan : Bagaimana elemen CardLayout diatur? (AWT)

Jawaban : Elemen CardLayout ditumpuk, satu di atas yang lain, seperti setumpuk kartu.

Pertanyaan : Apa hubungan antara clipping dan repainting? (AWT)

Jawaban : Ketika sebuah jendela dicat ulang oleh thread lukisan AWT, itu mengatur daerah kliping ke area jendela yang membutuhkan pengecatan ulang.

Pertanyaan : Apa hubungan antara antarmuka event-listener dan kelas eventadapter? (AWT)

Jawaban : Antarmuka pendengar peristiwa menentukan metode yang harus diterapkan oleh penangan peristiwa untuk jenis peristiwa tertentu. Adaptor peristiwa menyediakan implementasi default dari antarmuka pemroses peristiwa.

Pertanyaan : Bagaimana komponen GUI menangani acaranya sendiri? (AWT)

Jawaban : Sebuah komponen dapat menangani kejadiannya sendiri dengan mengimplementasikan antarmuka eventlistener yang diperlukan dan menambahkan dirinya sebagai pemroses kejadiannya sendiri.

Pertanyaan : Bagaimana elemen GridBagLayout diatur? (AWT)

Jawaban : Elemen GridBagLayout diatur menurut kisi. Namun, elemen memiliki ukuran berbeda dan dapat menempati lebih dari satu baris atau kolom kisi. Selain itu, baris dan kolom mungkin memiliki ukuran yang berbeda.

Pertanyaan : Apa keuntungan yang diberikan oleh pengelola tata letak Java dibandingkan sistem jendela tradisional? (AWT)

Jawaban : Java menggunakan pengelola layout untuk meletakkan komponen secara konsisten di semua platform windowing. Karena pengelola tata letak Java tidak terikat pada ukuran dan pemosisian absolut, mereka dapat mengakomodasi perbedaan khusus platform di antara sistem windowing.

Pertanyaan : Apa perbedaan antara metode paint() dan repaint()? (AWT)

Jawaban : Metode paint() mendukung lukisan melalui objek Grafik. Metode repaint() digunakan untuk menyebabkan paint() dipanggil oleh thread lukisan AWT.

Pertanyaan : Bagaimana kelas Checkbox digunakan untuk membuat tombol radio? (AWT)

Jawaban : Dengan mengaitkan objek Checkbox dengan CheckboxGroup

Pertanyaan : Apa perbedaan antara Choice dan List? (AWT)

Jawaban : Pilihan ditampilkan dalam bentuk ringkas yang mengharuskan Anda menariknya ke bawah untuk melihat daftar pilihan yang tersedia. Hanya satu item yang dapat dipilih dari sebuah Pilihan. Daftar dapat ditampilkan sedemikian rupa sehingga beberapa item Daftar terlihat. Daftar mendukung pemilihan satu atau lebih item Daftar.

Pertanyaan : Antarmuka apa yang diperluas oleh event listener AWT? (AWT)

Jawaban : Semua event listener AWT memperluas antarmuka java.util.EventListener.

Pertanyaan : Apa itu manajer tata letak? (AWT)

Jawaban : Manajer tata letak adalah objek yang digunakan untuk mengatur komponen dalam wadah.

Pertanyaan : Subclass Komponen mana yang digunakan untuk menggambar dan melukis? (AWT)

Jawaban : Kanvas.

Pertanyaan : Apa masalah yang dihadapi oleh programmer Java yang tidak menggunakan manajer tata letak? (AWT)

Jawaban : Tanpa manajer tata letak, programmer Java dihadapkan pada penentuan bagaimana GUI mereka akan ditampilkan di beberapa sistem windowing dan menemukan ukuran dan pemosisian umum yang akan bekerja dalam batasan yang diberlakukan oleh setiap sistem windowing.

Pertanyaan : Apa perbedaan antara Scrollbar dan ScrollPane? (Ayunan)

Jawaban : Scrollbar adalah Komponen, tetapi bukan Container. ScrollPane adalah Container. ScrollPane menangani acara sendiri dan melakukan penggulirannya sendiri.

Pertanyaan : Apa itu Collections API? (Java util)

Jawaban : Collections API adalah sekumpulan kelas dan antarmuka yang mendukung operasi pada kumpulan objek.

Pertanyaan : Apa itu antarmuka Daftar? (Java util)

Jawaban : Antarmuka Daftar menyediakan dukungan untuk koleksi objek yang dipesan.

Pertanyaan : Apa itu kelas Vector? (Java util)

Jawaban : Kelas Vector menyediakan kemampuan untuk mengimplementasikan array objek yang dapat dikembangkan.

Pertanyaan : Apa itu antarmuka Iterator? (Java util)

Jawaban : Antarmuka Iterator digunakan untuk menelusuri elemen Koleksi.

Pertanyaan : Kelas dan antarmuka java.util mana yang mendukung penanganan acara?

Jawaban : Kelas EventObject dan pemrosesan acara antarmuka dukungan EventListener.

Pertanyaan : Apa itu kelas GregorianCalendar? (Java util)

Jawaban : GregorianCalendar memberikan dukungan untuk kalender Barat tradisional.

Pertanyaan : Apa itu kelas Lokal? (Java util)

Jawaban : Kelas Lokal digunakan untuk menyesuaikan keluaran program dengan konvensi wilayah geografis, politik, atau budaya tertentu.

Pertanyaan : Apa itu kelas SimpleTimeZone? (Java util)

Jawaban : Kelas SimpleTimeZone memberikan dukungan untuk kalender Gregorian.

Pertanyaan : Apa itu antarmuka Peta? (Java util)

Jawaban : Antarmuka peta menggantikan kelas kamus JDK 1.1 dan digunakan kunci asosiasi dengan nilai.

Pertanyaan : Apa kelas peristiwa tingkat tertinggi dari model delegasi peristiwa?

Jawaban : Kelas java.util.EventObject adalah kelas tingkat tertinggi dalam hierarki kelas eventdelegation.

Pertanyaan : Apa itu antarmuka Collection? (Java util)

Jawaban : Antarmuka Koleksi menyediakan dukungan untuk implementasi tas matematika - kumpulan objek tak berurutan yang mungkin berisi duplikat.

Pertanyaan : Apa itu antarmuka Set? (Java util)

Jawaban : Antarmuka Set menyediakan metode untuk mengakses elemen himpunan matematika hingga. Set tidak mengizinkan elemen duplikat.

Pertanyaan : Apa tujuan dari metode enableEvents()? (Java util)

Jawaban : Metode enableEvents() digunakan untuk mengaktifkan acara untuk objek tertentu.

Biasanya, acara diaktifkan ketika pendengar ditambahkan ke objek untuk acara tertentu. Metode enableEvents() digunakan oleh objek yang menangani peristiwa dengan menimpa metode pengiriman peristiwa mereka.

Pertanyaan : Apa itu kelas ResourceBundle? (Java util)

Jawaban : Kelas ResourceBundle digunakan untuk menyimpan sumber daya spesifik lokal yang dapat dimuat oleh program untuk menyesuaikan tampilan program dengan lokal tertentu tempat ia dijalankan.

Pertanyaan : Apa perbedaan antara menyerah dan tidur? (Thread)

Jawaban : Ketika sebuah tugas memanggil metode yield(), ia kembali ke status siap. Ketika sebuah tugas memanggil metode sleep(), ia kembali ke status menunggu.

Pertanyaan : Ketika sebuah thread memblokir I/O, status apa yang dimasuki? (Thread)

Jawaban : Sebuah thread memasuki status menunggu ketika ia memblokir I/O.

Pertanyaan : Saat utas dibuat dan dimulai, apa status awalnya? (Thread)

Jawaban : Sebuah utas dalam keadaan siap setelah dibuat dan dimulai.

Pertanyaan : Apa yang memanggil metode run() thread? (Thread)

Jawaban : Setelah utas dimulai, melalui metode start() atau kelas Thread, JVM memanggil metode run() utas saat utas pertama kali dijalankan.

Pertanyaan : Metode apa yang dipanggil agar objek mulai dieksekusi sebagai utas terpisah? (Thread)

Jawaban : Metode start() dari kelas Thread dipanggil untuk menyebabkan objek mulai dieksekusi sebagai thread terpisah.

Pertanyaan : Apa tujuan dari metode wait(), notify(), dan notifyAll()? (Thread)

Jawaban : Metode wait(), notify(), dan notifyAll() digunakan untuk menyediakan cara yang efisien bagi thread untuk menunggu sumber daya bersama. Ketika sebuah thread menjalankan metode wait() objek, ia memasuki status menunggu. Ini hanya memasuki status siap setelah thread lain memanggil metode notify() atau notifyAll() objek.

Pertanyaan : Apa status utas tingkat tinggi? (Thread)

Jawaban : Status utas tingkat tinggi siap, berjalan, menunggu, dan mati.

Pertanyaan : Apa yang terjadi jika utas tidak bisa mendapatkan kunci pada suatu objek? (Thread)

Jawaban : Jika utas mencoba menjalankan metode tersinkronisasi atau pernyataan tersinkronisasi dan tidak dapat memperoleh kunci objek, utas memasuki status menunggu hingga kunci tersedia.

Pertanyaan : Bagaimana multithreading terjadi di komputer dengan satu CPU? (Thread)

Jawaban : Penjadwal tugas sistem operasi mengalokasikan waktu eksekusi untuk beberapa tugas. Dengan beralih cepat di antara menjalankan tugas, ini menciptakan kesan bahwa tugas dijalankan secara berurutan.

Pertanyaan : Apa yang terjadi jika Anda menjalankan metode interupsi utas saat sedang tidur atau menunggu? (Thread)

Jawaban : Ketika metode interrupt() tugas dijalankan, tugas memasuki status siap. Saat berikutnya tugas memasuki status berjalan, InterruptedException dilemparkan.

Pertanyaan: Status thread saat dijalankan? (Threads)

Jawaban: Thread yang dieksekusi sedang dalam status berjalan.

Pertanyaan: Apa tiga cara di mana utas dapat memasuki status menunggu? (Thread)

Jawaban: Sebuah utas bisa memasuki status menunggu dengan memanggil metode sleep(), dengan memblokir pada I/O, dengan tidak berhasil mencoba untuk

mendapatkan kunci objek, atau dengan memanggil metode wait() objek. Itu juga bisa memasuki keadaan menunggu dengan memanggil metode suspend() nya (deprecated).

Pertanyaan : Metode apa yang harus diterapkan oleh semua utas? (Thread)

Jawaban : Semua tugas harus mengimplementasikan metode run(), apakah itu subclass dari Thread atau mengimplementasikan antarmuka Runnable.

Pertanyaan : Apa dua cara dasar di mana kelas yang dapat dijalankan sebagai utas dapat didefinisikan? (Thread)

Jawaban : Kelas thread dapat dideklarasikan sebagai subclass dari Thread, atau dapat mengimplementasikan antarmuka Runnable.

Pertanyaan : Bagaimana Anda dapat menyimpan karakter internasional/Unicode ke dalam cookie? (JSP)

Jawaban : Salah satu caranya adalah, sebelum menyimpan cookie URLEncode-nya. URLEnocder.encoder (str); Dan gunakan URLDecoder.decode (str) ketika Anda mendapatkan cookie yang disimpan.

Pertanyaan : Apa perbedaan antara URL instance dan URLConnection instance? (Jaringan)

Jawaban : Contoh URL mewakili lokasi sumber daya, dan contoh URLConnection mewakili link untuk mengakses atau berkomunikasi dengan sumber daya di lokasi tersebut.

Pertanyaan : Apa dua kelas Socket TCP yang penting? (Jaringan)

Jawaban : Socket dan ServerSocket. ServerSocket digunakan untuk komunikasi soket dua arah normal. Kelas soket memungkinkan kita membaca dan menulis melalui soket. getInputStream() dan getOutputStream() adalah dua metode yang tersedia di kelas Socket.

Pertanyaan : Bagaimana cara memanggil Prosedur Tersimpan dari JDBC? (JDBC)

Jawaban : Langkah pertama adalah membuat objek CallableStatement. Seperti halnya objek Pernyataan dan PreparedStatement, ini dilakukan dengan objek Connection terbuka. Objek CallableStatement berisi panggilan ke prosedur tersimpan. Misalnya. CallableStatement cs = con.prepareCall ("{call SHOW_SUPPLIERS}"); ResultSet rs = cs.executeQuery(); Pertanyaan Apa objek implisit? (JSP) Jawaban Objek implisit adalah objek yang dibuat oleh wadah web dan berisi informasi yang berkaitan dengan permintaan, halaman, atau aplikasi tertentu. Mereka adalah: request response pageContext session application out config page exception

Pertanyaan : Apakah teknologi JSP dapat dikembangkan? (JSP)

Jawaban : YA. Teknologi JSP dapat dikembangkan melalui pengembangan tindakan kustom, atau tag, yang dikemas dalam pustaka tag

Pertanyaan : Apa yang memberi java sifat "tulis sekali dan jalankan di mana saja"?

Jawaban : Java dikompilasi menjadi kode byte yang merupakan bahasa perantara antara kode sumber dan kode mesin. Kode byte ini tidak spesifik platorm dan karenanya dapat diumpankan ke platform apa pun. Setelah diumpankan ke JVM, yang khusus untuk sistem operasi tertentu, kode mesin khusus platform kode dibuat sehingga membuat platform java independen.

Pertanyaan : Apa empat batu penjuru OOP? (Core java)

Jawaban : Abstraksi, Enkapsulasi, Polimorfisme dan Inheritance.

Pertanyaan : Perbedaan antara Kelas dan Objek? (Core java)

Jawaban : Kelas adalah definisi atau prototipe sedangkan objek adalah contoh atau representasi hidup dari prototipe.

Pertanyaan : Apa perbedaan antara method overriding dan overloading? (Core java)

Jawaban : Overriding adalah metode dengan nama dan argumen yang sama seperti pada induknya, sedangkan overloading adalah nama metode yang sama tetapi argumen yang berbeda.

Pertanyaan : Apa itu protokol "stateless"? (Core java)

Jawaban : Tanpa berdebat panjang lebar, secara umum diterima bahwa protokol seperti HTTP tidak memiliki kewarganegaraan yaitu tidak ada retensi status antara transaksi yang merupakan kombinasi respons permintaan tunggal.

Pertanyaan : Apa itu rangkaian konstruktor dan bagaimana cara melakukannya di Java?

Jawaban : Konstruktor objek anak selalu pertama-tama perlu membangun induknya (yang pada gilirannya memanggil konstruktor induknya.). Di Java, ini dilakukan melalui panggilan implisit ke konstruktor no-args sebagai pernyataan pertama.

Pertanyaan : Apa yang dilewatkan oleh ref dan apa nilainya? (Core java)

Jawaban : Semua argumen metode Java diteruskan oleh nilai. Namun, Java memang memanipulasi objek dengan referensi, dan semua variabel objek itu sendiri adalah referensi.

Pertanyaan : Dapatkah aplikasi berbasis RMI dan Corba berinteraksi? (RMI)

Jawaban : Ya, mereka bisa. RMI tersedia dengan IIOP sebagai protokol transport daripada JRMP.

Pertanyaan : Anda dapat membuat objek String sebagai String str = "abc"; Mengapa objek tombol tidak bisa dibuat sebagai Button bt = "abc" ;? Jelaskan (Core Java)

Jawaban : Alasan utama Anda tidak dapat membuat tombol dengan Tombol bt1 = "abc"; karena "abc" adalah string literal (sesuatu yang sedikit berbeda dari objek String, by-the-way) dan bt1 adalah objek Button. Satu-satunya objek di Java yang bisa diberi String literal adalah java.lang.String. Penting untuk dicatat bahwa Anda TIDAK memanggil konstuktor java.lang.String saat Anda mengetik String s = "abc";

Pertanyaan : Apa arti keyword "abstrak" di depan metode? Kelas? (Core java)

Jawaban : Keyword abstrak mendeklarasikan metode atau kelas. Jika suatu metode memiliki keyword abstrak di depannya, itu disebut metode abstrak. Metode abstrak hs no body. Metode abstrak hanya memiliki argumen dan tipe kembalian. Metode abstrak bertindak sebagai metode placeholder yang diimplementasikan di subclass. Kelas abstrak tidak dapat dibuat instance-nya Jika sebuah kelas dideklarasikan sebagai abstrak, tidak ada objek dari kelas tersebut yang dapat dibuat. Jika sebuah kelas berisi metode abstrak, ia harus dideklarasikan sebagai abstrak.

Pertanyaan : Berapa banyak metode yang Anda terapkan jika mengimplementasikan Antarmuka Serializable? (Core java)

Jawaban : Antarmuka Serializable hanyalah sebuah "penanda" antarmuka, tanpa metode sendiri untuk diterapkan. Antarmuka 'penanda' lainnya adalah java.rmi.Remote java.util.EventListener

Pertanyaan : Apa keuntungan praktis, jika ada, dari mengimpor kelas tertentu daripada seluruh paket (misalnya import java.net. * Versus import java.net.Socket)? (Core java)

Jawaban : Tidak ada perbedaan dalam file kelas yang dihasilkan karena hanya kelas yang benar-benar digunakan yang direferensikan oleh file kelas yang dihasilkan. Ada manfaat praktis lain untuk mengimpor kelas tunggal, dan ini muncul ketika dua (atau lebih) paket memiliki kelas dengan nama yang sama.

Ambil java.util.Timer dan javax.swing.Timer, misalnya. Jika saya mengimpor java.util. * Dan javax.swing. * Dan kemudian mencoba menggunakan "Timer", saya mendapatkan error saat mengkompilasi (nama kelas ambigu di antara kedua paket).

Katakanlah apa yang sebenarnya Anda inginkan adalah kelas javax.swing.Timer, dan satu-satunya kelas yang Anda rencanakan untuk digunakan di java.util adalah Collection dan HashMap. Dalam kasus ini, beberapa orang akan lebih memilih untuk mengimpor java.util.Collection dan mengimpor java.util.HashMap daripada mengimpor java.util. *. Ini sekarang akan memungkinkan mereka untuk menggunakan Timer, Collection, HashMap, dan kelas javax.swing lainnya tanpa menggunakan nama kelas yang sepenuhnya memenuhi syarat di.

Pertanyaan : Apa perbedaan antara independensi data logis dan independensi data fisik? (Core java)

Jawaban : Independensi Data Logis - yang berarti kekebalan skema eksternal terhadap perubahan dalam skema konseptual. Kemandirian Data Fisik - yang berarti kekebalan skema konseptual terhadap perubahan skema internal.

Pertanyaan : Apa yang dimaksud dengan pengecualian yang ditentukan pengguna? (Core java)

Jawaban : Terlepas dari pengecualian yang telah ditentukan di pustaka paket Java, pengguna dapat menentukan kelas pengecualiannya sendiri dengan memperluas kelas Exception.

BAB 16
PERTANYAAN DAN JAWABAN WAWANCARA III

Latihan Catatan Serius

Beberapa (Sangat Sedikit) Pertanyaan Berulang, Tapi Jangan Khawatir, Saya sebutkan dalam kaitannya dengan beberapa topik lain.

- *Apa itu variabel transien?*

Jawaban : Variabel transien adalah variabel yang tidak dapat diserialkan.

- *Penampung mana yang menggunakan Layout perbatasan sebagai tata letak defaultnya?*

Jawaban : Jendela, kelas Frame dan Dialog menggunakan tata letak perbatasan sebagai tata letak default mereka.

- *Mengapa utas memblokir I/O?*

Jawaban : Threads memblokir pada I/O (yang memasuki status menunggu) sehingga thread lain dapat dieksekusi saat Operasi I/O dilakukan.

- *Bagaimana Observer dan Observable digunakan?*

Jawaban : Objek yang merupakan subclass dari kelas Observable memelihara daftar pengamat. Saat objek Observable diperbarui, ia memanggil metode update() dari setiap pengamatnya untuk memberi tahu pengamat bahwa statusnya telah berubah. Antarmuka Observer diimplementasikan oleh objek yang mengamati objek Observable.

- *Apa itu sinkronisasi dan mengapa itu penting?*

Jawaban : Sehubungan dengan multithreading, sinkronisasi adalah kemampuan untuk mengontrol akses beberapa utas ke sumber daya bersama. Tanpa sinkronisasi, ada kemungkinan satu utas memodifikasi objek bersama sementara utas lain sedang dalam proses menggunakan atau memperbarui nilai objek itu. Ini sering kali menyebabkan kesalahan yang signifikan.

- *Bisakah kunci diperoleh di kelas?*

Jawaban : Ya, kunci dapat diperoleh di kelas. Kunci ini diperoleh pada objek Kelas.

- *Apa yang baru dengan metode stop(), suspend() dan resume() di JDK 1.2?*

Jawaban: Metode stop(), suspend() dan resume() sudah tidak digunakan lagi di JDK 1.2.

- *Apakah null sebuah keyword?*

Jawaban: Nilai null bukan keyword.

- *Berapa ukuran komponen yang disukai?*

Jawaban : Ukuran yang disukai dari sebuah komponen adalah ukuran komponen minimum yang memungkinkan komponen untuk ditampilkan secara normal.

- *Metode apa yang digunakan untuk menentukan tata letak penampung?*

Jawaban: Metode setLayout() digunakan untuk menetapkan tata letak wadah.

- *Penampung mana yang menggunakan FlowLayout sebagai tata letak defaultnya?*

Jawaban: Kelas Panel dan Applet menggunakan FlowLayout sebagai tata letak default mereka.

- *Status apa yang dimasuki thread saat menghentikan pemrosesannya?*

Jawaban : Saat thread menghentikan pemrosesannya, thread memasuki status mati.

- *Apa itu Collections API?*

Jawaban : Collections API adalah sekumpulan kelas dan antarmuka yang mendukung operasi pada koleksi objek.

- *Karakter mana yang dapat digunakan sebagai karakter kedua dari sebuah pengenal, tetapi tidak sebagai karakter pertama dari sebuah pengenal?*

Jawaban : Digit 0 sampai 9 tidak dapat digunakan sebagai karakter pertama dari sebuah identifier tetapi mereka dapat digunakan setelah karakter pertama dari sebuah identifier.

- *Apa itu antarmuka Daftar?*

Jawaban : Antarmuka Daftar menyediakan dukungan untuk koleksi objek yang dipesan.

- *Bagaimana Java menangani overflows dan underflows integer?*

Jawaban : Ini menggunakan byte urutan rendah dari hasil yang dapat sesuai dengan ukuran jenis yang diizinkan oleh operasi.

- *Apa itu kelas Vektor?*

Jawaban : Kelas Vector menyediakan kemampuan untuk mengimplementasikan array objek yang dapat dikembangkan.

- *Pengubah apa yang dapat digunakan dengan inner class yang merupakan anggota kelas luar?*

Jawaban : A inner class (non-lokal) dapat dideklarasikan sebagai publik, dilindungi, pribadi, statis, final, atau abstrak.

- *Apa itu antarmuka Iterator?*

Jawaban : Antarmuka Iterator digunakan untuk menelusuri elemen-elemen Collection.

- *Apa perbedaan antara operator >> dan >>>?*

Jawaban : Operator >> membawa sedikit tanda saat menggeser ke kanan. Bit >>> pengisian nol yang telah digeser keluar.

- *Metode kelas Komponen mana yang digunakan untuk mengatur posisi dan ukuran komponen?*

Jawaban: setBounds()

- *Berapa banyak bit yang digunakan untuk merepresentasikan karakter Unicode, ASCII, UTF-16, dan UTF-8?*

Jawaban : Unicode membutuhkan 16 bit dan ASCII membutuhkan 7 bit. Meskipun himpunan karakter ASCII hanya menggunakan 7 bit, biasanya direpresentasikan sebagai 8 bit. UTF-8 mewakili karakter yang menggunakan pola 8, 16, dan 18 bit. UTF-16 menggunakan pola bit 16-bit dan yang lebih besar.

- *Apa perbedaan antara yielding dan sleeping?*

Jawaban : Ketika sebuah tugas memanggil metode yield(), ia kembali ke status siap. Ketika sebuah tugas memanggil metode sleep(), ia kembali ke status menunggu.

- *Kelas dan antarmuka java.util mana yang mendukung penanganan acara?*

Jawaban: Kelas EventObject dan pemrosesan acara antarmuka dukungan EventListener.

- *Apakah sizeof sebuah keyword?*

Jawaban Ukuran operator bukanlah keyword.

- *Apa itu wrapper classes?*

Jawaban : Wrapper classes adalah kelas yang memungkinkan tipe primitif untuk diakses sebagai objek.

- *Apakah Garbage Collection menjamin bahwa program tidak akan kehabisan memori?*

Jawaban : Garbage collection tidak menjamin bahwa program tidak akan kehabisan memori. Ada kemungkinan program menggunakan sumber daya memori lebih cepat daripada sampah yang dikumpulkan. Mungkin juga program membuat objek yang tidak tunduk pada Garbage Collection.

- *Pembatasan apa yang ditempatkan di lokasi pernyataan paket dalam file kode sumber?*

Jawaban : Pernyataan paket harus muncul sebagai baris pertama dalam file kode sumber (tidak termasuk baris kosong dan komentar).

- *Dapatkah metode objek finalize()dipanggil sambil dijangkau?*

Jawaban: Metode objek finalize()tidak bisa dipanggil oleh pengumpul sampah saat objek masih bisa dijangkau. Namun, metode finalize() objek mungkin dipanggil oleh objek lain.

- *Apa superclass langsung dari kelas Applet?*

Jawaban : Panel.

- *Apa perbedaan antara penjadwalan preemptive dan pembagian waktu?*

Jawaban : Di bawah penjadwalan preemptive, tugas prioritas tertinggi dijalankan hingga memasuki status menunggu atau mati atau tugas dengan prioritas lebih tinggi muncul. Di bawah pemotongan waktu, tugas dijalankan untuk potongan waktu yang telah ditentukan sebelumnya dan kemudian memasukkan kembali kumpulan tugas yang siap. Penjadwal kemudian menentukan tugas mana yang harus dijalankan selanjutnya, berdasarkan prioritas dan faktor lainnya.

- *Sebutkan tiga subclass Komponen yang mendukung pengecatan.*

Jawaban : Kelas Canvas, Frame, Panel, dan Applet mendukung lukisan.

- *Nilai apa yang dikembalikan readLine() ketika telah mencapai akhir file?*

Jawaban: Metode readLine() mengembalikan null ketika telah mencapai akhir file.

- *Apa superclass langsung dari kelas Dialog?*

Jawaban :Window

- *Apa itu kliping?*

Jawaban : Clipping adalah proses membatasi operasi cat pada area atau bentuk tertentu.

- *Apa metode asli?*

Jawaban : Metode native adalah metode yang diimplementasikan dalam bahasa selain Java.

- *Bisakah pernyataan for loop tanpa batas?*

Jawaban: Ya, pernyataan for bisa berulang tanpa batas. Misalnya, pertimbangkan hal berikut: for (;;);

- *Apa urutan prioritas dan asosiativitas, dan bagaimana penggunaannya?*

Jawaban: Urutan prioritas menentukan urutan di mana operator dievaluasi dalam ekspresi. Keterkaitan menentukan apakah ekspresi dievaluasi dari kiri ke kanan atau kanan ke kiri.

- *Ketika sebuah thread memblokir I/O, status apa yang dimasuki?*

Jawaban : Sebuah thread memasuki status menunggu ketika ia memblokir di I/O.

- *Untuk nilai apa variabel tipe String diinisialisasi secara otomatis?*

Jawaban : Nilai default dari tipe String adalah null.

- *Apa aturan catch atau decare untuk deklarasi metode?*

Jawaban: Jika pengecualian yang dicentang dapat dilemparkan ke dalam bagan metode, metode tersebut harus menangkap pengecualian tersebut atau mendeklarasikannya dalam klausa lemparannya.

- *Apa perbedaan antara MenuItem dan CheckboxMenuItem?*

Jawaban: Kelas CheckboxMenuItem memperluas kelas MenuItem untuk mendukung item menu yang dapat dicentang atau tidak dicentang.

- *Apaprioritas tugas dan bagaimana itu digunakan dalam penjadwalan?*

Jawaban: Prioritas tugas adalah nilai integer yang mengidentifikasi urutan relatif di mana tugas tersebut harus dijalankan sehubungan dengan tugas lainnya. Penjadwal mencoba menjadwalkan tugas dengan prioritas lebih tinggi sebelum tugas dengan prioritas lebih rendah.

- *Kelas apa yang berada di puncak hierarki peristiwa AWT?*

Jawaban: Kelas java.awt.AWTEvent adalah kelas tingkat tertinggi dalam hierarki eventclass AWT.

- *Saat utas dibuat dan dimulai, apa status awalnya?*

Jawaban: Sebuah utas dalam keadaan siap setelah dibuat dan dimulai.

- *Bisakah kelas anonim dideklarasikan sebagai menerapkan antarmuka dan memperluas kelas?*

Jawaban: Kelas anonim dapat mengimplementasikan antarmuka atau memperluas superclass, tetapi mungkin tidak dideklarasikan untuk melakukan keduanya.

- ***Berapa** kisaran tipe pendek?*

Jawaban: Kisaran tipe pendek adalah - (2 ^ 15) hingga 2 ^ 15 - 1.

- *Berapakah kisaran tipe karakter?*

Jawaban: Kisaran tipe karakter adalah 0 hingga 2 ^ 16 - 1.

- *Di paket mana yang ditentukan sebagian besar peristiwa AWT yang mendukung model delegasi peristiwa?*

Jawaban: Sebagian besar peristiwa terkait AWT model delegasi peristiwa ditentukan dalam paket java.awt.event. Kelas AWTEvent didefinisikan dalam paket java.awt.

- *Apa superclass langsung dari Menu?*

Jawaban: MenuItem.

- *Apa tujuan finalisasi?*

Jawaban: Tujuan finalisasi adalah untuk memberikan kesempatan kepada objek yang tidak terjangkau untuk melakukan pemrosesan pembersihan sebelum objek tersebut dikumpulkan sampahnya.

- *Kelas mana yang merupakan superclass langsung dari MenuComponent*

Jawaban: class.Object

- *Apa yang memanggil metode run() thread?*

Jawaban: Setelah utas dimulai, melalui metode start() atau kelas Thread, JVM memanggil metode run() utas saat utas pertama kali dijalankan.

- *Apa perbedaan antara Boolean & Operator dan Operator &&?*

Jawaban: Jika ekspresi yang melibatkan operator Boolean & dievaluasi, kedua operan dievaluasi. Kemudian operator & diterapkan ke operan. Saat ekspresi yang melibatkan operator && dievaluasi, operan pertama dievaluasi. Jika operan pertama mengembalikan nilai true maka operan kedua dievaluasi. Operator && kemudian diterapkan ke operan pertama dan kedua. Jika operan pertama bernilai false, evaluasi operan kedua akan dilewati.

- *Sebutkan tiga subclass dari kelas Komponen.*

Jawaban: Box.Filler, Button, Canvas, Checkbox, Choice, Container, Label, List, Scrollbar, atau TextComponent

- *Apa itu kelas GregorianCalendar?*

Jawaban: GregorianCalendar memberikan dukungan untuk kalender Barat tradisional.

- *Metode Kontainer mana yang digunakan untuk membuat kontainer yang ditata dan ditampilkan kembali?*

Jawaban: validate()

- *Apa tujuan dari kelas Runtime?*

Jawaban: Tujuan kelas runtime adalah untuk menyediakan akses ke sistem runtime Java.

- *Berapa kali metode finalize() objek dipanggil oleh pengumpul sampah?*

Jawaban: Metode finalize() objek hanya dapat dipanggil sekali oleh pengumpul sampah.

- *Apa tujuan dari klausa akhirnya dari pernyataan coba-tangkap-akhirnya?*

Jawaban: Klausa akhirnya digunakan untuk memberikan kemampuan untuk mengeksekusi kode tidak peduli apakah pengecualian dilempar atau ditangkap atau tidak.

- *Apa tipe argumen dari metode main() program?*

Jawaban: Metode main() program mengambil argumen dari tipe String [].

- *Operator Java manakah yang benar asosiatif?*

Jawaban: Operator = adalah asosiatif yang benar.

- *Apa itu kelas Lokal?*

Jawaban: Kelas Lokal digunakan untuk menyesuaikan keluaran program dengan konvensi wilayah geografis, politik, atau budaya tertentu.

- *Bisakah nilai ganda diubah menjadi satu byte?*

Jawaban: Ya, nilai ganda dapat diubah menjadi satu byte.

- *Apa perbedaan antara pernyataan break dan pernyataan lanjutan?*

Jawaban: Pernyataan istirahat menghasilkan penghentian pernyataan yang berlaku (switch, for, do, or while). Pernyataan lanjutkan digunakan untuk mengakhiri iterasi pengulangan saat ini dan mengembalikan kontrol ke pernyataan pengulangan.

- *Apa yang harus dilakukan kelas untuk mengimplementasikan antarmuka?*

Jawaban: Ini harus menyediakan semua metode dalam antarmuka dan mengidentifikasi antarmuka dalam klausul implementasinya.

- *Metode apa yang dipanggil untuk menyebabkan objek mulai dieksekusi sebagai utas terpisah?*

Jawaban: Metode start() dari kelas Thread dipanggil untuk menyebabkan objek mulai dieksekusi sebagai thread terpisah.

- *Sebutkan dua subclass dari kelas TextComponent.*

Jawaban: TextField dan TextArea

- *Apa keuntungan dari model event-delegation dibandingkan model eventinheritance sebelumnya?*

Jawaban: Model event-delegation memiliki dua keunggulan dibandingkan model eventinheritance. Pertama, ini memungkinkan penanganan peristiwa untuk ditangani oleh objek selain yang menghasilkan peristiwa (atau penampungnya). Ini memungkinkan pemisahan yang bersih antara desain komponen dan penggunaannya. Keuntungan lain dari model event-delegation adalah model ini bekerja jauh lebih baik dalam aplikasi di mana banyak peristiwa dihasilkan. Peningkatan kinerja ini disebabkan oleh fakta bahwa model event-delegation tidak harus berulang kali memproses kejadian yang tidak tertangani, seperti halnya model event-inheritance.

- *Kontainer mana yang mungkin memiliki MenuBar?*

Jawaban: Bingkai.

- *Bagaimana koma digunakan di bagian inisialisasi dan iterasi dari pernyataan for?*

Jawaban: Koma digunakan untuk memisahkan beberapa pernyataan dalam bagian inisialisasi dan iterasi dari pernyataan for.

- *Apa tujuan dari metode wait(), notify(), dan notifyAll()?*

Jawaban: Metode wait(), notify(), dan notifyAll() digunakan untuk menyediakan cara yang efisien bagi thread untuk menunggu sumber daya bersama. Ketika sebuah thread menjalankan metode wait() objek, ia memasuki status menunggu. Ini hanya memasuki status siap setelah thread lain memanggil metode notify() atau notifyAll() objek.

- *Apa itu metode abstrak?*

Jawaban: Metode abstrak adalah metode yang implementasinya ditangguhkan ke subclass.

- *Bagaimana nama file kode sumber Java?*

Jawaban: File kode sumber Java menggunakan nama kelas atau antarmuka publik yang ditentukan di dalam file. File kode sumber dapat berisi paling banyak satu kelas atau antarmuka publik. Jika kelas atau antarmuka publik didefinisikan dalam file kode sumber, maka file kode sumber harus menggunakan nama kelas atau antarmuka publik. Jika tidak ada kelas atau antarmuka publik yang didefinisikan dalam file kode sumber, maka file tersebut harus menggunakan nama yang berbeda dari kelas dan antarmukanya. File kode sumber menggunakan ekstensi .java.

- *Apa hubungan antara kelas Canvas dan kelas Graphics?*

Jawaban: Objek Canvas menyediakan akses ke objek Graphics melalui metode paint().

- *Apa status utas tingkat tinggi?*

Jawaban: Status utas tingkat tinggi siap, berjalan, menunggu, dan mati.

- *Nilai apa yang dikembalikan read() ketika telah mencapai akhir file?*

Jawaban: Metode read() mengembalikan -1 ketika telah mencapai akhir file.

- *Bisakah objek Byte dilemparkan ke nilai ganda?*

Jawaban: Tidak, sebuah objek tidak dapat diubah menjadi nilai primitif.

- *Apa perbedaan antara inner class statis dan non-statis?*

Jawaban: Inner class non-statis mungkin memiliki instance objek yang terkait dengan instance kelas luar kelas. Inner class statis tidak memiliki instance objek apa pun.

- *Apa perbedaan antara kelas String dan StringBuffer?*

Jawaban: Objek string adalah konstanta. Objek StringBuffer tidak.

- *Jika sebuah variabel dideklarasikan sebagai private, di manakah variabel tersebut dapat diakses?*

Jawaban: Variabel privat hanya dapat diakses di dalam kelas yang dideklarasikan.

- *Apa kunci sebuah objek dan objek mana yang memiliki kunci?*

Jawaban: Kunci objek adalah mekanisme yang digunakan oleh beberapa utas untuk mendapatkan akses yang disinkronkan ke objek. Sebuah utas dapat menjalankan metode tersinkronisasi dari suatu objek hanya setelah ia memperoleh kunci objek tersebut. Semua objek dan kelas memiliki kunci. Kunci kelas diperoleh pada objek Kelas kelas.

- *Apa itu kelas Kamus?*

Jawaban: Kelas Dictionary menyediakan kemampuan untuk menyimpan pasangan nilai kunci.

- *Bagaimana elemen-elemen BorderLayout diatur?*

Jawaban: Elemen BorderLayout disusun di perbatasan (Utara, Selatan, Timur, dan Barat) dan di tengah container.

- *Apa operator% itu?*

Jawaban: Ini disebut sebagai modulo atau operator sisa. Ini mengembalikan sisa pembagian operan pertama dengan operan kedua.

- *Kapan referensi objek dapat dikirim ke referensi antarmuka?*

Jawaban: Referensi objek dilemparkan ke referensi antarmuka ketika objek mengimplementasikan antarmuka yang direferensikan.

- *Apa perbedaan antara Jendela dan Bingkai?*

Jawaban: Kelas Frame memperluas Window untuk menentukan jendela aplikasi utama yang dapat memiliki menu bar.

- *Kelas mana yang diperluas oleh semua kelas lainnya?*

Jawaban: Kelas Object diperluas oleh semua kelas lainnya.

- *Apakah sebuah objek dapat dikumpulkan dari sampah selama masih dapat dijangkau?*

Jawaban : Objek yang dapat dijangkau tidak dapat dikumpulkan dari sampah. Hanya objek yang tidak terjangkau yang dapat dikumpulkan sampah ..

- *Apakah operator terner tertulis x: y? z atau x? y: z?*

Jawaban: Ada tertulis x? y: z.

- *Apa perbedaan antara kelas Font dan FontMetrics?*

Jawaban: Kelas FontMetrics digunakan untuk mendefinisikan properti khusus implementasi, seperti naik dan turun, dari objek Font.

- *Bagaimana pembulatan dilakukan di bawah pembagian integer?*

Jawaban : Bagian pecahan dari hasil tersebut dipotong. Ini dikenal sebagai pembulatan menuju nol.

- *Apa yang terjadi jika utas tidak bisa mendapatkan kunci pada suatu objek?*

Jawaban: Jika utas mencoba menjalankan metode tersinkronisasi atau pernyataan tersinkronisasi dan tidak dapat memperoleh kunci objek, utas memasuki status menunggu hingga kunci tersedia.

- *Apa perbedaan antara hierarki kelas Reader/Writer dan hierarki kelas InputStream/OutputStream?*

Jawaban: Hierarki kelas Reader/Writer berorientasi pada karakter, dan hierarki kelas InputStream/OutputStream berorientasi pada byte.

- *Kelas pengecualian apa yang dapat ditangkap oleh klausa catch?*

Jawaban: Klausa catch dapat menangkap pengecualian apa pun yang mungkin ditetapkan ke tipe Throwable. Ini termasuk tipe Error dan Exception.

- *Jika sebuah kelas dideklarasikan tanpa pengubah akses, di mana kelas tersebut dapat diakses?*

Jawaban: Kelas yang dideklarasikan tanpa pengubah akses dikatakan memiliki akses paket. Ini berarti bahwa kelas tersebut hanya dapat diakses oleh kelas dan antarmuka lain yang ditentukan dalam paket yang sama.

- *Apa itu kelas SimpleTimeZone?*

Jawaban: Kelas SimpleTimeZone memberikan dukungan untuk kalender Gregorian.

- *Apa itu Map Interface?*

Jawaban: Map Interface menggantikan kelas kamus JDK 1.1 dan digunakan kunci asosiasi dengan nilai.

- *Apakah kelas mewarisi konstruktor superclass-nya?*

Jawaban: Kelas tidak mewarisi konstruktor dari salah satu kelas supernya.

- *Untuk pernyataan mana yang masuk akal menggunakan label?*

Jawaban: Satu-satunya pernyataan yang masuk akal untuk menggunakan label adalah pernyataan yang dapat menyertakan pernyataan istirahat atau lanjutkan.

- *Apa tujuan system class?*

Jawaban: Tujuan system class adalah untuk menyediakan akses ke sumber daya sistem.

- *Metode TextComponent mana yang digunakan untuk menyetel TextComponent ke status hanya baca?*

Jawaban: setEditable()

- *Bagaimana elemen CardLayout diatur?*

Jawaban: Elemen CardLayout ditumpuk, satu di atas yang lain, seperti setumpuk kartu.

- Apakah && = merupakan operator Java yang valid?

Jawaban: Tidak.

- *Sebutkan delapan tipe Java primitif.*

Jawaban : Kedelapan tipe primitif adalah byte, char, short, int, long, float, double, dan boolean.

- *Kelas mana yang harus Anda gunakan untuk memperoleh informasi desain tentang suatu objek?*

Jawaban: Kelas Class digunakan untuk memperoleh informasi tentang desain suatu objek.

- *Apa hubungan antara clipping dan repainting?*

Jawaban: Ketika sebuah jendela dicat ulang oleh thread lukisan AWT, itu mengatur daerah kliping ke area jendela yang membutuhkan pengecatan ulang.

- *Apakah "abc" adalah nilai primitif?*

Jawaban: String literal "abc" bukanlah nilai primitif. Ini adalah objek String.

- *Apa hubungan antara antarmuka pendengar peristiwa dan kelas adaptor peristiwa?*

Jawaban: Antarmuka pendengar peristiwa menentukan metode yang harus diterapkan oleh penangan peristiwa untuk jenis peristiwa tertentu. Adaptor peristiwa menyediakan implementasi default dari antarmuka pemroses peristiwa.

- *Batasan apa yang ditempatkan pada nilai setiap kasus pernyataan switch?*

Jawaban: Selama kompilasi, nilai dari setiap kasus pernyataan switch harus dievaluasi ke nilai yang dapat dipromosikan ke nilai int.

- *Pengubah apa yang dapat digunakan dengan deklarasi antarmuka?*

Jawaban: Sebuah antarmuka dapat dideklarasikan sebagai publik atau abstrak.

- *Apakah sebuah kelas merupakan subkelasnya sendiri?*

Jawaban: Kelas adalah subkelas itu sendiri.

- *Apa kelas acara tingkat tertinggi dari model delegasi acara?*

Jawaban: Kelas java.util.EventObject adalah kelas tingkat tertinggi dalam hierarki *kelas eventdelegation*

.

- *Acara apa yang dihasilkan dari mengklik tombol?*

Jawaban: Acara ActionEvent dihasilkan sebagai hasil dari mengklik sebuah tombol.

- *Bagaimana komponen GUI menangani acaranya sendiri?*

Jawaban: Sebuah komponen dapat menangani kejadiannya sendiri dengan mengimplementasikan antarmuka pemroses kejadian yang diperlukan dan menambahkan dirinya sendiri sebagai pemroses kejadiannya sendiri.

- *Apa perbedaan antara pernyataan while dan pernyataan do?*

Jawaban: Pernyataan while memeriksa awal pengulangan untuk melihat apakah pengulangan berikutnya harus terjadi. Pernyataan do memeriksa di akhir pengulangan untuk melihat apakah pengulangan berikutnya dari pengulangan harus terjadi. Pernyataan do akan selalu mengeksekusi badan loop setidaknya sekali.

- *Bagaimana elemen GridBagLayout diatur?*

Jawaban: Elemen GridBagLayout diatur menurut kisi. Namun, elemen memiliki ukuran berbeda dan dapat menempati lebih dari satu baris atau kolom kisi. Selain itu, baris dan kolom mungkin memiliki ukuran yang berbeda.

- *Apa keuntungan yang diberikan oleh manajer layout Java dibandingkan sistem windowing tradisional?*

Jawaban: Java menggunakan pengelola layout untuk meletakkan komponen secara konsisten di semua platform windowing. Karena pengelola tata letak Java tidak terikat pada ukuran dan pemosisian absolut, mereka dapat mengakomodasi perbedaan khusus platform di antara sistem windowing.

- *Apa itu antarmuka Collection?*

Jawaban: Antarmuka Koleksi menyediakan dukungan untuk implementasi tas matematika - kumpulan objek tak berurutan yang mungkin berisi duplikat.

- *Pengubah apa yang dapat digunakan dengan inner class lokal?*

Jawaban: Kelas batin lokal mungkin bersifat final atau abstrak.

- *Apa perbedaan antara Variabel Statis dan Non-Statis?*

Jawaban: Variabel statis dikaitkan dengan kelas secara keseluruhan daripada dengan instance kelas tertentu. Variabel non-statis mengambil nilai unik dengan setiap instance objek.

- *Apa perbedaan antara metode paint() dan repaint()?*

Jawaban: Metode paint() mendukung lukisan melalui objek Grafik. Metode repaint() digunakan untuk menyebabkan paint() dipanggil oleh thread lukisan AWT.

- *Apa tujuan dari kelas File?*

Jawaban: Kelas File digunakan untuk membuat objek yang menyediakan akses ke file dan direktori sistem file lokal.

- *Bisakah pengecualian dicabut kembali?*

Jawaban: Ya, pengecualian dapat dicabut kembali.

- *Metode Matematika mana yang digunakan untuk menghitung nilai absolut sebuah angka?*

Jawaban: Metode abs() digunakan untuk menghitung nilai absolut.

- *Bagaimana multithreading terjadi di komputer dengan satu CPU?*

Jawaban: Penjadwal tugas sistem operasi mengalokasikan waktu eksekusi untuk beberapa tugas. Dengan beralih cepat di antara menjalankan tugas, ini menciptakan kesan bahwa tugas dijalankan secara berurutan.

- *Kapan kompilator menyediakan konstruktor default untuk kelas?*

Jawaban: Kompilator menyediakan konstruktor default untuk kelas jika tidak ada konstruktor lain yang disediakan.

- *Kapan klausa akhirnya dari pernyataan coba-tangkap-akhirnya dieksekusi?*

Jawaban: Klausa terakhir dari pernyataan coba-tangkap-akhirnya selalu dijalankan kecuali utas eksekusi dihentikan atau pengecualian terjadi dalam eksekusi klausa akhirnya.

- *Kelas manakah yang merupakan superclass langsung dari kelas Container?*

Jawaban: Komponen.

- *Jika suatu metode dideklarasikan sebagai dilindungi, di manakah metode tersebut dapat diakses?*

Jawaban: Metode yang dilindungi hanya dapat diakses oleh kelas atau antarmuka dari paket yang sama atau dengan subkelas kelas di mana ia dideklarasikan.

- *Bagaimana kelas Kotak centang digunakan untuk membuat tombol radio?*

Jawaban: Dengan mengaitkan objek Checkbox dengan CheckboxGroup.

- *Karakter huruf non-Unicode mana yang dapat digunakan sebagai karakterpertama pengenal?*

Jawaban: Karakter huruf non-Unicode $ dan _ mungkin muncul sebagai karakter pertama pengenal.

- *Batasan apa yang diterapkan pada metode overloading?*

Jawaban: Dua metode mungkin tidak memiliki nama dan daftar argumen yang sama tetapi tipe kembalian yang berbeda.

- *Apa yang terjadi jika Anda menjalankan metode interupsi utas saat sedang* tidur atau menunggu?

Jawaban: Ketika metode interrupt() tugas dijalankan, tugas memasuki status siap. Saat berikutnya tugas memasuki status berjalan, InterruptedException dilemparkan.

- *Apa itu casting?*

Jawaban: Ada dua jenis casting, casting antara tipe numerik primitif dan casting di antara referensi objek. Transmisi antar tipe numerik digunakan untuk mengonversi nilai yang lebih besar, seperti nilai ganda, menjadi nilai yang lebih kecil, seperti nilai byte. Transmisi antar referensi objek digunakan untuk merujuk ke objek dengan referensi kelas, antarmuka, atau tipe array yang kompatibel.

- *Apa tipe kembalian dari metode main() program?*

Jawaban: Metode main() program memiliki tipe pengembalian void.

- *Sebutkan empat kelas Container.*

Jawaban: Jendela, Bingkai, Dialog, FileDialog, Panel, Applet, atau ScrollPane

- *Apa perbedaan antara Choice dan List?*

Jawaban: Pilihan ditampilkan dalam bentuk ringkas yang mengharuskan Anda menariknya ke bawah untuk melihat daftar pilihan yang tersedia. Hanya satu item yang dapat dipilih dari sebuah Pilihan. Daftar dapat ditampilkan sedemikian rupa sehingga beberapa item Daftar terlihat. Daftar mendukung pemilihan satu atau lebih item Daftar.

- *Kelas pengecualian apa yang dihasilkan oleh sistem run-time Java?*

Jawaban: Sistem runtime Java menghasilkan pengecualian RuntimeException dan Error.

- *Kelas apa yang memungkinkan Anda membaca objek langsung dari stream?*

Jawaban: Kelas ObjectInputStream mendukung pembacaan objek dari input stream.

- *Apa perbedaan antara variabel lapangan dan variabel lokal?*

Jawaban: Variabel lapangan adalah variabel yang dideklarasikan sebagai anggota kelas. Variabel lokal adalah variabel yang dideklarasikan secara lokal ke suatu metode.

- *Dalam kondisi apa metode finalize() objek dipanggil oleh pengumpul sampah?*

Jawaban: Pengumpul sampah memanggil metode finalize() objek ketika mendeteksi bahwa objek menjadi tidak terjangkau.

- *Bagaimana this() dan super() digunakan dengan konstruktor?*

Jawaban: this() digunakan untuk memanggil konstruktor dari kelas yang sama. super() digunakan untuk menjalankan konstruktor superclass.

- *Apa hubungan antara klausa lemparan metode dan pengecualian yang dapat dilemparkan selama eksekusi metode?*

Jawaban: Klausa lemparan metode harus mendeklarasikan pengecualian yang dicentang yang tidak ada di dalam tubuh metode.

- *Apa perbedaan antara model peristiwa JDK 1.02 dan model delegasi peristiwa yang diperkenalkan dengan JDK 1.1?*

Jawaban: Model peristiwa JDK 1.02 menggunakan peinheritance peristiwa atau pendekatan menggelegak. Dalam model ini, komponen diperlukan untuk menangani kejadiannya sendiri. Jika mereka tidak menangani acara tertentu, acara tersebut diwarisi oleh (atau menggelembung ke) wadah komponen. Kontainer kemudian menangani acara atau menggelembung ke kontainernya dan seterusnya, hingga kontainer tingkat tertinggi telah dicoba. Dalam model delegasi acara, objek tertentu ditetapkan sebagai penangan kejadian untuk komponen GUI.

Objek-objek ini mengimplementasikan antarmuka event-listener. Model event delegation lebih efisien daripada model event-inheritance karena menghilangkan pemrosesan yang diperlukan untuk mendukung penggelembungan acara yang tidak tertangani.

- *Bagaimana mungkin dua objek String dengan nilai identik tidak sama di bawah operator ==?*

Jawaban: Operator == membandingkan dua objek untuk menentukan apakah mereka adalah objek yang sama di memori. Itu mungkin. agar dua objek String memiliki nilai yang sama, tetapi terletak di area memori yang acuh tak acuh

- *Mengapa metode kelas Matematika bersifat statis?*

Jawaban: Jadi mereka dapat dipanggil seolah-olah mereka adalah perpustakaan kode matematika.

- *Metode Kotak Centang apa yang memungkinkan Anda mengetahui jika Kotak Centang dicentang*?

Jawaban: getState()

- *Keadaan apa utas saat dijalankan?*

Jawaban: Utas yang menjalankan sedang dalam keadaan berjalan.

- *Apa operand legal dari operator instanceof?*

Jawaban: Operan kiri adalah referensi objek atau nilai null dan operan kanan adalah kelas, antarmuka, atau tipe array.

- *Bagaimana elemen GridLayout diatur?*

Jawaban: Elemen tata letak GridBad memiliki ukuran yang sama dan ditata menggunakan kotak kotak.

- *Apa itu filter I/O?*

Jawaban: Filter I/O adalah objek yang membaca dari satu stream dan menulis ke stream lain, biasanya mengubah data dengan cara tertentu saat diteruskan dari satu stream ke stream lainnya.

- *Jika suatu objek dikumpulkan sampah, apakah dapat dijangkau lagi?*

Jawaban: Setelah sebuah objek dikumpulkan sampah, ia tidak ada lagi. Itu tidak bisa lagi dijangkau lagi.

- *Apa itu antarmuka Set?*

Jawaban: Antarmuka Set menyediakan metode untuk mengakses elemen himpunan matematika hingga. Set tidak mengizinkan elemen duplikat.

- *Kelas pengecualian apa yang dapat dilemparkan oleh pernyataan lemparan?*

Jawaban: Pernyataan lemparan dapat memunculkan ekspresi apa pun yang mungkin ditetapkan ke tipe Lempar.

- *Apa itu E dan PI?*

Jawaban: E adalah basis dari logaritma natural dan PI adalah nilai matematika pi.

- *Apakah keyword benar dan salah?*

Jawaban: Nilai benar dan salah bukanlah keyword.

- *Apa yang dimaksud dengan tipe pengembalian void?*

Jawaban: Jenis pengembalian kosong menunjukkan bahwa metode tidak mengembalikan nilai.

- *Apa tujuan dari metode enableEvents()?*

Jawaban: Metode enableEvents() digunakan untuk mengaktifkan acara untuk objek tertentu. Biasanya, acara diaktifkan ketika pendengar ditambahkan ke objek untuk acara tertentu. Metode enableEvents() digunakan oleh objek yang menangani peristiwa dengan menimpa metode pengiriman peristiwa mereka.

- *Apa perbedaan antara kelas File dan RandomAccessFile?*

Jawaban: Kelas File merangkum file dan direktori dari sistem file lokal. Kelas RandomAccessFile menyediakan metode yang diperlukan untuk mengakses data secara langsung yang terdapat di bagian mana pun dari file.

- *Apa yang terjadi jika Anda menambahkan nilai ganda ke String?*

Jawaban: Hasilnya adalah objek String.

- *Apa pengkodean karakter default platform Anda?*

Jawaban: Jika Anda menjalankan Java pada platform Windows Inggris, kemungkinan Cp1252. Jika Anda menjalankan platform Javaon English Solaris, kemungkinan besar 8859_1

- *Paket mana yang selalu diimpor secara default?*

Jawaban: Paket java.lang selalu diimpor secara default.

- *Antarmuka apa yang harus diterapkan objek sebelum dapat ditulis ke stream sebagai objek?*

Jawaban: Sebuah objek harus mengimplementasikan antarmuka Serializable atau Externalizable sebelum dapat ditulis ke stream sebagai objek.

- *Bagaimana ini dan super digunakan?*

Jawaban: ini digunakan untuk merujuk ke instance objek saat ini. super digunakan untuk merujuk ke variabel dan metode kelas super dari instance objek saat ini.

- *Apa tujuan Garbage Collection?*

Jawaban : Tujuan dari Garbage Collection adalah untuk mengidentifikasi dan membuang objek yang tidak lagi dibutuhkan oleh program sehingga sumber dayanya dapat diambil kembali dan digunakan kembali.

- *Apa itu unit kompilasi?*

Jawaban: Unit kompilasi adalah file kode sumber Java.

- *Antarmuka apa yang diperluas oleh pendengar acara AWT?*

Jawaban: Semua event listener AWT memperluas antarmuka java.util.EventListener.

- *Batasan apa yang diterapkan pada penggantian metode?*

Jawaban: Metode yang diganti harus memiliki nama, daftar argumen, dan tipe kembalian yang sama. Metode penggantian mungkin tidak membatasi akses metode yang ditimpanya. Metode penggantian tidak boleh menampilkan pengecualian apa pun yang mungkin tidak muncul oleh metode yang diganti.

- *Bagaimana thread yang mati dapat dimulai ulang?*

Jawaban : Sebuah thread mati tidak dapat di restart.

- *Apa yang terjadi jika pengecualian tidak tertangkap?*

Jawaban: Pengecualian yang tidak tertangkap menghasilkan metode uncaughtException() dari ThreadGroup thread yang dipanggil, yang pada akhirnya mengakibatkan penghentian program tempat ia dilempar.

- *Apa itu manajer tata letak?*

Jawaban: Manajer tata letak adalah objek yang digunakan untuk mengatur komponen dalam wadah.

- *Operasi aritmatika mana yang dapat menghasilkan ArithmeticException?*

Jawaban: Integer/dan% dapat menghasilkan ArithmeticException.

- *Apa tiga cara di mana utas dapat memasuki status menunggu?*

Jawaban: Sebuah utas bisa memasuki status menunggu dengan memanggil metode sleep(), dengan memblokir pada I/O, dengan tidak berhasil mencoba untuk mendapatkan kunci objek, atau dengan memanggil metode wait() objek. Itu juga bisa memasuki keadaan menunggu dengan memanggil metode suspend() nya (deprecated).

- *Bisakah kelas abstrak menjadi final?*

Jawaban: Kelas abstrak tidak boleh dinyatakan sebagai kelas final.

- *Apa itu kelas ResourceBundle?*

Jawaban: Kelas ResourceBundle digunakan untuk menyimpan sumber daya khusus lokal yang dapat dimuat oleh program untuk menyesuaikan tampilan program dengan lokal tertentu tempat ia dijalankan.

- *Apa yang terjadi jika pernyataan coba-tangkap-akhirnya tidak memiliki klausacatch untuk menangani pengecualian yang dilemparkan ke dalam isi pernyataan coba?*

Jawaban: Pengecualian menyebar ke tingkat pernyataan coba-tangkap berikutnya yang lebih tinggi (jika ada) atau mengakibatkan penghentian program.

- *Apa itu promosi numerik?*

Jawaban: Promosi numerik adalah konversi jenis numerik yang lebih kecil ke jenis numerik yang lebih besar, sehingga operasi integer dan floating-point dapat berlangsung. Dalam promosi numerik, nilai byte, char, dan short diubah menjadi nilai int. Nilai int juga diubah menjadi nilai panjang, jika perlu. Nilai long dan float diubah menjadi nilai ganda, sesuai kebutuhan.

- *Apa perbedaan antara Scrollbar dan ScrollPane?*

Jawaban: Scrollbar adalah Komponen, tetapi bukan Container. ScrollPane adalah Container. ScrollPane menangani acara sendiri dan melakukan penggulirannya sendiri.

- *Apa perbedaan antara kelas publik dan non-publik?*

Jawaban: Kelas publik dapat diakses di luar paketnya. Kelas non-publik tidak dapat diakses di luar paketnya.

- *Untuk nilai apa variabel tipe boolean diinisialisasi secara otomatis?*

Jawaban: Nilai default dari tipe boolean adalah salah.

- *Bisakah pernyataan percobaan bersarang?*

Jawaban: Coba pernyataan dapat diuji.

- *Apa perbedaan antara bentuk prefix dan postfix dari operator++?*

Jawaban: Formulir awalan melakukan operasi kenaikan dan mengembalikan nilai operasi kenaikan. Formulir postfix mengembalikan nilai saat ini semua ekspresi dan kemudian melakukan operasi penambahan pada nilai itu.

- *Apa tujuan blok pernyataan?*

Jawaban: Blok pernyataan digunakan untuk mengatur urutan pernyataan sebagai satu kelompok pernyataan.

- *Apa itu paket Java dan bagaimana cara menggunakannya?*

Jawaban: Paket Java adalah konteks penamaan untuk kelas dan antarmuka. Paket digunakan untuk membuat ruang nama terpisah untuk grup kelas dan antarmuka. Paket juga digunakan untuk mengatur kelas dan antarmuka terkait ke dalam satu unit API dan untuk mengontrol aksesibilitas ke kelas dan antarmuka ini.

- *Pengubah apa yang dapat digunakan dengan kelas tingkat atas?*

Jawaban: Kelas tingkat atas mungkin publik, abstrak, atau final.

- *Untuk apa kelas Objek dan Kelas digunakan?*

Jawaban: Kelas Object adalah kelas tingkat tertinggi dalam hierarki kelas Java. Kelas Kelas digunakan untuk mewakili kelas dan antarmuka yang dimuat oleh program Java.

- *Bagaimana pernyataan coba menentukan klausa catch mana yang harusdigunakan untuk menangani pengecualian?*

Jawaban: Ketika pengecualian dilemparkan ke dalam tubuh pernyataan percobaan, klausul tangkapan dari pernyataan percobaan diperiksa dalam urutan kemunculannya. Klausa catch pertama yang mampu menangani pengecualian akan dieksekusi. Klausul tangkapan yang tersisa diabaikan.

- *Bisakah objek yang tidak terjangkau menjadi dapat dijangkau lagi?*

Jawaban: Objek yang tidak dapat dijangkau dapat menjadi dapat dijangkau kembali. Ini bisa terjadi ketika metode finalize() objek dipanggil dan objek melakukan operasi yang menyebabkannya bisa diakses oleh objek yang bisa dijangkau.

- *Kapan suatu objek menjadi sasaran Garbage Collection?*

Jawaban: Sebuah objek tunduk pada Garbage Collection ketika menjadi tidak terjangkau oleh program yang mengeluarkannya.

- *Metode apa yang harus diterapkan oleh semua utas?*

Jawaban: Semua tugas harus mengimplementasikan metode run(), apakah itu subclass dari Thread atau mengimplementasikan antarmuka Runnable.

- *Metode apa yang digunakan untuk mendapatkan dan menyetel label teks yangditampilkan oleh objek Tombol?*

Jawaban: getLabel() dan setLabel()

- *Subkelas Komponen mana yang digunakan untuk menggambar dan melukis?*

Jawaban: Kanvas.

- *Apa itu metode tersinkronisasi dan pernyataan tersinkronisasi?*

Jawaban: Metode tersinkronisasi adalah metode yang digunakan untuk mengontrol akses ke suatu objek.

Sebuah thread hanya menjalankan metode tersinkronisasi setelah mendapatkan kunci untuk objek atau kelas metode tersebut. Pernyataan tersinkronisasi mirip dengan metode tersinkronisasi. Pernyataan tersinkronisasi hanya dapat dijalankan setelah utas memperoleh kunci untuk objek atau kelas yang dirujuk dalam pernyataan tersinkronisasi

- *Apa dua cara dasar di mana kelas yang dapat dijalankan sebagai utas dapat didefinisikan?*

Jawaban: Kelas thread dapat dideklarasikan sebagai subclass dari Thread, atau dapat mengimplementasikan antarmuka Runnable.

- *Apa masalah yang dihadapi oleh programmer Java yang tidak menggunakanmanajer tata letak?*

Jawaban: Tanpa manajer tata letak, programmer Java dihadapkan pada penentuan bagaimana GUI mereka akan ditampilkan di beberapa sistem windowing dan menemukan ukuran dan pemosisian umum yang akan bekerja dalam batasan yang diberlakukan oleh setiap sistem windowing.

- *Apa perbedaan antara pernyataan if dan pernyataan switch?*

Jawaban: Pernyataan if digunakan untuk memilih di antara dua alternatif. Ini menggunakan ekspresi boolean untuk memutuskan alternatif mana yang harus dieksekusi. Pernyataan switch digunakan untuk memilih di antara beberapa alternatif. Ini menggunakan ekspresi int untuk menentukan alternatif mana yang harus dieksekusi.

- *Jelaskan apa yang terjadi ketika sebuah objek dibuat di Java?*

Jawaban: Beberapa hal terjadi dalam urutan tertentu untuk memastikan objek dibangun dengan benar:

- Memori dialokasikan dari heap untuk menampung semua variabel instan dan data spesifik implementasi dari objek dan kelas supernya. Data khusus implementasi mencakup petunjuk ke data kelas dan metode.
- Variabel instance dari objek diinisialisasi ke nilai defaultnya.
- Konstruktor untuk kelas yang paling diturunkan dipanggil. Hal pertama yang dilakukan konstruktor adalah memanggil konstruktor untuk kelas supernya. Proses ini berlanjut hingga konstruktor untuk java.lang.Object dipanggil, karena java.lang.Object adalah kelas dasar untuk semua objek di java.
- Sebelum isi konstruktor dijalankan, semua penginisialisasi variabel instance dan blok inisialisasi dijalankan. Kemudian tubuh konstruktor dijalankan. Jadi, konstruktor untuk kelas dasar diselesaikan terlebih dahulu dan konstruktor untuk kelas yang paling diturunkan diselesaikan terakhir. Metode kelas dan hierarki induknya tersedia. Terakhir, alamat objek dikembalikan.
- *Pertanyaan: Di Java, Anda dapat membuat objek String seperti di bawah ini: String str = "abc"; & String str = String baru ("abc"); Mengapa objek tombol tidak dapat dibuat sebagai: Button bt = "abc". Mengapa wajib membuat objek tombol sebagai: Button bt = new Button ("abc")*
- Jawaban: String bukan hanya kelas lain di Java. Ada banyak kasus khusus di kelas String. Jika Anda perhatikan dengan hati-hati, String adalah kelas yang tidak dapat diubah dimana itu tidak berlaku untuk Button atau sebagian besar kelas lainnya. Compiler Java serta operator '+' dan '+ =' mengonversi karakter yang dikutip menjadi String. aku percaya
- Desainer Java ingin memperlakukan String sedekat mungkin dengan tipe primitif dan karenanya ada beberapa perbedaan. Penting untuk diperhatikan

bahwa Anda TIDAK memanggil konstruktor java.lang.String saat Anda mengetik String s = "abc";

- *Sebagai Contoh;*
- String x = "abc";
- String y = "abc";
- mengacu pada objek yang sama.
- Sementara
- String x1 = String baru ("abc");
- String x2 = String baru ("abc");
- mengacu pada dua objek yang berbeda.

- *Pertanyaan: Apa keuntungan dari OOP?*

Jawaban: Anda akan mendapatkan berbagai jawaban untuk pertanyaan ini tergantung pada siapa Anda bertanya. Keuntungan utama OOP adalah:

- *Kesederhanaan*: model objek perangkat lunak objek dunia nyata, sehingga kompleksitas berkurang dan struktur program sangat jelas;
- *Modularitas*: setiap objek membentuk entitas terpisah yang pekerjaan internalnya dipisahkan dari bagian lain dari sistem;
- *Modifiabilias*: mudah untuk membuat perubahan kecil dalam representasi data atau prosedur dalam program OO. Perubahan di dalam kelas tidak mempengaruhi bagian lain dari program, karena satu-satunya antarmuka publik yang dimiliki dunia luar ke kelas adalah melalui penggunaan metode;
- *Ekstensibilitas*: menambahkan fitur baru atau menanggapi perubahan lingkungan operasi dapat diselesaikan dengan memperkenalkan beberapa objek baru dan memodifikasi beberapa objek yang sudah ada;
- *Maintainabiltasy*: objek dapat dipelihara secara terpisah, membuat pencarian dan perbaikan masalah menjadi lebih mudah;
- *Re-usability*: objek dapat digunakan kembali dalam program yang berbeda.

- *Pertanyaan: Apa perbedaan utama antara Java dan C++?*

Jawaban: Perbedaan utama antara C++ dan Java adalah:

- Semuanya adalah objek di Java (Hierarki root tunggal karena semuanya diturunkan dari java.lang.Object)
- Java tidak memiliki semua aspek rumit dari C++ (Misalnya: Pointer, template, unions, operator overloading, structure dll ...)
- Promotor bahasa Java awalnya mengatakan "Tidak ada petunjuk!", Tetapi ketika banyak pemrogram mempertanyakan bagaimana Anda dapat bekerja tanpa petunjuk, promotor mulai mengatakan "Petunjuk yang dibatasi”. Anda dapat memutuskan apakah itu benar-benar sebuah pointer atau bukan. Bagaimanapun, tidak ada aritmatika pointer.
- Tidak ada perusak di Java (Garbage Collection otomatis).
- Java tidak mendukung kompilasi bersyarat (# ifdef/# tipe ifndef).
- Dukungan thread dibangun ke dalam java tetapi tidak di C++.
- Java tidak mendukung argumen default.

- Tidak ada operator resolusi cakupan: di Java. Java menggunakan titik untuk semuanya, tetapi bisa juga digunakan karena Anda hanya dapat mendefinisikan elemen di dalam kelas. Bahkan definisi metode harus selalu terjadi di dalam kelas, jadi tidak perlu ada resolusi cakupan di sana.
- Tidak ada pernyataan "goto" di java.
- Java tidak menyediakan multiple inheritance (MI), setidaknya tidak dalam pengertian yang sama seperti C++.
- Exception handlingdi java berbeda karena tidak ada destruktor.
- Java memiliki metode overloading, tetapi tidak ada operator yang kelebihan beban. Kelas String memang menggunakan operator + dan + = untuk menggabungkan string dan ekspresi String menggunakan konversi jenis otomatis, tetapi itu adalah kasus bawaan khusus.
- Java diinterpretasikan untuk sebagian besar dan karenanya platform independen.

- *Pertanyaan: Apa itu antarmuka?*

Jawaban: Antarmuka memberikan cara yang lebih canggih untuk mengatur dan mengontrol objek di sistem Anda. Keyword antarmuka membawa konsep abstrak selangkah lebih maju. Anda dapat menganggapnya sebagai kelas abstrak "murni".

Ini memungkinkan pembuat untuk menetapkan formulir untuk kelas: nama metode, daftar argumen, dan tipe kembalian, tetapi tidak ada badan metode. Antarmuka juga dapat berisi bidang. Sebuah antarmuka mengatakan:

"Seperti inilah tampilan semua kelas yang mengimplementasikan antarmuka khusus ini." Jadi, kode apa pun yang menggunakan antarmuka tertentu mengetahui metode apa yang mungkin dipanggil untuk antarmuka itu, dan itu saja.

Jadi antarmuka digunakan untuk membuat "protokol" antar kelas (beberapa bahasa pemrograman berorientasi objek memiliki keyword yang disebut protokol untuk melakukan hal yang sama).

Contoh umum tercantum di bawah ini (dari "Coe Java Professional"):

```
import java.util.*;
interface Instrument {
      int i = 5; // static & final
      // Cannot have method definitions:
      void play(); // Automatically public
      String what();
      void adjust();
}

class Wind implements Instrument {
      public void play() {
      System.out.println("Wind.play()");
      }
      public String what() { return "Wind"; }
      public void adjust() {}
}
```

- *Pertanyaan: Bagaimana Anda bisa mencapai Inheritance Ganda di Java?*

Jawaban: Mekanisme antarmuka Java dapat digunakan untuk mengimplementasikan beberapa inheritance, dengan satu perbedaan penting dari cara c++ melakukan MI: antarmuka yang diwariskan harus abstrak. Ini meniadakan kebutuhan untuk memilih di antara implementasi yang berbeda, karena dengan antarmuka tidak ada implementasi.

Example:

```
interface CanFight {
   void fight();
}

interface CanSwim {
   void swim();
}
interface CanFly {
   void fly();
}
class ActionCharacter {
   public void fight() {}
}
class Hero extends ActionCharacter implements CanFight, CanSwim,
CanFly {
   public void swim() {}
   public void fly() {}
}
```

```
interface A {
   void methodA();
}

class AImpl implements A {
   void methodA() { //do stuff }
}

interface B {
   void methodB();
}

class BImpl implements B {
   void methodB() { //do stuff }
}

class Multiple implements A, B {
   private A a = new AImpl();
   private B b = new BImpl();
   void methodA() { a.methodA(); }
   void methodB() { b.methodB(); }
}
```

Lihat di atas, Anda bahkan dapat mencapai bentuk beberapa inheritance di mana Anda dapat menggunakan * fungsionalitas * kelas daripada hanya antarmuka:

Ini benar-benar memecahkan masalah tradisional beberapa inheritance di C++ di mana bentrokan nama terjadi antara beberapa kelas dasar. Pembuat kode kelas turunan harus secara eksplisit menyelesaikan setiap bentrokan.

- *Pertanyaan: Apa perbedaan antara String Buffer dan kelas String?*

Jawaban: Buffer string mengimplementasikan urutan karakter yang bisa berubah. Buffer string seperti String, tetapi dapat dimodifikasi. Urutan berisi beberapa urutan karakter tertentu kapan saja, tetapi panjang dan konten urutan dapat diubah melalui panggilan metode tertentu.

Kelas String mewakili string karakter. Semua string literal dalam program Java, seperti "abc" adalah konstan dan diimplementasikan sebagai instance kelas ini; nilainya tidak dapat diubah setelah dibuat.

- *Pertanyaan: Jelaskan, secara umum, cara kerja pengumpul sampah java?*

Jawaban: Lingkungan runtime Java menghapus objek saat menentukan bahwa mereka tidak lagi digunakan. Proses ini dikenal sebagai Garbage Collection. Lingkungan runtime Java mendukung pengumpul sampah yang secara berkala membebaskan memori yang digunakan oleh objek yang tidak lagi diperlukan. Pengumpul sampah Java adalah pengumpul sampah mark-sweep yang memindai area memori dinamis Java untuk menemukan objek, menandai yang direferensikan.

Setelah semua kemungkinan jalur ke objek diselidiki, objek yang tidak ditandai (yaitu tidak direferensikan) diketahui sebagai sampah dan dikumpulkan (Penjelasan yang lebih lengkap tentang algoritme Garbage Collection kami mungkin adalah "Pemadatan, menandai pengumpul tangis dengan beberapa pemindaian konservatif ")

Pengumpul sampah berjalan serentak saat sistem kehabisan memori, atau sebagai tanggapan atas permintaan dari program Java. Program Java Anda bisa

meminta pengumpul sampah untuk dijalankan kapan saja dengan memanggil System.gc().

Pengumpul sampah memerlukan waktu sekitar 20 milidetik untuk menyelesaikan tugasnya, jadi, program Anda seharusnya hanya menjalankan pengumpul sampah saat tidak akan ada dampak kinerja dan program mengantisipasi periode diam yang cukup lama bagi pengumpul sampah untuk menyelesaikan tugasnya.

catatan:

Meminta agar Garbage Collection dijalankan tidak menjamin bahwa objek Anda akan dikumpulkan.

Pengumpul sampah Java berjalan secara asinkron saat sistem tidak aktif pada sistem yang memungkinkan runtime Java untuk mencatat kapan thread telah dimulai dan untuk mengganggu thread lain (seperti Windows 95).

Segera setelah utas lain aktif, pengumpul sampah diminta untuk mendapatkan status yang konsisten dan kemudian menghentikannya. (Dari “dari" Coe Java Professional "oleh Harry”).

- *Pertanyaan: Apa perbedaan utama antara metode == operatorand equals?*

Jawaban: Metode sama dengan dapat dianggap melakukan perbandingan nilai suatu objek secara mendalam, sedangkan operator == melakukan perbandingan yang dangkal. Metode equal membandingkan ke mana suatu objek menunjuk daripada pointer itu sendiri (jika kita dapat mengakui bahwa Java memiliki pointer). Tipuan ini mungkin tampak jelas bagi pemrogram C++.

Ingatlah untuk menimpa metode toString() yang disediakan oleh java.lang.Object ketika Anda ingin menggunakan metode equals() pada objek khusus seperti untuk Objek, metode toString hanya mengembalikan alamat memori dan membandingkan dua alamat memori bukanlah yang Anda inginkan ketika memeriksa kesetaraan Objek.

Kelas string hadir dengan implementasi toString() yang melakukan perbandingan karakter dengan karakter, jadi Anda tidak perlu khawatir tentang toString() saat menggunakan persamaan pada objek string.

- *Pertanyaan: Apa itu kelas abstrak dan metode abstrak?*

Jawaban: Secara sederhana berbicara kelas atau metode yang memenuhi syarat dengan keyword "abstrak" adalah kelas abstrak atau metode abstrak. Anda membuat kelas abstrak saat Anda ingin memanipulasi sekumpulan kelas melalui antarmuka umum. Semua metode kelas turunan yang cocok dengan tanda tangan deklarasi kelas dasar akan dipanggil menggunakan mekanisme pengikatan dinamis.

Jika Anda memiliki kelas abstrak, objek dari kelas itu hampir selalu tidak memiliki arti. Artinya, kelas abstrak dimaksudkan untuk hanya mengekspresikan antarmuka dan terkadang beberapa implementasi metode default, dan bukan implementasi tertentu, jadi membuat objek kelas abstrak tidak masuk akal dan tidak diizinkan (kompilasi akan memberi Anda pesan kesalahan jika Anda mencoba buat

satu). Metode abstrak adalah metode yang tidak lengkap. Ini hanya memiliki deklarasi dan tidak ada tubuh metode. Berikut adalah sintaks untuk deklarasi metode abstrak

abstrak kekosongan f();

Jika sebuah kelas berisi satu atau lebih metode abstrak, kelas tersebut harus memenuhi syarat sebagai abstrak. (Jika tidak, kompilator memberi Anda pesan kesalahan.). Dimungkinkan untuk membuat kelas sebagai abstrak tanpa menyertakan metode abstrak apa pun. Ini berguna ketika Anda memiliki kelas yang tidak memungkinkan untuk memiliki metode abstrak apa pun, namun Anda ingin mencegah setiap instance dari kelas itu. Kelas dan metode abstrak dibuat karena mereka membuat keabstrakan kelas menjadi eksplisit, dan memberi tahu pengguna dan kompiler bagaimana kelas itu dimaksudkan untuk digunakan.

For example:

```
abstract class Instrument
{
   int i; // storage allocated for each
   public abstract void play();
   public String what()
   {
        return "Instrument";
   }
     public abstract void adjust();
   }
   class Wind extends Instrument
   {
       public void play()
       {
          System.out.println("Wind.play()");
       }
       public String what()
       {
          return "Wind";
       }
       public void adjust()
       {
       }
   }
```

- *Pertanyaan: Bagaimana Anda bisa memaksa semua kelas turunan untuk mengimplementasikan metode yang ada di kelas dasar?*

Jawaban: Kelas abstrak adalah pilihan yang tepat. Antarmuka benar-benar cara yang salah. Cara terbaik untuk memikirkan sebuah Interface adalah apa adanya, kelas yang sepenuhnya abstrak. Satu perbedaan utama antara antarmuka dan kelas abstrak adalah bahwa dalam kelas abstrak Anda dapat mendefinisikan beberapa metode tetapi tidak semua (meskipun Anda masih tidak dapat membuat contoh kelas abstrak).

Dalam sebuah antarmuka Anda hanya dapat mendeklarasikan metode, tetapi Anda tidak dapat mendefinisikannya. Singkatnya, kelas abstrak adalah cara yang tepat. Ini akan memungkinkan Anda untuk meninggalkan metode tunggal yang dimaksud sebagai kelas abstrak dan memungkinkannya untuk didefinisikan oleh subkelas.

- *Pertanyaan: Java mengatakan "tulis sekali, jalankan di mana saja". Apa beberapa hal yang tidak sepenuhnya benar?*

Jawaban: Ada beberapa masalah yang mengharuskan Anda untuk melakukan porting sistem yang ditulis untuk satu OS agar berjalan di OS lain:

Anda memiliki kemampuan untuk menggunakan JNI, yang akan menggunakan pustaka asli, yang dapat bekerja secara berbeda pada sistem yang berbeda.

Anda memiliki kemampuan untuk melakukan panggilan sistem dengan Runtime.getRuntime() .exec ("<system command>") yang dapat membuat kode bergantung pada perangkat keras, mis. "ls" tidak akan berfungsi di Windows.

Anda memiliki kemampuan untuk menggunakan koordinat XY dalam Piksel alih-alih menentukan Manajer Tata Letak di Swing, yang akan mengubah tampilan GUI pada sistem yang berbeda.

Ada juga beberapa bug dalam implementasi VM pada sistem berbeda yang dapat menyebabkan program Anda berjalan dengan baik di Sun tetapi crash di HP jika program Anda mengalami bug pada VM HP yang tidak ada dalam implementasi SUN. Saya pribadi mengalami ini.

Anda harus melakukan penyetelan kinerja terpisah, mis. parameter Garbage Collection, ukuran heap awal, dll. untuk setiap OS tempat Anda menjalankan sistem, bergantung pada performa setiap OS dan perangkat keras yang menjalankannya.

Utas diterapkan secara berbeda pada setiap sistem. Misalnya, pada Windows bahkan utas prioritas rendah akan mendapatkan beberapa irisan waktu, sedangkan pada UNIX utas prioritas rendah tidak akan pernah berjalan jika utas prioritas tinggi tidak menyerah.

Ada perbedaan operasi kecil lainnya juga, mis. Anda mengirim perintah kill -S SIGQUIT pada UNIX untuk melihat thread dump, dan pada Windows Anda menekan tombol Break.

- *Pertanyaan: Apa perbedaan antara Vector dan Array? Diskusikan kelebihan dan kekurangan keduanya?*

Jawaban: Perbedaan utama antara Vector dan Array adalah:

Vektor dapat berisi objek dari berbagai jenis sedangkan array hanya dapat berisi objek dari satu jenis.

Vektor dapat berkembang pada saat run-time, sementara panjang array tetap.

Metode vektor disinkronkan sedangkan metode Array tidak

Gunakan ArrayList jika Anda menginginkan fungsionalitas yang mirip dengan vektor, tetapi tidak memerlukan sinkronisasi.

- *Pertanyaan: Apa arti keyword "sinkronkan" di Java? Kapan Anda menggunakannya? Apa kerugian dari sinkronisasi*

Jawaban: Sinkronisasi digunakan ketika Anda ingin membuat thread metode Anda aman. Kerugian dari sinkronisasi adalah akan memperlambat program. Juga jika tidak ditangani dengan baik akan berakhir dengan dead lock.

Ada beberapa hal lagi yang perlu diketahui tentang sinkronisasi:

Hanya gunakan (dan minimalkan penggunaannya) sinkronisasi saat menulis kode multithread karena ada biaya yang terkait dengan penggunaannya (hingga lima hingga enam kali lebih lambat, tergantung pada waktu eksekusi metode tersinkronisasi/tidak tersinkronisasi).

Dalam kasus pengubah metode tersinkronisasi, kode byte yang dihasilkan sama persis dengan metode tidak tersinkronisasi. Satu-satunya perbedaan adalah bahwa flag yang disebut flag properti ACC_SYNCRONIZED dalam struktur method_info metode disetel jika pengubah metode tersinkronisasi ada.

Selain itu, keyword tersinkronisasi dapat membuat ukuran kode lebih besar jika digunakan dalam tubuh metode karena bytecode untuk monitorenter/monitorexit dihasilkan sebagai tambahan untuk exception handling apa pun.

Jadi, intinya adalah menggunakan sinkronisasi dengan sangat hati-hati, memahami implikasi kinerja, dan lebih memilih pengubah metode sinkronisasi daripada blok yang disinkronkan.

- *Pertanyaan: Apa itu JDBC? Jelaskan langkah-langkah yang diperlukan untuk menjalankan kueri SQL menggunakan JDBC.*

Jawaban: JDBC adalah API berbasis java untuk mengakses data dari database relasional. JDBC menyediakan sekumpulan kelas dan antarmuka untuk melakukan berbagai operasi database.

Langkah-langkahnya adalah:

- Daftarkan/muat driver jdbc dengan manajer driver.
- Buat koneksi melalui
- DriverManager.getConnection();
- Aktifkan SQL melalui conn.executeStatement()
- Ambil hasil dalam set hasil
- Proses hasilnya
- Tutup set pernyataan/hasil dan objek koneksi.

- *Pertanyaan: Apakah konstruktor diwariskan? Dapatkah subclass memanggil konstruktor classe induk? Kapan?*

Jawaban: Pembangun tidak diwariskan. Pernyataan pertama dalam konstruktor dari sub-kelas harus menjadi panggilan ke konstruktor kelas super.

Ex:

```
class A
{
        int i;
        A(int a)
        {
            i=a;
        }
}
class B extends A
{
        B()
        {
            super(10);//This is must.
        }
}
```

Pernyataan pertama dalam konstruktor dari sub-kelas harus menjadi panggilan ke konstruktor kelas super, hanya dan hanya jika, konstruktor kelas dasar memiliki argumen. Namun jika konstruktor kelas dasar Anda tidak memiliki argumen, maka konstruktor kelas dasar secara otomatis dipanggil dalam sub-kelas tersebut.

- *Pertanyaan: Jelaskan model keamanan java*

Jawaban: Java memiliki model keamanan "kotak pasir", di mana kode yang tidak tepercaya dapat dijauhkan dari data yang seharusnya tidak dapat disentuh, dengan menggunakan analisis operasi individual yang terperinci. Ini berlawanan dengan model keamanan "pengguna tepercaya", yang didorong oleh Microsoft dengan ActiveX, dll. Dalam model ini, penekanannya adalah pada memperoleh rangkaian sertifikat pengguna tepercaya. Setelah sesuatu dipercaya, semuanya telah penuh. akses untuk melakukan semua operasi.

BAB 17
10 PERTANYAAN WAWANCARA CORE JAVA
LANJUTAN UNTUK PROGRAMMER SENIOR DAN BERPENGALAMAN

Pertanyaan wawancara Java untuk programmer Senior dan Berpengalaman

Seperti yang kamu ketahui, Java sangat besar dan tidak ada cara untuk mempersiapkan sepenuhnya untuk wawancara core java apa pun tetapi ada tingkat pertanyaan tergantung pada pengalamanmu, jika kamu lebih segar daripada pertanyaan yang diajukan dalam wawancara Java sebagian besar didasarkan pada fundamental seperti Iterator vs Enumeration di Java, Mengapa main public static dan void atau mungkin ArrayList vs LinkedList di Java.

Hal-hal berubah ketika kamu melamar posisi pengembang senior, Pemimpin Teknis atau Pemimpin Tim Java, pertanyaan yang diajukan pada tingkat itu lebih maju dan kurang populer di kalangan Java, kamu mungkin ditanyai pertanyaan dari pola desain, pertanyaan dari multi-threading, Koleksi dan bahkan diminta untuk menulis kode, mendesain kelas, dan mempersiapkan pengujian JUnit.

Di Bagian Java ini saya akan membagikan beberapa pertanyaan wawancara Core Java tingkat lanjut yang muncul dalam wawancara tingkat Senior sebagian besar pada 4 hingga 6 tahun dan pengalaman 6 hingga 8 tahun. Satu hal penting yang perlu diperhatikan adalah kamu tidak dapat menghapus wawancara hanya dengan menjambret jawaban dari pertanyaan ini karena pewawancara kemungkinan besar akan menanyakan pertanyaan tindak lanjut berdasarkan tanggapanmu dan satu-satunya cara untuk melewatinya adalah dengan memahami topik dengan baik.

Berikut adalah daftar pertanyaan saya yang perlu dipersiapkan jika kamu akan melakukan wawancara core java untuk posisi senior, pertanyaan-pertanyaan ini sebagian besar didasarkan pada Garbage Collection, Konkurensi, Koleksi, desain, Pengkodean, dan pengujian. Bagaimana cara kerja ConcurrentHashMap di Java?

- Pertanyaan: apakah ConcurrentHashMap aman untuk thread?
- Pertanyaan: Bisakah kita mengganti Hashtable ke ConcurrentHashMap tanpa sinkronisasi eksternal?

Seperti yang kamu lihat, ada beberapa pertanyaan dalam kategori ini, yang sebagian besar merupakan tindak lanjut dan dapat dijawab jika kamu memahami ConcurrentHashMap dengan baik.

- Pertanyaan: Apa itu CountDownLatch di Java?
- Pertanyaan: Apa itu CyclicBarrier di Java?
- Pertanyaan: Apa perbedaan antara CountDownLatch dan CyclicBarrier di Java?

Concurrency adalah topik favorit pada wawancara core java lanjutan dan diharapkan dari pengembang Java berpengalaman dan senior untuk memiliki pemahaman yang baik tentang multi-threading dan API konkurensi di Java.

- *Pertanyaan*: Bagaimana kondisi Race? Apakah Anda pernah menghadapi kondisi balapan?

- *Pertanyaan*: Apa itu deadLock di Java? Tulis kode untuk menghindari kebuntuan di Java

Kondisi balapan dan kebuntuan adalah tantangan utama saat menulis aplikasi Java bersamaan berkinerja tinggi dan pengalaman langsung menangani masalah sinkronisasi dan konkurensi yang diharapkan dari programmer Java senior dan berpengalaman. Lihat bagaimana menghindari kebuntuan di java dan Apa itu kondisi balapan di java untuk lebih jelasnya.

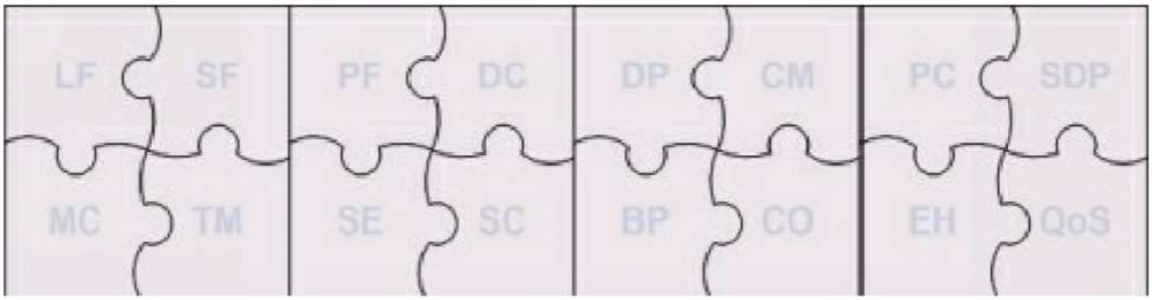
- *Pertanyaan*: Apa itu ruang PermGen?
- *Pertanyaan*: Apa itu kebocoran memori di Java?
- *Pertanyaan*: Apa itu OutOfMemoryError di PermGen mens?
- *Pertanyaan*: Apa perbedaan antara NoClassDefFoundError dan ClassNotFoundException di Java?

Ini adalah beberapa pertanyaan yang terkait dengan Kesalahan dan Pengecualian di Java, *Out Of Memory Error dan No class Def Found Error* adalah kesalahan yang paling umum namun ditakuti di Java.

- *Pertanyaan*: Keterampilan apa yang Anda harapkan sebagai pengembang senior Java?

Keterampilan # 1: Keterampilan "keahlian perangkat lunak" yang bagus. Dengan kata lain, pahami 16 bidang utama teknis yang tercantum di bawah ini. Kamu harus memiliki kemampuan untuk mengajukan pertanyaan yang tepat (misalnya haruskah saya menggunakan pemrograman berorientasi aspek (yaitu AOP) di sini ?, haruskah saya menyukai penguncian yang optimis atau pesimis di sini? Haruskah pemanggilan metode dilakukan dalam konteks transaksional?) Dan menyelesaikan masalah (misalnya kebocoran memori, keamanan thread, serangan injeksi SQL, performa, dll.) yang berkaitan dengan 16 area kunci teknis ini.

Apa 16 bidang utama ini?



- Language Fundamentals (LF)
- Specification Fundamentals (SF)
- Platform Fundamentals (PF)
- Design Considerations (DC)
- Design Patterns (DP)
- Concurrency Management (CM)
- Performance Considerations (PC)
- Memory/Resource Considerations (MC)

- Transaction Management (TM)
- Security (SE)
- Scalability (SC)
- Best Practices (BP)
- Coding (CO)
- Exception Handling (EH)
- Software Development Processes (SDP)
- Quality of Service (QoS)

Bagaimana cara mendapatkan pekerjaan pengembang Java tingkat awal?

Pengembang Java tingkat awal dan pengubah karier terjebak dalam lingkaran setan di mana "Kamu tidak bisa mendapatkan pekerjaan tanpa pengalaman langsung, tetapi pemberi kerja tidak ingin mempekerjakan kamu tanpa pengalaman". Pengusaha mencari pengembang tingkat awal yang dapat mulai berkontribusi sejak mereka bergabung. Ini tidak berarti bahwa pemberi kerja tidak akan memberimu pelatihan dan dukungan, melainkan mencari keterampilan di luar kualifikasi akademis seperti - bekerja secara mandiri, pelajar yang cepat, semangat untuk profesi yang dipilih, pemahaman tentang industri, kemampuan untuk mengkomunikasikan pemikiran kamu/kemampuan dan beberapa "pengalaman langsung" yang sangat dibutuhkan. Tidak ada rumus ajaib lainnya. Kebanyakan majikan tidak peduli dari mana kamu mendapatkan pengalaman itu.

Jadi, jika kamu terjebak dalam lingkaran setan ini, kuncinya adalah mendapatkan pengalaman untuk memutus lingkaran ini melalui proyek otodidak, kontribusi sumber terbuka, dan kerja sukarela di java (misalnya, organisasi amal dan proyek komunitas lokal). Tidak peduli jalan apa yang kamu ambil, dan setiap pengalaman itu penting. Kamu perlu menunjukkan semangat & pemahamanmu tentang profesi yang dipilih, dan kemampuan untuk bekerja secara mandiri dan sebagai tim. Berikut adalah beberapa panduan langkah demi langkah untuk membuat bola bergulir.

Hal-hal yang tidak membuat perbedaan nyata dalam mengamankan pekerjaan:

- Berpikir bahwa menumpuk sertifikasimu dapat membantu.
- Berpikir bahwa kualifikasi pasca sarjana dapat meningkatkan peluangmu.
- Berpikir bahwa membaca buku Java dari depan ke belakang dapat membantu. Sangat penting untuk mempelajari dasar-dasarnya, tetapi gabungkan mempelajari dasar-dasar dengan mendapatkan pengalaman langsung yang sangat dibutuhkan. Bisakah Anda mengendarai mobil hanya dengan mempelajari cara kerja kontrol?

Hal-hal yang harus Anda lakukan untuk membuat terobosan sebagai pengembang Java: Satu-satunya cara untuk belajar mengendarai mobil adalah dengan berada di belakang kemudi. Jadi, singkatnya 3 hal yang harus Anda lakukan.

Investasikan dalam meningkatkan keterampilan berburu pekerjaan Anda yang mencakup keterampilan wawancara, jaringan, dan resume menulis.

Tetap melamar pekerjaan melalui pasar kerja yang diterbitkan & tersembunyi.

Ikuti tip yang diuraikan di bawah ini untuk meningkatkan pengalaman langsung Anda sambil tetap memperhatikan poin 1 & 2.

Semua 3 poin di atas harus berjalan seiring. Itu adalah pekerjaan penuh waktu atau paruh waktu kamu untuk "mencari pekerjaan"

Mendapatkan pengalaman langsung yang sangat dibutuhkan

Langkah # 1:

Java sangat mudah diakses dan semua yang berikut ini tersedia secara gratis. Langkah-langkah yang kamu ambil mungkin sedikit berbeda tergantung pada pemahamanmu dengan Java dan alat-alatnya.

Komputer - atas meja atau putaran atas.

Unduh versi terbaru Java (JDK dan JRE).

Unduh versi terbaru dari IDE gerhana.

Unduh Tomcat atau JBoss untuk menerapkan aplikasimu.

Unduh dan instal database MySQL. Semua aplikasi non-sepele membutuhkan informasi untuk disimpan ke database.

Siapkan Maven sebagai alat manajemen build dan dependensi sehingga kamu dapat mendownload framework yang diinginkan seperti Spring dan Hibernate.

Langkah # 2:

Mulailah dengan dasar-dasarnya terlebih dahulu. Enterprise Java memiliki ratusan kerangka kerja dan pustaka dan mudah bagi pemula untuk bingung. Setelah kamu mencapai titik tertentu, kamu akan memahami mereka dengan lebih baik, tetapi untuk memulai, tetap ikuti langkah-langkah dasar berikut. Jangan ragu untuk membuat perubahan sesuai keinginanmu.

Dasar-dasar Java Inti. Tulis program Java mandiri sederhana menggunakan konsep OO. Tulis pengujian unit dengan JUnit.

Pelajari SQL dan tulis program Java yang berdiri sendiri yang terhubung ke database MySQL melalui JDBC.

Tulis aplikasi web sederhana menggunakan Servelts dan JSP menggunakan Java perusahaan. Aplikasi perlu menyimpan data ke database MySQL. Terapkan aplikasi Anda ke server Tomcat atau JBoss dan jalankan sebagai aplikasi perusahaan. Gunakan Maven untuk membangun dan manajemen ketergantungan.

Perluas proyek Anda yang dibuat dengan JSP, Servlets, dan JDBC untuk menggunakan kerangka kerja yang dicari. Pelajari konsep "injeksi ketergantungan". Mulailah memasang kerangka kerja yang dicari seperti Spring. Mata air sangat luas, dan dimulai dengan inti pegas dan pegas jdbc. Inti pegas digunakan untuk injeksi ketergantungan dan Spring jdbc adalah untuk menyambungkan ke database dan menjalankan kueri SQL.

Pelajari pola desain MVC (Model View Controller) untuk pengembangan web. Ubah JSP dan Layanan Anda menjadi aplikasi web berbasis Spring-mvc.

Tulis layanan web RESTFul menggunakan Spring MVC.

Dapatkan pengetahuan kerja dalam HTML, JavaScript/jQuery/JSON, ajax, dan CSS. Ini sangat penting karena semakin banyak organisasi yang bergerak menuju kerangka kerja MVC berbasis JavaScript seperti angularjs atau backbone. Kerangka kerja ini membuat panggilan layanan web RESTFul untuk mendapatkan data dalam format JSON dan mengisi front end. Akan berguna untuk mempelajari node.js juga jika waktu mengizinkan.

Pastikan kamu menulis pengujian unit untuk kodemu dengan JUnit dan kerangka kerja tiruan seperti Mockito.

Langkah # 3:

Setelah kamu terbiasa dan berpengalaman dalam mengembangkan aplikasi perusahaan dengan Java, cobalah berkontribusi untuk proyek sumber terbuka atau jika proyek otodidakmu tidak sepele, cobalah untuk membuka proyek otodidakmu. Kamu dapat belajar banyak dengan melihat kode orang lain.

Langkah # 4:

Carilah pekerjaan sukarela untuk meningkatkan pengalaman langsungmu. Jangan terlalu memaksakan diri. Alokasikan katakanlah 2 hingga 3 hari untuk membangun situs web untuk amal atau organisasi komunitas.

Langkah # 5:

Bagikan pengalaman langsung kamu yang diperoleh melalui tip 1-4 di resumemu dan melalui blog (awalnya dapat dirahasiakan). Sangat penting untuk menangkap pengalamanmu melalui blogging. Tingkatkan keterampilan menulis resume dan wawancaramu melalui banyak posting praktis yang ditemukan di blog ini atau elseware di internet. Penting bahwa saat kamu mengerjakan tip 1-5, tetaplah melamar pekerjaan berbayar.

Langkah # 6:

Kerja sukarela dan peluang jaringan lainnya melalui Java User Group (JUG) dan pameran dagang lulusan dapat menghubungkanmu dengan para profesional di industri dan membuka lebih banyak pintu untukmu. Langkah 1-5 juga akan membedakan dirimu dari pengembang tingkat awal lainnya. Buku dan blog saya telah membahas banyak pertanyaan dan jawaban wawancara Java.

Latih pertanyaan dan jawaban tersebut karena banyak perusahaan memiliki penyaringan telepon awal dan tes teknis untuk memastikan pengetahuan Java kamu, terutama di core java dan pengembangan web (misalnya protokol HTTP tanpa negara, sesi, cookie, dll.).

Yang diperlukan hanyalah mempelajari 10 Tanya Jawab setiap hari sambil mendapatkan pengalaman langsung dan melamar pekerjaan tingkat pemula. 20 pertanyaan dan jawaban wawancara teknis Java teratas untuk pengembang Java yang berpengalaman.

Menguasai 16 area kunci teknis dan alat praktis untuk meningkatkan produktivitasmu sebagai pengembang Java. Ingin menjadi front-end developer, dan bingung harus mulai dari mana?

Tren baru adalah menggunakan kerangka kerja berbasis JavaScript seperti AngularJS. Jadi, mempelajari JavaScript, HTML (5), dan CSS adalah suatu keharusan. AngularJS berkembang pesat. Standar JEE saat ini adalah JSF. Proyek-proyek yang lebih baru bergerak menuju kerangka kerja MVW (Model View Apapun) sisi klien menggunakan AngularJS, Backbone, ExtJs, dll. Jika Anda akhirnya meningkatkan aplikasi yang sudah ada, Anda memiliki peluang bagus untuk mengerjakan salah satu kerangka kerja Web berikut JSF, Spring MVC , Struts (2), dan GWT.

Menemukan pekerjaan pengembang Java tingkat awal itu sendiri merupakan pekerjaan penuh waktu.

Kamu memiliki 8 jam + pekerjaan yang cocok untuk dirimu. Tinjau kemajuan kamu secara teratur, dan sesuaikan pendekatanmu. Kadang-kadang kamu mungkin memerlukan giliran 360 derajat (misalnya beralih dari menumpuk sertifikasi menjadi mendapatkan pengalaman langsung yang sangat dibutuhkan, beralih dari membaca buku-buku Java dari sampul ke sampul untuk mengambil pekerjaan sukarela, mengubah pencarian pekerjaanmu dari 2 jam sehari menjadi 8 jam sehari, dll) untuk pendekatanmu.

Sangat penting untuk melakukan berbagai hal agar tetap menarik dan membuat kamu tetap termotivasi. Mudah sekali kehilangan motivasi. Bagikan pengalamanmu dengan sesama pengembang Java tingkat awal. Sangat penting bagi kamu untuk terus melamar pekerjaan berbayar sambil mengerjakan tip di atas.

BAB 18
50 TERATAS JAVA MULTI-THREADING
(JAWABAN DAN PERTANYAAN WAWANCARA KURENSI PEMULA, DAN PROGRAMMER BERPENGALAMAN)

Ketika kamu pergi ke wawancara Java, senior atau junior, pengalaman atau pemula, kamu pasti akan melihat beberapa pertanyaan dari utas, konkurensi, dan multi-threading. Faktanya, dukungan konkurensi bawaan ini adalah salah satu poin terkuat dari bahasa pemrograman Java dan membantunya mendapatkan popularitas di antara dunia perusahaan dan pemrogram secara setara. Sebagian besar posisi pengembang Java yang menguntungkan menuntut keterampilan dan pengalaman multi-threading Java inti yang sangat baik dalam mengembangkan, men-debug, dan menyetel aplikasi Java bersamaan dengan latensi rendah yang berkinerja tinggi.

Inilah alasannya, ini adalah salah satu keterampilan yang paling dicari dalam wawancara. Dalam wawancara Java yang khas, Pewawancara perlahan-lahan memulai dari konsep dasar Thread dengan mengajukan pertanyaan seperti, mengapa kamu membutuhkan utas, cara membuat utas, mana cara yang lebih baik untuk membuat utas, mis.

Dengan memperluas kelas thread atau mengimplementasikan Runnable dan kemudian perlahan-lahan masuk ke masalah Concurrency, tantangan yang dihadapi selama pengembangan aplikasi Java bersamaan, model memori Java, utilitas konkurensi tingkat tinggi yang diperkenalkan di JDK 1.5, prinsip dan pola desain aplikasi Java bersamaan, masalah multithreading klasik, mis. konsumen produsen, filsuf makan, penulis pembaca atau masalah penyangga terbatas. Karena tidak cukup hanya mengetahui dasar-dasar threading, kamu harus tahu cara menangani masalah konkurensi, mis. kebuntuan, kondisi balapan, ketidakkonsistenan memori, dan berbagai masalah terkait keamanan thread.

Keterampilan ini diuji secara menyeluruh dengan menghadirkan berbagai masalah multi-threading dan konkurensi. Banyak pengembang Java terbiasa hanya melihat dan membaca pertanyaan wawancara sebelum pergi untuk wawancara, yang tidak buruk tetapi Anda tidak boleh terlalu jauh.

Mengumpulkan pertanyaan dan melakukan latihan yang sama juga memakan banyak waktu, itulah mengapa saya membuat daftar 50 pertanyaan terkait multi-threading dan konkurensi Java, yang dikumpulkan dari berbagai wawancara. Saya hanya akan menambahkan pertanyaan wawancara baru dan terkini ketika saya akan menemukannya.

Ngomong-ngomong, saya belum memberikan jawaban atas pertanyaan ini di sini, Mengapa? karena saya berharap sebagian besar pengembang Java mengetahui jawaban dari pertanyaan ini dan jika tidak, Berikut adalah daftar pertanyaan teratas saya dari thread Java, concurrency dan multi-threading. Anda dapat menggunakan daftar ini untuk mempersiapkan wawancara Java Anda dengan baik.

Apa itu Thread di Java?

Thread adalah jalur eksekusi independen. Ini cara untuk memanfaatkan banyak CPU yang tersedia di mesin. Dengan menggunakan banyak utas, Anda dapat mempercepat tugas terikat CPU. Misalnya, jika satu utas membutuhkan 100 milidetik untuk melakukan pekerjaan, Anda dapat menggunakan 10 utas untuk mengurangi tugas itu menjadi 10 milidetik. Java menyediakan dukungan yang sangat baik untuk multithreading pada tingkat bahasa, dan itu juga salah satu nilai jual yang kuat.

Perbedaan antara Thread dan Proses di Java?

Utas adalah bagian dari Proses, dengan kata lain satu proses dapat berisi banyak utas. Dua proses berjalan pada ruang memori yang berbeda, tetapi semua utas berbagi ruang memori yang sama. Jangan bingung ini dengan memori tumpukan, yang berbeda untuk utas berbeda dan digunakan untuk menyimpan data lokal ke utas itu.

Bagaimana Anda menerapkan Thread di Java?

Di tingkat bahasa, ada dua cara untuk mengimplementasikan Thread di Java. Sebuah instance dari java.lang.Thread mewakili sebuah utas tetapi itu membutuhkan tugas untuk dieksekusi, yang merupakan turunan dari antarmuka java.lang.Runnable. Karena kelas Thread itu sendiri mengimplementasikan Runnable, Anda bisa mengganti metode run() baik dengan memperluas kelas Thread atau hanya mengimplementasikan antarmuka Runnable. Untuk jawaban dan diskusi terperinci, lihat artikel ini.

Kapan menggunakan Runnable vs Thread di Java?

Ini adalah tindak lanjut dari pertanyaan wawancara multi-threading sebelumnya. Seperti yang kita ketahui, kita dapat mengimplementasikan utas baik dengan memperluas kelas Thread atau mengimplementasikan antarmuka Runnable, muncul pertanyaan, mana yang lebih baik dan kapan harus menggunakannya? Pertanyaan ini akan mudah dijawab, jika Anda tahu bahwa bahasa pemrograman Java tidak mendukung multiple inheritance of class, tetapi memungkinkan Anda untuk mengimplementasikan beberapa antarmuka. Artinya, lebih baik mengimplementasikan Runnable daripada memperluas Thread, jika Anda juga ingin memperluas kelas lain, mis. Canvas atau CommandListener.

Perbedaan antara metode start() dan run() dari kelas Thread?

Salah satu pertanyaan jebakan Java dari hari-hari awal, tetapi masih cukup baik untuk membedakan antara pemahaman dangkal model threading Java metode start() digunakan untuk memulai utas yang baru dibuat, sementara start() secara internal memanggil metode run(), ada perbedaan yang memanggil run() secara langsung. Saat Anda memanggil run() sebagai metode normal, yang dipanggil di thread yang sama, tidak ada thread baru yang dimulai, yang merupakan kasus saat Anda memanggil metode start().

Perbedaan antara Runnable dan Callable di Java?

Baik Runnable dan Callable mewakili tugas yang dimaksudkan untuk dijalankan di utas terpisah. Runnable ada di sana dari JDK 1.0, sementara Callable ditambahkan di JDK 1.5. Perbedaan utama antara keduanya adalah bahwa metode Callable's call() bisa mengembalikan nilai dan membuang Exception, yang tidak mungkin dilakukan dengan metode run() Runnable. Callable return objek Future, yang bisa menampung hasil komputasi.

Perbedaan antara CyclicBarrier dan CountDownLatch di Java?

Meskipun CyclicBarrier dan CountDownLatch menunggu jumlah utas pada satu atau beberapa peristiwa, perbedaan utama di antara keduanya adalah Anda tidak dapat menggunakan kembali CountDownLatch setelah hitungan mencapai nol, tetapi Anda dapat menggunakan kembali CyclicBarrier yang sama bahkan setelah penghalang rusak.

Apa itu model Memori Java?

Model Memori Java adalah sekumpulan aturan dan pedoman yang memungkinkan program Java berperilaku secara deterministik di beberapa arsitektur memori, CPU, dan sistem operasi. Ini sangat penting dalam kasus multi-threading. Java Memory Model memberikan jaminan perubahan mana yang dibuat oleh satu thread harus terlihat oleh orang lain, salah satunya adalah hubungan terjadi-sebelum.

Hubungan ini mendefinisikan beberapa aturan yang memungkinkan pemrogram untuk mengantisipasi dan menalar perilaku program Java secara bersamaan. Misalnya, hubungan terjadi-sebelum menjamin:

Setiap tindakan di utas terjadi-sebelum setiap tindakan di utas itu yang muncul kemudian dalam urutan program, ini dikenal sebagai aturan urutan program.

Buka kunci pada kunci monitor terjadi-sebelum setiap kunci berikutnya pada kunci monitor yang sama, juga dikenal sebagai Aturan kunci monitor.

Penulisan ke bidang volatil terjadi sebelum setiap pembacaan berikutnya dari bidang yang sama, yang dikenal sebagai Aturan variabel volatil.

Panggilan ke Thread.start di thread terjadi-sebelum thread lain mendeteksi thread tersebut telah dihentikan, baik dengan berhasil kembali dari Thread.join() atau dengan Thread.isAlive() mengembalikan false, juga dikenal sebagai aturan Thread start.

Sebuah utas memanggil interupsi pada utas lain terjadi-sebelum utas terputus mendeteksi interupsi (baik dengan membuat InterruptedException dilemparkan, atau memanggil isInterrupted atau interrupted), yang dikenal sebagai aturan Interupsi Thread.

Akhir konstruktor untuk objek terjadi-sebelum dimulainya finalizer untuk objek itu, yang dikenal sebagai aturan Finalizer.

Jika A terjadi-sebelum B, dan B terjadi-sebelum C, maka A terjadi-sebelum C, yang berarti terjadi-sebelum menjamin Transitivitas.

Apa variabel volatile di Java?

volatile adalah pengubah khusus, yang hanya dapat digunakan dengan variabel instan. Dalam program Java bersamaan, perubahan yang dibuat oleh beberapa utas pada variabel contoh tidak terlihat oleh orang lain jika tidak ada sinkronisasi apa pun, mis. keyword atau kunci tersinkronisasi. Variabel volatil menjamin bahwa penulisan akan terjadi sebelum pembacaan berikutnya, seperti yang dinyatakan "aturan variabel volatil" dalam pertanyaan sebelumnya.

Apa itu keamanan thread? apakah Vector adalah kelas yang aman untuk thread? (Ya, lihat detailnya)

Keamanan thread adalah properti dari objek atau kode yang menjamin bahwa jika dijalankan atau digunakan oleh banyak utas dengan cara apa pun, mis. baca vs tulis itu akan berperilaku seperti yang diharapkan. Misalnya, objek penghitung yang aman untuk thread tidak akan melewatkan hitungan apa pun jika instance yang sama dari penghitung tersebut dibagikan di antara beberapa utas. Rupanya, Anda juga dapat membagi kelas koleksi dalam dua kategori, aman untuk thread dan tidak aman untuk thread. Vektor memang kelas yang aman untuk thread dan mencapai keamanan thread dengan menyinkronkan metode yang mengubah status Vector, di sisi lain, rekannya ArrayList tidak aman untuk thread.

Bagaimana kondisi ras di java? Diberikan satu contoh?

Kondisi balapan adalah penyebab dari beberapa bug pemrograman halus saat program Java diekspos ke lingkungan eksekusi bersamaan. Seperti namanya, kondisi balapan terjadi karena balapan antara beberapa utas, jika utas yang seharusnya dieksekusi pertama kali kehilangan balapan dan dieksekusi kedua, perilaku kode berubah, yang muncul sebagai bug nondeterministic. Ini adalah salah satu bug tersulit untuk ditemukan dan diproduksi ulang karena sifat acak balapan antar utas. Salah satu contoh kondisi balapan adalah pemrosesan yang tidak sesuai pesanan.

Bagaimana cara menghentikan utas di Java?

Saya selalu mengatakan bahwa Java menyediakan API yang kaya untuk semuanya tetapi ironisnya Java tidak menyediakan cara yang pasti untuk menghentikan utas. Ada beberapa metode kontrol di JDK 1.0 mis. stop(), suspend() dan resume() yang tidak digunakan lagi pada rilis selanjutnya karena potensi ancaman kebuntuan, sejak saat itu perancang Java API tidak melakukan upaya apa pun untuk menyediakan cara yang konsisten, aman untuk thread, dan elegan untuk menghentikan thread. Programmer terutama mengandalkan fakta bahwa thread berhenti secara otomatis segera setelah mereka menyelesaikan eksekusi metode run() atau call(). Untuk menghentikan secara manual, programmer memanfaatkan variabel boolean volatile dan memeriksa setiap iterasi jika metode run memiliki loop atau utas interupsi untuk membatalkan tugas secara tiba-tiba.

Apa yang terjadi ketika pengecualian terjadi di utas?

Ini adalah salah satu pertanyaan bagus tentang Java yang pernah saya lihat dalam wawancara. Dengan kata sederhana, Jika thread tidak tertangkap akan mati, jika pengendali pengecualian yang tidak tertangkap terdaftar maka ia akan mendapat panggilan kembali. Thread.UncaughtExceptionHandler adalah antarmuka, yang didefinisikan sebagai ested interface untuk penangan yang dipanggil saat Thread tiba-tiba berhenti karena pengecualian yang tidak tertangkap. Saat utas akan dihentikan karena pengecualian yang tidak tertangkap, Mesin Virtual Java akan meminta utas untuk UncaughtExceptionHandler menggunakan Thread.getUncaughtExceptionHandler() dan akan memanggil metode uncaughtException() penangan, meneruskan utas dan pengecualian sebagai argumen.

Bagaimana Anda berbagi data antara dua utas di Java?

Anda dapat berbagi data antar utas dengan menggunakan objek bersama, atau struktur data bersamaan seperti BlockingQueue. Lihat tutorial ini untuk mempelajari komunikasi antar thread di Java. Ini mengimplementasikan pola konsumen Produsen menggunakan metode tunggu dan beri tahu, yang melibatkan berbagi objek antara dua utas.

Perbedaan antara notify dan notifyAll di Java?

Ini adalah pertanyaan rumit lainnya dari wawancara core java, karena beberapa utas dapat menunggu pada kunci monitor tunggal, perancang Java API menyediakan metode untuk menginformasikan hanya satu dari mereka atau semuanya, setelah kondisi menunggu berubah, tetapi mereka menyediakan setengah implementasi.

Ada metode notify() tidak menyediakan cara apa pun untuk memilih utas tertentu, itulah mengapa ini hanya berguna ketika Anda tahu bahwa hanya ada satu utas yang menunggu. Di sisi lain, notifyAll() mengirimkan pemberitahuan ke semua utas dan memungkinkan mereka bersaing untuk mendapatkan kunci, yang memastikan bahwa setidaknya satu utas akan melanjutkan lebih jauh.

Mengapa menunggu, beri tahu, dan beri tahu Semua tidak berada di dalam kelas utas?

Ini adalah pertanyaan terkait desain, yang memeriksa apa yang dipikirkan kandidat tentang sistem yang ada atau apakah dia pernah memikirkan sesuatu yang sangat umum tetapi terlihat tidak pantas pada awalnya. Untuk menjawab pertanyaan ini, Anda harus memberikan beberapa alasan mengapa masuk akal untuk ketiga metode ini berada di kelas Object, dan mengapa tidak di kelas Thread.

Salah satu alasan yang jelas adalah bahwa Java menyediakan kunci pada level objek bukan pada level thread. Setiap benda memiliki kunci, yang diperoleh dengan thread. Sekarang jika utas perlu menunggu kunci tertentu, masuk akal untuk memanggil wait() pada objek itu daripada pada utas itu. Jika metode wait() dideklarasikan pada kelas Thread, tidak jelas untuk thread kunci mana yang menunggu.

Singkatnya, karena wait, notify dan notifyAll beroperasi pada level kunci, maka masuk akal untuk mendefinisikannya pada kelas objek karena kunci milik objek.

Apa itu Variabel ThreadLocal di Java?

Variabel ThreadLocal adalah jenis variabel khusus yang tersedia untuk programmer Java. Sama seperti variabel instance adalah per instance, variabel ThreadLocal adalah per thread. Ini adalah cara yang bagus untuk mencapai keamanan thread dari objek yang mahal untuk dibuat, misalnya Anda dapat membuat SimpleDateFormat thread-safe menggunakan ThreadLocal. Karena kelas itu mahal, tidak baik untuk menggunakannya dalam lingkup lokal, yang memerlukan instance terpisah pada setiap pemanggilan.

Dengan menyediakan salinan masing-masing utas, Anda menembak dua burung dalam satu panah. Pertama, Anda mengurangi jumlah instance objek mahal dengan menggunakan kembali jumlah instance tetap, dan Kedua, Anda mencapai keamanan thread tanpa membayar biaya sinkronisasi atau kekekalan. Contoh bagus lainnya dari variabel lokal utas adalah kelas ThreadLocalRandom, yang mengurangi jumlah instance objek acak yang mahal untuk dibuat di lingkungan multi-threading.

Apa FutureTask di Java?

FutureTask mewakili komputasi asinkron yang dapat dibatalkan dalam aplikasi Java bersamaan.

Kelas ini menyediakan implementasi dasar Future, dengan metode untuk memulai dan membatalkan komputasi, kueri untuk melihat apakah komputasi sudah selesai, dan mengambil hasil komputasi. Hasilnya hanya bisa diambil setelah komputasi selesai; metode get akan memblokir jika komputasi belum selesai. Objek FutureTask dapat digunakan untuk menggabungkan objek Callable atau Runnable. Karena FutureTask juga mengimplementasikan Runnable, FutureTask dapat dikirimkan ke Executor untuk dieksekusi.

Perbedaan antara metode interrupted dan isInterrupted di java?

Perbedaan utama antara interrupted() dan isInterrupted() adalah bahwa mantan menghapus status interupsi sementara nanti tidak. Mekanisme interupsi dalam multi-threading Java diimplementasikan menggunakan bendera internal yang dikenal sebagai status interupsi. Menginterupsi utas dengan memanggil Thread.interrupt() menyetel tanda ini. Ketika thread terputus memeriksa interupsi dengan menjalankan metode statis Thread.interrupted(), status interupsi dihapus. Metode isInterrupted() non-statis, yang digunakan oleh satu thread untuk menanyakan status interupsi yang lain, tidak mengubah tanda status interupsi. Menurut konvensi, metode apa pun yang keluar dengan melontarkan InterruptedException akan menghapus status interupsi saat ia melakukannya. Namun, selalu ada kemungkinan bahwa status interupsi akan segera disetel lagi, oleh utas lain yang meminta interupsi

Mengapa metode menunggu dan memberi tahu dipanggil dari blok tersinkronisasi?

Alasan utama untuk memanggil metode tunggu dan beri tahu baik dari blok atau metode tersinkronisasi adalah karena itu dibuat wajib oleh Java API. Jika Anda tidak memanggilnya dari konteks yang disinkronkan, kode Anda akan menampilkan IllegalMonitorStateException. Alasan yang lebih halus adalah untuk menghindari kondisi balapan antara menunggu dan memberi tahu panggilan.

Mengapa Anda harus memeriksa kondisi menunggu dalam satu putaran?

Ada kemungkinan untaian menunggu menerima peringatan palsu dan panggilan bangun palsu, jika tidak memeriksa kondisi menunggu secara berulang, ia akan keluar begitu saja meskipun kondisi tidak terpenuhi. Dengan demikian, saat thread menunggu aktif, ia tidak dapat berasumsi bahwa status yang ditunggunya masih valid. Ini mungkin valid di masa lalu, tetapi statusnya mungkin telah berubah setelah metode notify() dipanggil dan sebelum thread yang menunggu diaktifkan. Itulah mengapa selalu lebih baik memanggil metode wait() dari loop, Anda bahkan dapat membuat template untuk memanggil wait dan notify di Eclipse.

Perbedaan antara koleksi tersinkronisasi dan serentak di Java?

Meskipun koleksi yang disinkronkan dan bersamaan menyediakan koleksi yang aman untuk thread yang sesuai untuk akses multi-thread dan akses bersamaan, nantinya lebih skalabel daripada sebelumnya. Sebelum Java 1.5, programmer Java hanya memiliki koleksi tersinkronisasi yang menjadi sumber pertikaian jika beberapa thread mengaksesnya secara bersamaan, yang menghambat skalabilitas sistem. Java 5 memperkenalkan koleksi bersamaan seperti ConcurrentHashMap, yang tidak hanya menyediakan keamanan thread tetapi juga meningkatkan skalabilitas dengan menggunakan teknik modern seperti lock stripping dan partisi tabel internal.

Perbedaan antara Stack dan Heap di Java?

Mengapa seseorang mempertanyakan ini sebagai bagian dari multi-threading dan concurrency? karena Stack adalah area memori yang terkait erat dengan utas. Untuk menjawab pertanyaan ini, stack dan heap adalah memori khusus dalam aplikasi Java. Setiap utas memiliki tumpukannya sendiri, yang digunakan untuk menyimpan variabel lokal, parameter metode, dan tumpukan panggilan. Variabel yang disimpan dalam satu tumpukan Thread tidak terlihat oleh yang lain. Di sisi lain, heap adalah area memori umum yang digunakan bersama oleh semua utas. Objek baik lokal atau di tingkat mana pun dibuat di dalam heap. Untuk meningkatkan performa, thread cenderung menyimpan nilai cache dari heap ke tumpukannya, yang dapat menimbulkan masalah jika variabel tersebut dimodifikasi oleh lebih dari satu thread, di sinilah muncul variabel volatil. volatile menyarankan utas untuk membaca nilai variabel selalu dari memori utama.

Apa itu kumpulan thread? Mengapa Anda harus melakukan thread pool di Java?

Membuat utas mahal dalam hal waktu dan sumber daya. Jika Anda membuat utas pada saat pemrosesan permintaan, ini akan memperlambat waktu respons Anda, juga hanya ada sejumlah utas terbatas yang dapat dibuat oleh proses. Untuk menghindari kedua masalah ini, kumpulan utas dibuat saat aplikasi dimulai dan utas digunakan kembali untuk pemrosesan permintaan. Kumpulan utas ini dikenal sebagai "kumpulan utas" dan utas dikenal sebagai utas pekerja. Dari rilis JDK 1.5, Java API menyediakan kerangka kerja Pelaksana, yang memungkinkan Anda membuat berbagai jenis kumpulan utas, mis. kumpulan utas tunggal, yang memproses satu tugas dalam satu waktu, kumpulan utas tetap (kumpulan utas dengan jumlah tetap) atau kumpulan utas cache (kumpulan utas yang dapat diperluas cocok untuk aplikasi dengan banyak tugas berumur pendek).

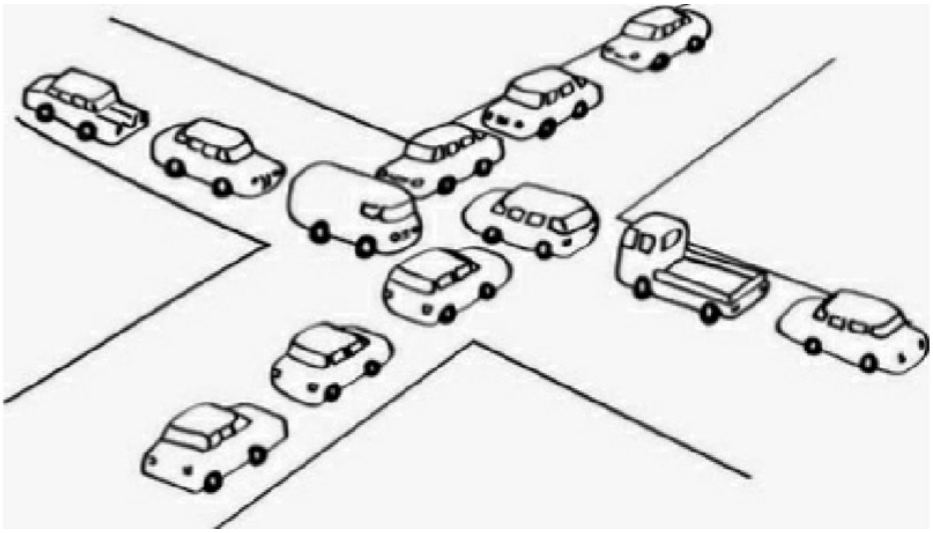
Tulis kode untuk memecahkan masalah Konsumen Produsen di Java?

Sebagian besar masalah threading yang Anda selesaikan di dunia nyata adalah dari kategori pola konsumen Produsen, di mana satu utas menghasilkan tugas dan utas lainnya memakannya. Anda harus tahu bagaimana melakukan komunikasi antar utas untuk mengatasi masalah ini. Pada level terendah, Anda dapat menggunakan menunggu dan memberi tahu untuk menyelesaikan masalah ini, dan pada level tinggi Anda dapat memanfaatkan Semaphore atau BlockingQueue untuk mengimplementasikan pola konsumen Producer.

Bagaimana Anda memeriksa apakah Thread memegang kunci atau tidak?

Saya bahkan tidak tahu bahwa Anda dapat memeriksa apakah sebuah Thread telah terkunci sebelum pertanyaan ini menghantam saya dalam wawancara telepon dengan Java. Ada metode yang disebut holdLock() di java.lang.Thread, ini mengembalikan nilai true jika dan hanya jika thread saat ini memegang kunci monitor pada objek yang ditentukan.

Bagaimana Anda menghindari kebuntuan di Java? Tulis Kode?



Gambar 19.1 trafic dipersimpangan

Deadlock adalah kondisi di mana dua utas menunggu satu sama lain untuk mengambil tindakan yang memungkinkan mereka untuk bergerak lebih jauh. Ini masalah serius karena ketika itu terjadi program Anda macet dan tidak melakukan tugas yang seharusnya. Agar kebuntuan terjadi, empat syarat berikut harus benar:

Pengecualian Bersama: Setidaknya satu sumber daya harus disimpan dalam mode tidak dapat dibagikan. Hanya satu proses yang dapat menggunakan sumber daya pada saat tertentu.

Tahan dan Tunggu: Sebuah proses saat ini memegang setidaknya satu sumber daya dan meminta sumber daya tambahan yang ditahan oleh proses lain.

Tanpa Pre-emption: Sistem operasi tidak boleh mengalokasikan sumber daya setelah dialokasikan; mereka harus dibebaskan melalui proses penahanan secara sukarela.

Circular Wait: Suatu proses harus menunggu sumber daya yang ditahan oleh proses lain, yang pada gilirannya menunggu proses pertama untuk melepaskan sumber daya.

Cara termudah untuk menghindari kebuntuan adalah dengan mencegah menunggu melingkar, dan ini dapat dilakukan dengan memperoleh kunci dalam urutan tertentu dan melepaskannya dalam urutan terbalik, sehingga utas hanya dapat melanjutkan untuk mendapatkan kunci jika memegang yang lain.

Perbedaan antara Livelock dan Deadlock di java?

Pertanyaan ini merupakan perpanjangan dari pertanyaan wawancara sebelumnya. Penguncian langsung mirip dengan kebuntuan, kecuali bahwa status utas atau proses yang terlibat dalam penguncian langsung terus berubah terkait satu sama lain, tanpa ada yang maju lebih jauh.

Livelock adalah kasus khusus dari kelaparan sumber daya. Contoh dunia nyata dari penguncian langsung terjadi ketika dua orang bertemu di koridor sempit, dan masing-masing mencoba bersikap sopan dengan bergerak ke samping untuk membiarkan yang lain lewat, tetapi mereka akhirnya bergoyang dari sisi ke sisi tanpa membuat kemajuan apa pun karena mereka berdua berulang kali bergerak. dengan cara yang sama pada waktu yang sama. Singkatnya, perbedaan utama antara penguncian langsung dan kebuntuan adalah bahwa dalam keadaan proses sebelumnya berubah tetapi tidak ada kemajuan yang dibuat.

Bagaimana Anda mengambil thread dump di Java?

Ada beberapa cara untuk mengambil thread dump dari proses Java tergantung pada sistem operasi. Saat Anda mengambil thread dump, status dump JVM dari semua thread di file log atau konsol kesalahan standar. Di windows Anda dapat menggunakan kombinasi tombol Ctrl + Break untuk mengambil thread dump, di Linux Anda dapat menggunakan perintah kill -3 untuk hal yang sama. Anda juga dapat menggunakan alat

yang disebut jstack untuk mengambil thread dump, ini beroperasi pada id proses, yang dapat ditemukan menggunakan alat lain yang disebut jps.

Parameter JVM mana yang digunakan untuk mengontrol ukuran tumpukan utas? Ini adalah yang sederhana, parameter -Xss digunakan untuk mengontrol ukuran tumpukan Thread di Java.

Perbedaan antara Synchronized dan ReentrantLock di Java?

Ada hari-hari ketika satu-satunya cara untuk memberikan pengecualian timbal balik di Java adalah melalui keyword tersinkronisasi, tetapi memiliki beberapa kekurangan, mis. Anda tidak dapat memperpanjang kunci di luar metode atau batasan blok, Anda tidak dapat menyerah untuk mencoba kunci, dll. Java 5 memecahkan masalah ini dengan menyediakan kontrol yang lebih canggih melalui antarmuka Kunci.

ReentrantLock adalah implementasi umum dari antarmuka Lock dan menyediakan Kunci pengecualian timbal balik yang masuk kembali dengan perilaku dasar dan semantik yang sama seperti kunci monitor implisit yang diakses menggunakan metode dan pernyataan tersinkronisasi, tetapi dengan kemampuan yang diperluas.

Ada tiga utas T1, T2 dan T3? Bagaimana Anda memastikan urutan T1, T2, T3 di java?

Pengurutan dalam multi-threading dapat dilakukan dengan berbagai cara, tetapi Anda cukup menggunakan metode join() dari kelas utas untuk memulai utas ketika utas lain selesai eksekusinya. Untuk memastikan tiga utas dieksekusi, Anda harus memulai yang terakhir dulu, mis. T3 dan kemudian panggil metode gabung dalam urutan terbalik mis. T3 memanggil T2. gabung, dan T2 memanggil T1. gabung, dengan cara ini T1 akan selesai lebih dulu dan T3 akan selesai terakhir.

Apa yang dilakukan metode hasil kelas Thread?

Metode hasil adalah salah satu cara untuk meminta thread saat ini untuk melepaskan CPU agar thread lain mendapat kesempatan untuk mengeksekusi. Yield adalah metode statis dan hanya menjamin bahwa utas saat ini akan melepaskan CPU tetapi tidak mengatakan apa pun tentang utas mana yang akan mendapatkan CPU. Ini mungkin untuk utas yang sama untuk mendapatkan kembali CPU dan memulai eksekusinya lagi.

Apa tingkat konkurensi ConcurrentHashMap di Java?

ConcurrentHashMap mencapai skalabilitas dan keamanan utasnya dengan mempartisi peta aktual menjadi beberapa bagian. Partisi ini dicapai dengan menggunakan level konkurensi. Ini parameter opsional dari konstruktor ConcurrentHashMap dan nilai defaultnya adalah 16. Tabel secara internal dipartisi untuk mencoba mengizinkan jumlah yang ditunjukkan dari pembaruan bersamaan tanpa perselisihan.

Apa itu Semaphore di Java?

Semaphore di Java adalah jenis sinkronisasi baru. Ini semafor penghitungan. Secara konseptual, semaphore memiliki satu set izin. Setiap memperoleh() blok jika perlu sampai izin tersedia, dan kemudian mengambilnya. Setiap rilis() menambahkan izin, berpotensi melepaskan pengakuisisi yang memblokir. Namun, sebenarnya tidak ada objek izin yang digunakan; Semaphore hanya menyimpan hitungan jumlah yang tersedia dan bertindak sesuai dengan itu.

Semaphore digunakan untuk melindungi sumber daya mahal yang tersedia dalam jumlah tetap mis. koneksi database di pool. pelajari lebih lanjut tentang menghitung Semafor di java.

Apa yang terjadi jika Anda mengirimkan tugas, ketika antrian kumpulan utas sudah terisi?

Ini adalah pertanyaan rumit lainnya dalam daftar saya. Banyak programmer akan berpikir bahwa itu akan memblokir sampai tugas diselesaikan tetapi itu benar. Metode submit() ThreadPoolExecutor akan menampilkan RejectedExecutionException jika tugas tidak dapat dijadwalkan untuk dieksekusi.

Perbedaan antara kumpulan thread metode submit() dan execute() di Java?

Kedua metode tersebut adalah cara untuk mengirimkan tugas ke kumpulan utas tetapi ada sedikit perbedaan di antara keduanya. mengeksekusi (perintah Runnable) didefinisikan di antarmuka Pelaksana dan mengeksekusi tugas yang diberikan di masa depan, tetapi yang lebih penting itu tidak mengembalikan apa pun. Jenis pengembaliannya tidak berlaku. Di sisi lain submit() adalah metode kelebihan beban, itu bisa mengambil tugas Runnable atau Callable dan bisa mengembalikan objek Future yang bisa menahan hasil komputasi yang tertunda. Metode ini didefinisikan pada antarmuka ExecutorService, yang memperluas antarmuka Executor, dan setiap kelas kumpulan utas lainnya, mis. ThreadPoolExecutor atau ScheduledThreadPoolExecutor mendapatkan metode ini.

Apa metode blocking di Java?

Metode blocking adalah metode yang memblokir hingga tugas selesai, misalnya metode accept() dari blok ServerSocket hingga klien terhubung. di sini pemblokiran berarti kontrol tidak akan kembali ke pemanggil sampai tugas selesai. Di sisi lain ada metode asynchronous atau non-blocking yang kembali bahkan sebelum tugas selesai.

Apakah swing thread aman? Apa yang Anda maksud dengan Swing thread aman?

Anda dapat dengan mudah menjawab pertanyaan ini karena Tidak, Swing tidak thread-safe, tetapi Anda harus menjelaskan apa yang Anda maksud dengan pertanyaan tersebut meskipun pewawancara tidak menanyakannya. Saat kami mengatakan swing tidak aman untuk thread, kami biasanya merujuk komponennya, yang tidak dapat dimodifikasi dalam beberapa thread. Semua pembaruan untuk

komponen GUI harus dilakukan pada utas AWT, dan Swing menyediakan metode panggilan balik sinkron dan asinkron untuk menjadwalkan pembaruan tersebut. Anda juga dapat membaca artikel saya untuk mempelajari lebih lanjut tentang ayunan dan keamanan thread untuk menjawab pertanyaan ini dengan lebih baik. Bahkan dua pertanyaan berikutnya juga terkait dengan konsep ini.

Perbedaan antara invokeAndWait dan invokeLater di Java?

Ini adalah dua metode Swing API yang menyediakan pengembang Java untuk memperbarui komponen GUI dari utas selain utas operator acara. InvokeAndWait() secara sinkron memperbarui komponen GUI, misalnya bilah kemajuan, setelah kemajuan dibuat, bilah juga harus diperbarui untuk mencerminkan perubahan itu.

Jika kemajuan dilacak di utas yang berbeda, itu harus memanggil invokeAndWait() untuk menjadwalkan pembaruan komponen itu oleh utas operator Peristiwa. Di sisi lain, invokeLater() adalah panggilan asinkron untuk memperbarui komponen.

Metode Swing API mana yang aman untuk thread di Java?

Pertanyaan ini lagi-lagi terkait dengan ayunan dan keamanan ulir, meskipun komponen tidak aman untuk ulir, ada metode tertentu yang dapat dipanggil dengan aman dari beberapa utas. Saya tahu tentang repaint(), dan revalidate() yang aman untuk thread tetapi ada metode lain pada komponen ayunan yang berbeda, mis. setText() metode JTextComponent, insert() dan append() dari JTextAreaclass.

Apa itu ReadWriteLock di Java?

Secara umum read write lock merupakan hasil dari teknik lock stripping untuk meningkatkan performansi aplikasi secara bersamaan. Di Java, ReadWriteLock adalah antarmuka yang ditambahkan pada rilis Java 5. Sebuah ReadWriteLock memelihara sepasang kunci terkait, satu untuk operasi hanya-baca dan satu lagi untuk menulis. Kunci baca dapat dipegang secara bersamaan oleh beberapa utas pembaca, selama tidak ada penulis. Kunci tulis bersifat eksklusif. Jika mau, Anda dapat mengimplementasikan antarmuka ini dengan kumpulan aturan Anda sendiri, jika tidak, Anda dapat menggunakan ReentrantReadWriteLock, yang disertakan dengan JDK dan mendukung maksimal 65535 kunci tulis rekursif dan 65535 kunci baca.

Apa itu busy spin dalam multi-threading?

Busy spin adalah teknik yang digunakan pemrogram secara bersamaan untuk membuat utas menunggu pada kondisi tertentu. Tidak seperti metode tradisional mis. wait(), sleep() atau yield() yang semuanya melibatkan pelepasan kontrol CPU, metode ini tidak melepaskan CPU, melainkan hanya menjalankan loop kosong. Mengapa seseorang melakukan itu? untuk mempertahankan cache CPU. Dalam sistem multi inti, mungkin saja utas yang dijeda melanjutkan pada inti yang berbeda, yang berarti membangun kembali cache lagi. Untuk menghindari biaya membangun kembali cache,

programmer lebih suka menunggu waktu yang jauh lebih singkat untuk melakukan putaran sibuk.

Perbedaan antara variabel volatil dan atom di Java?

Ini adalah pertanyaan yang menarik untuk programmer Java, pada awalnya variabel volatile dan atomic terlihat sangat mirip, tetapi mereka berbeda. Variabel volatile memberi Anda jaminan bahwa sebelum penulisan akan terjadi sebelum penulisan berikutnya, itu tidak menjamin atomisitas. Misalnya operasi count++ tidak akan menjadi atom hanya dengan mendeklarasikan variabel count sebagai volatile. Di sisi lain, kelas AtomicInteger menyediakan metode atom untuk melakukan operasi gabungan seperti itu secara atomik, mis. getAndIncrement() adalah atom pengganti dari operator increment. Ini dapat digunakan untuk menambah nilai arus secara atomik satu per satu. Demikian pula Anda memiliki versi atom untuk tipe data lain dan variabel referensi juga.

Apa yang terjadi jika utas melempar Pengecualian ke dalam blok tersinkronisasi?

Ini adalah satu lagi pertanyaan rumit untuk programmer Java rata-rata, jika dia dapat membawa fakta tentang apakah kunci dilepaskan atau tidak adalah indikator kunci pemahamannya. Untuk menjawab pertanyaan ini, tidak peduli bagaimana Anda ada blok tersinkronisasi, baik biasanya dengan menyelesaikan eksekusi atau tiba-tiba dengan melemparkan pengecualian, thread melepaskan kunci yang diperolehnya saat memasuki blok tersinkronisasi itu. Ini sebenarnya salah satu alasan saya suka blok tersinkronisasi melalui antarmuka kunci, yang memerlukan perhatian eksplisit untuk melepaskan kunci, umumnya ini dicapai dengan melepaskan kunci di blok akhirnya.

Apa itu penguncian ganda pada Singleton?

Ini adalah salah satu pertanyaan yang sangat populer di wawancara Java, dan terlepas dari popularitasnya, peluang kandidat untuk menjawab pertanyaan ini dengan memuaskan hanya 50%. Separuh dari waktu, mereka gagal menulis kode untuk penguncian yang diperiksa dua kali dan separuh dari waktu mereka gagal bagaimana cara rusak dan diperbaiki pada Java 1.5. Ini sebenarnya adalah cara lama untuk membuat singleton yang aman untuk thread, yang mencoba mengoptimalkan kinerja hanya dengan mengunci saat instance Singleton dibuat pertama kali, tetapi karena kerumitan dan fakta bahwa ia rusak untuk JDK 1.4, saya pribadi tidak menyukainya . Bagaimanapun, bahkan jika Anda tidak menyukai pendekatan ini, ada baiknya untuk mengetahui dari sudut pandang wawancara.

Bagaimana cara membuat Singleton yang aman untuk thread di Java?

Pertanyaan ini sebenarnya merupakan tindak lanjut dari pertanyaan sebelumnya. Jika Anda mengatakan Anda tidak suka penguncian yang dicentang ganda, maka Pewawancara pasti akan bertanya tentang cara-cara alternatif untuk membuat kelas Singleton yang aman untuk thread. Sebenarnya ada man, Anda dapat

memanfaatkan pemuatan kelas dan fitur inisialisasi variabel statis JVM untuk membuat instance Singleton, atau Anda dapat memanfaatkan jenis enumerasi yang kuat di Java untuk membuat Singleton.

Sebutkan 3 praktik terbaik multi-threading yang Anda ikuti?

Ini adalah pertanyaan favorit saya, karena saya yakin Anda harus mengikuti praktik terbaik tertentu saat menulis kode bersamaan yang membantu dalam performa, debugging, dan pemeliharaan. Berikut adalah tiga praktik terbaik, menurut saya rata-rata programmer Java harus mengikuti:

Selalu berikan nama yang bermakna ke utas Anda. Hal ini sangat membantu dalam menemukan bug atau melacak eksekusi dalam kode bersamaan. OrderProcessor, QuoteProcessor atau TradeProcessor jauh lebih baik daripada Thread-1. Thread-2 dan Thread-3. Nama harus mengatakan tentang tugas yang dilakukan oleh utas itu. Semua kerangka kerja utama dan bahkan JDK mengikuti praktik terbaik ini.

Hindari penguncian atau Kurangi cakupan Penguncian Sinkronisasi itu mahal dan context-switching bahkan lebih mahal. Cobalah untuk menghindari sinkronisasi dan penguncian sebanyak mungkin dan minimal, Anda harus mengurangi bagian kritis.

Itu sebabnya saya lebih suka blok tersinkronisasi daripada metode tersinkronisasi, karena ini memberi Anda kendali mutlak pada ruang lingkup penguncian.

Lebih suka Sinkronisasi daripada menunggu dan beri tahu Sinkronisasi seperti CountDownLatch, Semaphore, CyclicBarrier, atau Exchanger menyederhanakan pengkodean. Sangat sulit untuk menerapkan stream kontrol yang kompleks dengan benar menggunakan tunggu dan beri tahu. Kedua, kelas-kelas ini ditulis dan dikelola oleh yang terbaik dalam bisnis dan ada kemungkinan besar bahwa mereka dioptimalkan atau diganti dengan kode kinerja yang lebih baik dalam rilis JDK berikutnya. Dengan menggunakan utilitas sinkronisasi tingkat yang lebih tinggi, Anda secara otomatis mendapatkan semua manfaat ini.

Lebih Memilih Koleksi Bersamaan daripada Koleksi Tersinkronisasi Ini adalah praktik terbaik sederhana lainnya yang mudah diikuti tetapi menuai manfaat yang baik. Koleksi serentak lebih skalabel daripada rekan tersinkronnya, itulah mengapa lebih baik menggunakannya saat menulis kode serentak. Jadi lain kali jika Anda membutuhkan peta, pikirkan tentang ConcurrentHashMap sebelum memikirkan Hashtable.

Bagaimana Anda memaksa memulai Thread di Java?

Pertanyaan ini seperti bagaimana Anda memaksa Garbage Collection di Java, tidak mungkin, meskipun Anda dapat membuat permintaan menggunakan System.gc() tetapi tidak dijamin. Pada multi-threading Java, mereka tidak memiliki cara untuk memaksa memulai utas, ini dikontrol oleh penjadwal utas dan Java tidak mengekspos API untuk mengontrol jadwal utas. Ini masih sedikit acak di java.

Apa itu fork join framework di Java?

Kerangka kerja gabungan garpu, yang diperkenalkan di JDK 7 adalah alat canggih yang tersedia bagi pengembang Java untuk memanfaatkan beberapa prosesor server modern. Ini dirancang untuk pekerjaan yang dapat dipecah menjadi potongan-potongan kecil secara rekursif.

Sasarannya adalah menggunakan semua daya pemrosesan yang tersedia untuk meningkatkan kinerja aplikasi Anda. Satu keuntungan signifikan dari kerangka kerja fork/join adalah ia menggunakan algoritme pencuri kerja. Utas pekerja yang kehabisan hal yang harus dilakukan dapat mencuri tugas dari utas lain yang masih sibuk.

Apa perbedaan antara memanggil metode wait() dan sleep() di Java multi-threading?

Meskipun wait dan Sleep memperkenalkan beberapa bentuk jeda dalam aplikasi Java, keduanya adalah alat untuk kebutuhan yang berbeda.

Metode wait digunakan untuk komunikasi antar utas, ini melepaskan kunci jika kondisi menunggu benar dan menunggu pemberitahuan ketika karena tindakan kondisi menunggu utas lain menjadi salah. Di sisi lain, metode sleep() hanya untuk melepaskan CPU atau menghentikan eksekusi thread saat ini selama durasi waktu yang ditentukan. Memanggil metode tidur tidak melepaskan kunci yang dipegang oleh utas saat ini.

BAB 19
JAVA MULTI-THREADING DAN KURENSI
- PERTANYAAN WAWANCARA DENGA JAWABAN -

Multithreading atau Concurrency adalah salah satu topik populer dalam pertanyaan wawancara java. Di sini saya mencantumkan sebagian besar pertanyaan penting dari perspektif wawancara, tetapi Anda harus memiliki pengetahuan yang baik tentang utas java untuk menangani pertanyaan tindak lanjut.

Apa perbedaan antara Process dan Thread?

Proses adalah lingkungan eksekusi mandiri dan dapat dilihat sebagai program atau aplikasi sedangkan Thread adalah tugas tunggal eksekusi dalam proses. Lingkungan runtime Java berjalan sebagai proses tunggal yang berisi kelas dan program yang berbeda sebagai proses. Thread bisa disebut proses ringan. Thread membutuhkan lebih sedikit sumber daya untuk dibuat dan ada dalam proses, thread berbagi sumber daya proses.

Apa keuntungan dari pemrograman multi-threaded?

Dalam pemrograman Multi-Thread, beberapa thread dijalankan secara bersamaan yang meningkatkan performa karena CPU tidak menganggur jika beberapa thread sedang menunggu untuk mendapatkan beberapa resource. Beberapa utas berbagi memori heap, jadi sebaiknya buat beberapa utas untuk menjalankan beberapa tugas daripada membuat banyak proses. Misalnya, Servlet memiliki kinerja yang lebih baik daripada CGI karena Servlet mendukung multi-threading tetapi CGI tidak.

Apa perbedaan antara User Thread dan Daemon Thread?

Saat kami membuat Thread di program java, hal itu disebut sebagai utas pengguna. Untaian daemon berjalan di latar belakang dan tidak mencegah JVM untuk dihentikan. Ketika tidak ada utas pengguna yang berjalan, JVM mematikan program dan keluar. Utas anak yang dibuat dari daemon thread juga merupakan daemon thread.

Bagaimana cara membuat Thread di Java?

Ada dua cara untuk membuat Thread di Java - pertama dengan mengimplementasikan antarmuka Runnable dan kemudian membuat objek Thread darinya dan kedua adalah memperluas Kelas Thread.

Apa saja status berbeda dalam siklus hidup Thread?

Saat kita membuat Thread di program java, statusnya adalah New. Kemudian kami memulai utas yang mengubah statusnya menjadi Dapat Dijalankan. Thread Scheduler bertanggung jawab untuk mengalokasikan CPU ke thread di kumpulan

thread Runnable dan mengubah statusnya menjadi Running. Status Thread lainnya adalah Menunggu, Diblokir, dan Mati.

Bisakah kita memanggil metode run() dari kelas Thread?

Ya, kita dapat memanggil metode run() dari kelas Thread tetapi kemudian akan berperilaku seperti metode normal. Untuk benar-benar mengeksekusinya di Thread, kita perlu memulainya menggunakan metode Thread.start().

Bagaimana kita bisa menjeda eksekusi Thread untuk waktu tertentu?

Kita bisa menggunakan metode thread class sleep() untuk menjeda eksekusi Thread untuk waktu tertentu. Perhatikan bahwa ini tidak akan menghentikan pemrosesan utas untuk waktu tertentu, setelah utas bangun dari tidur, statusnya diubah menjadi dapat dijalankan dan berdasarkan penjadwalan utas, itu akan dieksekusi.

Apa yang Anda pahami tentang Prioritas Utas?

Setiap utas memiliki prioritas, biasanya utas dengan prioritas lebih tinggi mendapat prioritas dalam eksekusi tetapi itu tergantung pada implementasi Penjadwal Utas yang bergantung pada OS. Kita dapat menentukan prioritas utas, tetapi itu tidak menjamin bahwa utas dengan prioritas lebih tinggi akan dijalankan sebelum utas dengan prioritas lebih rendah. Prioritas utas adalah INT yang nilainya bervariasi dari 1 hingga 10 di mana 1 adalah utas prioritas terendah dan 10 adalah utas prioritas tertinggi.

Apa itu Thread Scheduler dan Time Slicing?

Thread Scheduler adalah layanan Sistem Operasi yang mengalokasikan waktu CPU ke thread yang dapat dijalankan yang tersedia. Setelah kami membuat dan memulai utas, pelaksanaannya bergantung pada penerapan Penjadwal Utas. Mengiris Waktu adalah proses untuk membagi waktu CPU yang tersedia ke utas yang dapat dijalankan yang tersedia. Alokasi waktu CPU ke thread dapat didasarkan pada prioritas thread atau thread yang menunggu lebih lama akan mendapatkan prioritas lebih dalam mendapatkan waktu CPU. Penjadwalan untaian tidak dapat dikontrol oleh java, jadi selalu lebih baik untuk mengontrolnya dari aplikasi itu sendiri.

Apa itu context-switching dalam multi-threading?

Context-switching adalah proses menyimpan dan memulihkan status CPU sehingga eksekusi Thread dapat dilanjutkan dari titik yang sama di lain waktu. Context-switching adalah fitur penting untuk sistem operasi multitasking dan dukungan untuk lingkungan multi-utas.

Bagaimana kita bisa memastikan main() adalah utas terakhir yang diselesaikan dalam Program Java?

Kita bisa menggunakan metode Thread join() untuk memastikan semua thread yang dibuat oleh program sudah mati sebelum menyelesaikan fungsi utama. Berikut adalah artikel tentang metode Thread join.

Bagaimana utas berkomunikasi satu sama lain?

Saat utas berbagi sumber daya, komunikasi antar Rangkaian penting untuk mengoordinasikan upaya mereka. Metode kelas objek wait(), notify() dan notifyAll() memungkinkan utas berkomunikasi tentang status kunci sumber daya.

Mengapa metode komunikasi thread wait(), notify() dan notifyAll() ada di kelas Object?

Di Java, setiap Object memiliki monitor dan menunggu, metode notify digunakan untuk menunggu monitor Object atau untuk memberi tahu thread lain bahwa Object monitor sudah bebas sekarang. Tidak ada monitor pada utas di java dan sinkronisasi dapat digunakan dengan Objek apa pun, itulah mengapa ini merupakan bagian dari kelas Objek sehingga setiap kelas di java memiliki metode penting ini untuk komunikasi antar utas.

Mengapa metode wait(), notify() dan notifyAll() harus dipanggil dari metode atau blok tersinkronisasi?

Ketika Thread memanggil wait() pada Objek apa pun, ia harus memiliki monitor pada Objek yang akan ditinggalkannya dan berada dalam status menunggu hingga thread lain memanggil notify() pada Objek ini. Demikian pula ketika sebuah thread memanggil notify() pada Object apapun, ia meninggalkan monitor pada Object dan thread menunggu lainnya bisa mendapatkan monitor pada Object. Karena semua metode ini memerlukan Thread untuk memiliki monitor Object, yang hanya dapat dicapai dengan sinkronisasi, mereka perlu dipanggil dari metode atau blok yang disinkronkan.

Mengapa metode Thread sleep() dan yield() bersifat statis?

Metode thread sleep() dan yield() bekerja pada thread yang sedang dijalankan. Jadi tidak ada gunanya menerapkan metode ini di beberapa utas lain yang berada dalam status menunggu. Itulah mengapa metode ini dibuat statis sehingga ketika metode ini dipanggil secara statis, metode ini bekerja pada thread yang sedang dieksekusi dan menghindari kebingungan bagi pemrogram yang mungkin berpikir bahwa mereka dapat memanggil metode ini pada beberapa thread yang tidak berjalan.

Mengapa metode wait(), notify() dan notifyAll() harus dipanggil dari metode atau blok tersinkronisasi?

Ketika Thread memanggil wait() pada Objek apa pun, ia harus memiliki monitor pada Objek yang akan ditinggalkannya dan berada dalam status menunggu hingga thread lain memanggil notify() pada Objek ini. Demikian pula ketika sebuah thread memanggil notify() pada Object apapun, ia meninggalkan monitor pada Object dan thread menunggu lainnya bisa mendapatkan monitor pada Object. Karena semua metode ini memerlukan Thread untuk memiliki monitor Object, yang hanya dapat dicapai dengan sinkronisasi, mereka perlu dipanggil dari metode atau blok yang disinkronkan.

Bagaimana kami dapat mencapai keamanan thread di Java?

Ada beberapa cara untuk mencapai keamanan thread di java - sinkronisasi, kelas atomic bersamaan, mengimplementasikan antarmuka Lock bersamaan, menggunakan keyword volatile, menggunakan kelas yang tidak dapat diubah dan kelas aman Thread. Pelajari lebih lanjut di tutorial keamanan thread.

Apa itu keyword volatile di Java Saat kami menggunakan keyword volatile dengan variabel, semua utas membaca nilainya langsung dari memori dan tidak menyimpannya dalam cache. Ini memastikan bahwa nilai yang dibaca sama dengan yang ada di memori.

Mana yang lebih disukai - Metode tersinkronisasi atau blok tersinkronisasi?

Blok tersinkronisasi lebih disukai karena tidak mengunci Objek, metode tersinkronisasi mengunci Objek dan jika ada beberapa blok sinkronisasi di kelas, meskipun tidak terkait, itu akan menghentikannya dari eksekusi dan menempatkannya dalam status menunggu untuk mengunci Object.

Bagaimana cara membuat daemon thread di Java?

Kelas thread setDaemon (true) dapat digunakan untuk membuat thread daemon di java. Kita perlu memanggil metode ini sebelum memanggil metode start(), jika tidak metode ini akan memunculkan IllegalThreadStateException.

Apa itu ThreadLocal?

Java ThreadLocal digunakan untuk membuat variabel thread-lokal. Kita tahu bahwa semua utas Objek berbagi variabelnya, jadi jika variabel tersebut tidak aman untuk utas, kita dapat menggunakan sinkronisasi tetapi jika kita ingin menghindari sinkronisasi, kita dapat menggunakan variabel ThreadLocal.

Setiap thread memiliki variabel Thread Local sendiri dan mereka dapat menggunakan metode get() dan set() untuk mendapatkan nilai default atau mengubah nilai lokal menjadi Thread.

Instance ThreadLocal biasanya adalah bidang statis pribadi di kelas yang ingin mengaitkan status dengan utas.

Apa itu Thread Group? Mengapa disarankan untuk tidak menggunakannya?

ThreadGroup adalah kelas yang dimaksudkan untuk memberikan informasi tentang grup utas. ThreadGroup API lemah dan tidak memiliki fungsi apa pun yang tidak disediakan oleh Thread. Dua dari fitur utama yang dimilikinya adalah mendapatkan daftar utas aktif dalam grup utas dan mengatur pengendali pengecualian yang tidak tertangkap untuk utas. Tetapi Java 1.5 telah menambahkan metode SETUNCAUGHTEXCEPTIONHANDLER (UNCAUGHTEXCEPTIONHANDLER EH) yang dengannya kita dapat menambahkan pengendali pengecualian yang tidak tertangkap ke utas. Jadi ThreadGroup sudah usang dan karenanya tidak disarankan untuk digunakan lagi.

```
o t1.setUncaughtExceptionHandler(new UncaughtExceptionHandler(){
o @Override
o Kehampaan publik uncaughtException(Thread t, Throwable e) {
o System.out.println("terjadi pengecualian:"+e.getMessage());          }
o });
```

Apa itu Java Thread Dump, Bagaimana cara mendapatkan Java Thread dump dari suatu Program?

Thread dump adalah daftar semua thread yang aktif di JVM, thread dump sangat membantu dalam menganalisis kemacetan dalam aplikasi dan menganalisis situasi kebuntuan. Ada banyak cara untuk menghasilkan Thread dump - Menggunakan Profiler, perintah Kill -3, alat jstack, dll. Saya lebih suka alat jstack untuk menghasilkan thread dump dari suatu program karena mudah digunakan dan dilengkapi dengan instalasi JDK. Karena ini adalah alat berbasis terminal, kita dapat membuat skrip untuk menghasilkan thread dump secara berkala untuk dianalisis nanti.

Apa itu Deadlock? Bagaimana menganalisis dan menghindari situasi kebuntuan?

Deadlock adalah situasi pemrograman di mana dua atau lebih utas diblokir selamanya, situasi ini muncul dengan setidaknya dua utas dan dua atau lebih sumber daya.

Untuk menganalisis kebuntuan, kita perlu melihat java thread dump aplikasi, kita perlu mencari thread dengan status DIBLOKIR dan kemudian sumber daya yang menunggu untuk dikunci, setiap sumber daya memiliki ID unik yang dapat kita temukan. utas mana yang sudah menahan kunci pada objek. Hindari Nested Locks, Lock Only What is Required dan Hindari menunggu tanpa batas waktu adalah cara umum untuk menghindari situasi kebuntuan.

Apa itu Kelas Pengatur Waktu Java? Bagaimana cara menjadwalkan tugas untuk dijalankan setelah interval tertentu?

java.util.Timer adalah kelas utilitas yang dapat digunakan untuk menjadwalkan utas agar dijalankan pada waktu tertentu di masa mendatang. Kelas Java Timer dapat digunakan untuk menjadwalkan tugas agar dijalankan satu kali atau dijalankan secara

berkala. java.util.TimerTask adalah kelas abstrak yang mengimplementasikan antarmuka Runnable dan kita perlu memperluas kelas ini untuk membuat TimerTask kita sendiri yang dapat dijadwalkan menggunakan kelas Java Timer.

Apa itu Thread Pool? Bagaimana kita bisa membuat Thread Pool di Java?

Kumpulan utas mengelola kumpulan utas pekerja, ini berisi antrian yang membuat tugas menunggu untuk dieksekusi. Kumpulan utas mengelola koleksi utas Runnable dan utas pekerja mengeksekusi Runnable dari antrean.

java.util.concurrent.Executors menyediakan implementasi antarmuka java.util.concurrent.Executor untuk membuat kumpulan thread di java. Program Contoh Thread Pool menunjukkan cara membuat dan menggunakan Thread Pool di java. Atau baca Contoh ScheduledThreadPoolExecutor untuk mengetahui cara menjadwalkan tugas setelah penundaan tertentu.

BAB 20
JAWABAN PERTANYAAN WAWANCARA KURENSI JAVA

Apa itu operasi atom? Apa itu kelas atom di Java Concurrency API?

Operasi atom dilakukan dalam satu unit tugas tanpa gangguan dari operasi lain. Operasi atom diperlukan dalam lingkungan multi-utas untuk menghindari ketidakkonsistenan data. int++ bukanlah operasi atom. Jadi pada saat satu utas membaca nilainya dan menambahnya satu per satu, utas lain telah membaca nilai yang lebih lama yang mengarah ke hasil yang salah. Untuk mengatasi masalah ini, kita harus memastikan bahwa operasi kenaikan pada hitungan adalah atom, kita dapat melakukannya menggunakan Sinkronisasi tetapi Java 5 java.util.concurrent.atomic menyediakan wrapperclasses untuk int dan long yang dapat digunakan untuk mencapai ini secara atomik tanpa menggunakan Sinkronisasi. Buka artikel ini untuk mempelajari lebih lanjut tentang kelas serentak atom.

Apa itu Lock interface di Java Concurrency API? Apa manfaatnya dibandingkan sinkronisasi?

Antarmuka kunci menyediakan operasi penguncian yang lebih ekstensif daripada yang dapat diperoleh dengan menggunakan metode dan pernyataan tersinkronisasi. Mereka memungkinkan penataan yang lebih fleksibel, mungkin memiliki properti yang sangat berbeda, dan mungkin mendukung beberapa objek Kondisi terkait.

Anda dapat mencoba mendapatkan kunci, tetapi segera kembali atau setelah batas waktu jika kunci tidak dapat diperoleh

Anda dapat memperoleh dan melepaskan kunci dalam cakupan yang berbeda, dan dalam urutan yang berbeda

Apa itu Kerangka Pelaksana?

Di Java 5, kerangka kerja Pelaksana diperkenalkan dengan antarmuka java.util.concurrent.Executor. Kerangka Pelaksana adalah kerangka kerja untuk standarisasi permintaan, penjadwalan, eksekusi, dan kontrol tugas asinkron sesuai dengan serangkaian kebijakan eksekusi.

Membuat banyak utas tanpa batas ke ambang batas maksimum dapat menyebabkan aplikasi kehabisan memori heap. Jadi, membuat ThreadPool adalah solusi yang lebih baik karena jumlah utas yang terbatas dapat dikumpulkan dan digunakan kembali. Kerangka pelaksana memfasilitasi proses pembuatan kumpulan Thread di java.

Apa itu BlockingQueue? Bagaimana kita bisa mengimplementasikan masalah Produsen-Konsumen menggunakan Blocking Queue?

java.util.concurrent.BlockingQueue adalah Antrean yang mendukung operasi yang menunggu antrean menjadi tidak kosong saat mengambil dan menghapus elemen, dan menunggu hingga ruang tersedia di antrean saat menambahkan elemen. BlockingQueue tidak menerima nilai null dan menampilkan NullPointerException jika Anda mencoba menyimpan nilai null dalam antrean.

Implementasi BlockingQueue aman untuk thread. Semua metode antrian bersifat atomic dan menggunakan kunci internal atau bentuk kontrol konkurensi lainnya. Antarmuka BlockingQueue adalah bagian dari kerangka kerja java collection dan terutama digunakan untuk mengimplementasikan masalah konsumen produsen.

Apa itu Callable dan Future?

Java 5 memperkenalkan antarmuka java.util.concurrent.Callable dalam paket konkurensi yang mirip dengan antarmuka Runnable tetapi dapat mengembalikan Objek apa pun dan dapat menampilkan Exception. Antarmuka yang dapat dipanggil menggunakan Generik untuk menentukan jenis objek yang dikembalikan. Executor class menyediakan metode yang berguna untuk mengeksekusi Callable di kumpulan thread. Karena tugas yang dapat dipanggil berjalan secara paralel, kita harus menunggu Object yang dikembalikan. Tugas yang dapat dipanggil mengembalikan objek java.util.concurrent.Future. Menggunakan Future kita bisa mengetahui status dari tugas yang Dapat Dipanggil dan mendapatkan Object yang dikembalikan. Ini menyediakan metode get() yang bisa menunggu Callable selesai dan kemudian mengembalikan hasilnya.

Apa itu FutureTask Class?

FutureTask adalah kelas implementasi dasar antarmuka Future dan kita dapat menggunakannya dengan Executors untuk pemrosesan asinkron.

Seringkali kita tidak perlu menggunakan kelas FutureTask tetapi ini sangat berguna jika kita ingin mengganti beberapa metode antarmuka Future dan ingin mempertahankan sebagian besar implementasi dasar. Kami hanya dapat memperluas kelas ini dan mengganti metode sesuai dengan kebutuhan kami.

Apa itu Concurrent Collection Classes?

Kelas Java Collection bersifat fail-fast yang berarti jika Collection akan diubah saat beberapa utas melewatinya menggunakan iterator, iterator.next() akan menampilkan

ConcurrentModificationException. Kelas Koleksi Serentak mendukung serentak pengambilan penuh dan konkurensi yang diharapkan dapat disesuaikan untuk pembaruan. Kelas utama adalah ConcurrentHashMap, CopyOnWriteArrayList dan CopyOnWriteArraySet,

Apa itu Executor class?

Kelas Executors menyediakan metode utilitas untuk kelas Executor, ExecutorService, ScheduledExecutorService, ThreadFactory, dan Callable. Executor class dapat digunakan untuk membuat Thread Pool dengan mudah di java, juga ini adalah satu-satunya kelas yang mendukung eksekusi implementasi Callable.

Apa saja peningkatan dalam Concurrency API di Java 8?

Beberapa penyempurnaan API bersamaan yang penting adalah:

- ConcurrentHashMap compute(), forEach(), forEachEntry(), forEachKey(), forEachValue(), merge(), reduce() dan search().
- CompletableFuture yang mungkin diselesaikan secara eksplisit (menyetel nilai dan statusnya).

Eksekutor metode newWorkStealingPool() untuk membuat kumpulan thread yang mencuri pekerjaan menggunakan semua prosesor yang tersedia sebagai level paralelisme targetnya.

Koleksi 40 Pertanyaan Dan Jawaban Wawancara Java

Kerangka Java Collections adalah aspek fundamental dari bahasa pemrograman java. Ini adalah salah satu topik penting untuk pertanyaan wawancara java. Di sini saya mencantumkan beberapa pertanyaan dan jawaban penting untuk kerangka Java Collections.

Apa fitur yang berhubungan dengan Koleksi di Java 8?

Java 8 telah membawa perubahan besar dalam Collection API. Beberapa perubahan tersebut adalah:

Java Stream API untuk kelas koleksi untuk mendukung pemrosesan sekuensial dan paralel

Interface Iterable diperluas dengan metode default forEach() yang bisa kita gunakan untuk mengulang koleksi. Ini sangat membantu ketika digunakan dengan Lambda expression karena itu argumen Konsumen adalah fungsi Interface

Perbaikan API Koleksi Lain-lain seperti metode forEachRemaining (Tindakan konsumen) antarmuka inIterator, metode Map replaceAll(), compute(), merge().

Apa itu Java Collections Framework? Sebutkan beberapa manfaat framework Koleksi?

Koleksi digunakan dalam setiap bahasa pemrograman dan rilis java awal berisi beberapa kelas untuk koleksi: Vector, Stack, Hashtable, Array. Tetapi melihat ruang lingkup dan penggunaan yang lebih besar, Java 1.2 hadir dengan Collections Framework yang mengelompokkan semua antarmuka, implementasi, dan algoritme koleksi.

Java collection telah berkembang pesat dengan penggunaan kelas Generik dan Koleksi Bersamaan untuk operasi yang aman untuk thread.

Ini juga termasuk memblokir antarmuka dan implementasinya dalam paket konkuren java. Beberapa manfaat dari kerangka koleksi adalah;

- Mengurangi upaya pengembangan dengan menggunakan kelas koleksi inti daripada menerapkan kelas koleksi kita sendiri.
- Kualitas kode ditingkatkan dengan penggunaan kelas kerangka koleksi yang teruji dengan baik.
- Mengurangi upaya pemeliharaan kode dengan menggunakan kelas koleksi yang dikirimkan dengan JDK.

Dapat digunakan kembali dan Interoperabilitas

Apa keuntungan dari Generics in Collections Framework?

Java 1.5 hadir dengan Generik dan semua antarmuka dan implementasi koleksi sangat menggunakannya. Generik memungkinkan kita untuk menyediakan tipe Objek yang dapat berisi koleksi, jadi jika Anda mencoba menambahkan elemen apa pun dari tipe lain, kesalahan waktu kompilasi akan muncul. Ini menghindari ClassCastException saat Runtime karena Anda akan mendapatkan error saat kompilasi. Juga Generik membuat kode bersih karena kita tidak perlu menggunakan operator transmisi dan INSTANCEOF.

Apa interface dasar dari Java Collections Framework?

Koleksi adalah akar dari hierarki koleksi. Koleksi mewakili sekelompok objek yang dikenal sebagai elemennya. Platform Java tidak menyediakan implementasi langsung apa pun dari antarmuka ini.

Set adalah koleksi yang tidak bisa berisi elemen duplikat. Antarmuka ini memodelkan abstraksi himpunan matematika dan digunakan untuk mewakili himpunan, seperti setumpuk kartu.

List/Daftar adalah koleksi yang dipesan dan dapat berisi elemen duplikat. Anda dapat mengakses elemen apa pun dari indeksnya. List lebih menyerupai array dengan panjang dinamis.

Map/Peta adalah objek yang memetakan kunci ke nilai. Peta tidak dapat berisi kunci duplikat: Setiap kunci dapat memetakan paling banyak ke satu nilai.

Beberapa interface lainnya adalah Queue, Dequeue, Iterator, SortedSet, SortedMap dan ListIterator.

Mengapa Collection tidak memperluas antarmuka Cloneable dan Serializable?

Antarmuka koleksi menentukan grup Objek yang dikenal sebagai elemen. Bagaimana elemen dipertahankan terserah pada implementasi konkret Collection. Misalnya, beberapa implementasi Koleksi seperti List mengizinkan elemen duplikat sedangkan implementasi lain seperti Set tidak.

Banyak implementasi Collection memiliki metode klon publik. Namun, tidak masuk akal untuk menyertakannya dalam semua penerapan Pengumpulan. Ini karena Collection adalah representasi abstrak. Yang penting adalah implementasinya.

Semantik dan implikasi dari kloning atau serialisasi ikut bermain ketika berhadapan dengan implementasi yang sebenarnya; jadi implementasi konkret harus memutuskan bagaimana itu harus dikloning atau diserialisasi, atau bahkan jika itu bisa dikloning atau serial. Jadi, mewajibkan kloning dan serialisasi di semua implementasi sebenarnya kurang fleksibel dan lebih membatasi. Implementasi spesifik harus membuat keputusan apakah bisa di kloning atau serial.

Mengapa antarmuka Peta tidak memperluas antarmuka Collections?

Meskipun antarmuka Peta dan implementasinya adalah bagian dari Kerangka Kerja collection, Peta bukanlah koleksi dan koleksi bukan Peta. Oleh karena itu, tidak masuk akal jika Peta memperluas Koleksi atau sebaliknya. Jika Map memperluas antarmuka Koleksi, lalu di manakah elemen-elemennya? Peta berisi pasangan nilai kunci dan menyediakan metode untuk mengambil daftar Kunci atau nilai sebagai Koleksi tetapi tidak sesuai dengan paradigma "kelompok elemen".

Apa itu Iterator?

Antarmuka Iterator menyediakan metode untuk mengulang Koleksi apa pun. Kita bisa mendapatkan instance iterator dari Collection menggunakan metode ITERATOR(). Iterator menggantikan Enumeration di Java Collections Framework. Iterator memungkinkan pemanggil untuk menghapus elemen dari koleksi yang mendasari selama iterasi. Java Collection iterator menyediakan cara umum untuk traversal melalui elemen koleksi dan mengimplementasikan Pola Desain Iterator.

Apa perbedaan antara antarmuka Enumerasi dan Iterator?

Pencacahan dua kali lebih cepat dari Iterator dan menggunakan lebih sedikit memori. Pencacahan sangat mendasar dan sesuai dengan kebutuhan dasar. Tapi Iterator jauh lebih aman dibandingkan dengan Enumeration karena ia selalu menolak utas lain untuk memodifikasi objek koleksi yang sedang diiterasi olehnya.

Iterator menggantikan Enumeration di Java Collections Framework. Iterator memungkinkan pemanggil untuk menghapus elemen dari koleksi yang mendasari yang tidak dimungkinkan dengan Pencacahan. Nama metode Iterator telah ditingkatkan untuk memperjelas fungsinya.

Mengapa tidak ada metode seperti Iterator.add() untuk menambahkan elemen ke Collections?

Semantiknya tidak jelas, mengingat kontrak untuk Iterator tidak memberikan jaminan tentang urutan iterasi. Namun, perhatikan bahwa ListIterator menyediakan operasi penambahan, karena ini menjamin urutan iterasi.

Mengapa Iterator tidak memiliki metode untuk mendapatkan elemen berikutnya secara langsung tanpa memindahkan kursor?

Ini dapat diterapkan di atas antarmuka Iterator saat ini, tetapi karena jarang digunakan, tidak masuk akal untuk memasukkannya ke dalam antarmuka yang harus diterapkan setiap orang.

Apa perbedaan antara Iterator dan ListIterator?

Kita dapat menggunakan Iterator untuk melintasi kumpulan Set dan List sedangkan ListIterator hanya dapat digunakan dengan List.

Iterator dapat melintasi ke arah depan hanya sedangkan ListIterator dapat digunakan untuk melintasi kedua arah.

ListIterator mewarisi dari antarmuka Iterator dan dilengkapi dengan fungsi tambahan seperti menambahkan elemen, mengganti elemen, mendapatkan posisi indeks untuk elemen sebelumnya dan berikutnya.

Apa sajakah cara berbeda untuk mengulang daftar?

Kita dapat mengulang daftar dengan dua cara berbeda - menggunakan iterator dan menggunakan for-each loop.

```
List&lt;String&gt; strList = new ArrayList&lt;&gt;();
//using for-each loop
for(String obj : strList){
System.out.println(obj);
}
//using iterator
Iterator&lt;String&gt; it = strList.iterator();
while(it.hasNext()){
 String obj = it.next();
System.out.println(obj);
}
```

Menggunakan iterator lebih aman untuk thread karena memastikan bahwa jika elemen daftar yang mendasarinya diubah, itu akan memunculkan ConcurrentModificationException.

Apa yang Anda pahami dengan properti fail-fast iterator?

Properti gagal-cepat Iterator memeriksa setiap modifikasi dalam struktur koleksi yang mendasarinya setiap kali kita mencoba mendapatkan elemen berikutnya. Jika ada modifikasi yang ditemukan, itu throwsConcurrentModificationException. Semua implementasi Iterator di kelas Collection gagal-cepat menurut desain kecuali kelas koleksi serentak seperti ConcurrentHashMap dan CopyOnWriteArrayList.

Apa perbedaan antara fail-fast dan fail-safe?

Properti aman-gagal Iterator berfungsi dengan klon dari koleksi yang mendasarinya, oleh karena itu tidak terpengaruh oleh modifikasi apa pun dalam koleksi. Secara desain, semua kelas koleksi dalam paket java.util gagal-cepat sedangkan kelas koleksi di java.util.concurrent adalah gagal aman. Iterator gagal-cepat menampilkan ConcurrentModificationException sedangkan iterator gagal-aman tidak pernah menampilkan ConcurrentModificationException.

Bagaimana cara menghindari ConcurrentModificationException saat mengulang koleksi?

Kita bisa menggunakan kelas koleksi serentak untuk menghindari ConcurrentModificationException saat mengulang koleksi, misalnya CopyOnWriteArrayList, bukan ArrayList.

Mengapa tidak ada implementasi konkret dari antarmuka Iterator?

Antarmuka Iterator mendeklarasikan metode untuk mengulang koleksi tetapi implementasinya adalah tanggung jawab kelas implementasi Collection. Setiap kelas koleksi yang mengembalikan iterator untuk traverse memiliki kelas bertingkat implementasi Iteratornya sendiri.

Ini memungkinkan kelas koleksi untuk memilih apakah iterator gagal-cepat atau tidak aman. Misalnya iterator ArrayList gagal-cepat sedangkan iterator CopyOnWriteArrayList gagal-aman.

Apa itu UnsupportedOperationException?

UnsupportedOperationException adalah pengecualian yang digunakan untuk menunjukkan bahwa operasi tidak didukung.

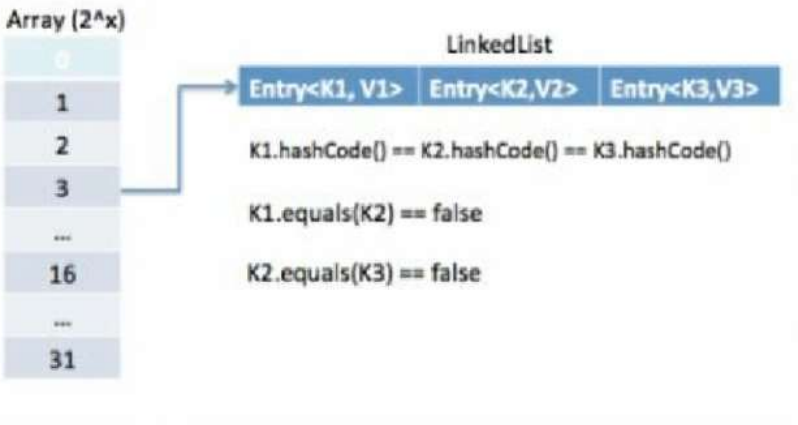
Ini digunakan secara ekstensif di kelas JDK, dalam framework java collection.util.Collections.UnmodifiableCollectionmenolak pengecualian ini untuk semua operasi tambah dan hapus.

Bagaimana HashMap bekerja di Java?

HashMap menyimpan pasangan nilai kunci di Map.Entry implementasi kelas bertingkat statis. HashMap bekerja pada algoritma hashing dan menggunakan metode hashCode() dan equals() dalam metode put dan get. Ketika kita memanggil metode put dengan melewatkan pasangan nilai-kunci, HashMap menggunakan Key hashCode() dengan hashing untuk mengetahui indeks untuk menyimpan pasangan kunci-nilai. Entri disimpan di LinkedList, jadi jika sudah ada entri yang ada, ini menggunakan metode equals() untuk memeriksa apakah kunci yang diteruskan sudah ada, jika ya itu menimpa nilai lain itu membuat entri baru dan menyimpan Entri nilai kunci ini .

Ketika kita memanggil metode get dengan melewatkan Key, sekali lagi ia menggunakan hashCode() untuk menemukan indeks dalam larik dan kemudian menggunakan metode equals() untuk menemukan Entri yang benar dan

mengembalikan nilainya. Gambar di bawah ini akan menjelaskan detail tersebut dengan jelas.



Hal penting lainnya yang perlu diketahui tentang HashMap adalah kapasitas, faktor beban, pengubahan ukuran ambang batas. Kapasitas default awal HashMap adalah 32 dan faktor beban adalah 0,75. Ambang batas adalah kapasitas yang dikalikan dengan faktor beban dan setiap kali kami mencoba menambahkan entri, jika ukuran peta lebih besar dari ambang batas, HashMap mengulangi konten peta menjadi larik baru dengan kapasitas yang lebih besar. Kapasitas selalu berkekuatan 2, jadi jika Anda tahu bahwa Anda perlu menyimpan banyak key-value pair, misalnya dalam caching data dari database, ada baiknya untuk menginisialisasi HashMap dengan kapasitas dan load factor yang benar.

Apa pentingnya metode hashCode() dan equals()?

HashMap menggunakan metode Key object hashCode() dan equals() untuk menentukan indeks untuk menempatkan pasangan kunci-nilai. Metode ini juga digunakan saat kami mencoba mendapatkan nilai dari HashMap. Jika metode ini tidak diterapkan dengan benar, dua Kunci yang berbeda mungkin menghasilkan keluaran hashCode() dan sama dengan() yang sama dan dalam hal ini daripada menyimpannya di lokasi yang berbeda, HashMap akan menganggapnya sama dan menimpanya.

Demikian pula, semua kelas koleksi yang tidak menyimpan data duplikat menggunakan hashCode() dan equals() untuk menemukan duplikat, jadi sangat penting untuk menerapkannya dengan benar. Implementasi equals() dan hashCode() harus mengikuti aturan ini.

Jika o1.equals (o2), maka o1.hashCode() == o2.hashCode() harus selalu benar.

Jika o1.hashCode() == o2.hashCode benar, bukan berarti o1.equals (o2) benar.

Bisakah kita menggunakan kelas apa pun sebagai kunci Peta?

Kita dapat menggunakan kelas apa pun sebagai Kunci Peta, namun poin-poin berikut harus dipertimbangkan sebelum menggunakannya.

Jika kelas mengganti metode equals(), itu juga harus mengganti metode hashCode().

Kelas harus mengikuti aturan yang terkait dengan equals() dan hashCode() untuk semua instance. Silakan lihat pertanyaan sebelumnya untuk aturan ini.

Jika bidang kelas tidak digunakan dalam equals(), Anda tidak boleh menggunakannya dalam metode hashCode().

Praktik terbaik untuk kelas kunci yang ditentukan pengguna adalah membuatnya tidak dapat diubah, sehingga nilai hashCode() dapat disimpan dalam cache untuk kinerja yang cepat. Juga kelas yang tidak dapat diubah memastikan bahwa hashCode() dan sama dengan() tidak akan berubah di masa depan yang akan menyelesaikan masalah apa pun dengan mutabilitas.

Misalnya, saya memiliki kelas MyKey yang saya gunakan untuk kunci HashMap.

```
//MyKeyword nama Argumen yang diteruskan digunakan untuk yang sama
MyKey key = new MyKey("Pankaj"); //assume hashCode=1234
myHashMap.put(key, "Value");
// Below code will change the key hashCode() and equals()
// but it's location is not changed. key.setName("Amit");
//Asumsu kode baru hashCode=7890
//di bawah ini akan mengembalikan nol, karena HashMap akan mencoba mencari kunci
//dalam indeks yang sama seperti yang disimpan tetapi karena kunci dimutasi,
//tidak akan ada kecocokan dan akan mengembalikan null.
myHashMap.get(new MyKey("Pankaj"));
```

Inilah alasan mengapa String dan Integer banyak digunakan sebagai kunci HashMap.

Apa perbedaan tampilan Koleksi yang disediakan oleh antarmuka Peta?

Interface Map/Antarmuka peta menyediakan tiga tampilan koleksi:

Set keySet(): Mengembalikan tampilan Set dari kunci yang terdapat dalam peta ini. Himpunan ini didukung oleh peta, sehingga perubahan pada peta tercermin dalam himpunan, dan sebaliknya.

Jika peta dimodifikasi saat iterasi atas set sedang berlangsung (kecuali tidak ditentukan. Set ini mendukung penghapusan elemen, yang menghapus pemetaan terkait dari peta, melalui operasi Iterator.remove, Set.remove, removeAll, retAll, dan clear . Itu tidak mendukung operasi add atau addAll.

Nilai koleksi(): Mengembalikan tampilan Koleksi dari nilai-nilai yang terdapat dalam peta ini. Koleksinya didukung oleh peta, sehingga perubahan pada peta tercermin dalam koleksi, dan sebaliknya.

Jika peta dimodifikasi saat iterasi atas koleksi sedang berlangsung (kecuali melalui operasi penghapusan iterator sendiri), hasil dari iterasi tidak ditentukan. Koleksi ini mendukung penghapusan elemen, yang menghapus pemetaan terkait dari peta, melalui operasi Iterator.remove, Collection.remove, removeAll, dipertahankanAll, dan hapus. Itu tidak mendukung operasi add atau addAll.

Setel <Map.Entry <K, V >> entrySet(): Mengembalikan tampilan Set dari pemetaan yang terdapat dalam peta ini. Himpunan ini didukung oleh peta, sehingga perubahan pada peta tercermin dalam himpunan, dan sebaliknya. Jika peta diubah saat iterasi atas himpunan sedang berlangsung (kecuali melalui operasi hapus iterator

sendiri, atau melalui operasi setValue pada entri peta yang dikembalikan oleh iterator) hasil dari iterasi tidak ditentukan.

Set ini mendukung penghapusan elemen, yang menghapus pemetaan terkait dari peta, melalui operasi Iterator.remove, Set.remove, removeAll, dipertahankanAll, dan hapus. Itu tidak mendukung operasi add atau addAll.

Apa perbedaan antara HashMap dan Hashtable?

HashMap dan Hashtable keduanya mengimplementasikan antarmuka Peta dan terlihat serupa, namun ada perbedaan berikut antara HashMap dan Hashtable.

HashMap mengizinkan kunci dan nilai nol sedangkan Hashtable tidak mengizinkan kunci dan nilai nol.

Hashtable disinkronkan tetapi HashMap tidak disinkronkan. Jadi HashMap lebih baik untuk lingkungan utas tunggal, Hashtable cocok untuk lingkungan multi-utas.

LinkedHashMap diperkenalkan di Java 1.4 sebagai subkelas dari HashMap, jadi jika Anda menginginkan urutan iterasi, Anda dapat dengan mudah beralih dari

HashMap ke LinkedHashMap tetapi tidak demikian halnya dengan Hashtable yang urutan iterasinya tidak dapat diprediksi. HashMap menyediakan Set kunci untuk mengulang dan karenanya cepat gagal tetapi Hashtable menyediakan Enumerasi kunci yang tidak mendukung fitur ini.

Hashtable dianggap sebagai kelas inheritance dan jika Anda mencari modifikasi Map saat melakukan iterasi, Anda harus menggunakan ConcurrentHashMap.

Bagaimana cara memutuskan antara HashMap dan TreeMap?

Untuk memasukkan, menghapus, dan menemukan elemen di Peta, HashMap menawarkan alternatif terbaik. Namun, jika Anda perlu melintasi kunci dalam urutan yang diurutkan, maka TreeMap adalah alternatif yang lebih baik.

Bergantung pada ukuran koleksi Anda, mungkin lebih cepat menambahkan elemen ke HashMap, lalu mengonversi peta menjadi TreeMap untuk traversal kunci yang diurutkan.

Apa persamaan dan perbedaan antara ArrayList dan Vector?

ArrayList dan Vector adalah kelas yang serupa dalam banyak hal.

Keduanya berbasis indeks dan didukung oleh array secara internal.

Keduanya mempertahankan urutan penyisipan dan kita bisa mendapatkan elemen dalam urutan penyisipan.

Implementasi iterator ArrayList dan Vector keduanya dirancang dengan cepat.

ArrayList dan Vector keduanya memungkinkan nilai null dan akses acak ke elemen menggunakan nomor indeks.

Inilah perbedaan antara ArrayList dan Vector.

Vektor disinkronkan sedangkan ArrayList tidak disinkronkan. Namun jika Anda mencari modifikasi daftar saat melakukan iterasi, Anda harus menggunakan CopyOnWriteArrayList.

ArrayList lebih cepat daripada Vector karena tidak memiliki overhead karena sinkronisasi.

ArrayList lebih serbaguna karena kita bisa mendapatkan daftar tersinkronisasi atau daftar hanya-baca darinya dengan mudah menggunakan kelas utilitas Koleksi.

Apa perbedaan antara Array dan ArrayList? Kapan Anda akan menggunakan Array di atas ArrayList?

Array dapat berisi objek primitif atau sedangkan ArrayList hanya dapat berisi Objek. Array adalah ukuran tetap sedangkan ukuran ArrayList dinamis. Array tidak menyediakan banyak fitur seperti ArrayList, seperti addAll, removeAll, iterator, dll. Meskipun ArrayList adalah pilihan yang jelas saat kita mengerjakan daftar, ada beberapa kali array bagus untuk digunakan.

Jika ukuran daftar tetap dan sebagian besar digunakan untuk menyimpan dan melintasi mereka. Untuk daftar tipe data primitif, meskipun Koleksi menggunakan autoboxing untuk mengurangi upaya pengkodean tetapi tetap membuatnya lambat saat bekerja pada tipe data primitif ukuran tetap. Jika Anda bekerja pada situasi multi-dimensi tetap, menggunakan [] [] jauh lebih mudah daripada List <List <>>

Apa perbedaan antara ArrayList dan LinkedList?

ArrayList dan LinkedList keduanya mengimplementasikan antarmuka Daftar tetapi ada beberapa perbedaan di antara keduanya.

ArrayList adalah struktur data berbasis indeks yang didukung oleh Array, sehingga memberikan akses acak ke elemennya dengan kinerja sebagai O (1) tetapi LinkedList menyimpan data sebagai daftar node di mana setiap node ditautkan ke node sebelumnya dan berikutnya. Jadi meskipun ada metode untuk mendapatkan elemen menggunakan indeks, secara internal melintasi dari awal untuk mencapai node indeks dan kemudian mengembalikan elemen tersebut, sehingga kinerjanya adalah O (n) yang lebih lambat dari ArrayList.

Penyisipan, penambahan, atau penghapusan elemen lebih cepat di LinkedList dibandingkan dengan ArrayList karena tidak ada konsep untuk mengubah ukuran array atau memperbarui indeks saat elemen ditambahkan di tengah.

LinkedList menggunakan lebih banyak memori daripada ArrayList karena setiap node di LinkedList menyimpan referensi dari elemen sebelumnya dan selanjutnya.

Kelas koleksi mana yang memberikan akses acak dari elemennya?

ArrayList, HashMap, TreeMap, kelas Hashtable menyediakan akses acak ke elemennya. Unduh java collections pdf untuk informasi lebih lanjut.

Apa itu EnumSet?

java.util.EnumSet adalah Set implementasi untuk digunakan dengan tipe enum. Semua elemen dalam set enum harus berasal dari satu jenis enum yang ditentukan, secara eksplisit atau implisit, saat set dibuat. EnumSet tidak disinkronkan dan elemen

null tidak diperbolehkan. Ini juga menyediakan beberapa metode yang berguna seperti copyOf (Collection c), dari (E first, E… rest) dan komplementOf (EnumSet s).

Kelas koleksi mana yang aman untuk thread?

Vektor, Hashtable, Properti, dan Tumpukan adalah kelas yang disinkronkan, sehingga keduanya aman untuk thread dan dapat digunakan di lingkungan multi-thread. Java 1.5 Concurrent API menyertakan beberapa kelas koleksi yang memungkinkan modifikasi koleksi saat iterasi karena mereka bekerja pada tiruan koleksi, sehingga aman digunakan dalam lingkungan multi-thread.

Apa itu Kelas Koleksi bersamaan?

Paket Java 1.5 Concurrent (java.util.concurrent) berisi kelas koleksi thread-safe yang memungkinkan koleksi dimodifikasi saat melakukan iterasi. Secara desain, iterator adalah failfast dan menampilkan ConcurrentModificationException. Beberapa kelas ini adalah CopyOnWriteArrayList, ConcurrentHashMap, CopyOnWriteArraySet.

Hindari ConcurrentModificationException

Contoh CopyOnWriteArrayList

HashMap vs ConcurrentHashMap

Apa itu BlockingQueue?

java.util.concurrent.BlockingQueue adalah Antrean yang mendukung operasi yang menunggu antrean menjadi tidak kosong saat mengambil dan menghapus elemen, dan menunggu hingga ruang tersedia di antrean saat menambahkan elemen. Antarmuka BlockingQueue adalah bagian dari kerangka kerja java collection dan terutama digunakan untuk mengimplementasikan masalah konsumen produsen. Kami tidak perlu khawatir menunggu ketersediaan ruang bagi produsen atau objek untuk konsumen di BlockingQueue karena ditangani oleh kelas implementasi BlockingQueue. Java menyediakan beberapa implementasi BlockingQueue seperti ArrayBlockingQueue, LinkedBlockingQueue, PriorityBlockingQueue, SynchronousQueue, dll.

Apa itu Queue dan Stack, sebutkan perbedaannya?

Baik Queue dan Stack digunakan untuk menyimpan data sebelum memprosesnya. java.util.Queue adalah antarmuka yang kelas implementasinya hadir dalam paket konkuren java. Antrean memungkinkan pengambilan elemen dalam urutan First-In-First-Out (FIFO) tetapi tidak selalu demikian. Ada juga antarmuka Deque yang memungkinkan elemen diambil dari kedua ujung antrean. Tumpukan mirip dengan antrean kecuali memungkinkan elemen diambil dalam urutan Last-InFirst-Out (LIFO). Stack adalah kelas yang memperluas Vektor sedangkan Queue adalah antarmuka.

Apa itu Kelas Koleksi?

java.util.Collections adalah kelas utilitas yang secara eksklusif terdiri dari metode statis yang beroperasi pada atau mengembalikan koleksi. Ini berisi algoritme polimorfik yang beroperasi pada koleksi, "pembungkus", yang mengembalikan koleksi baru yang didukung oleh koleksi tertentu, dan beberapa peluang dan akhir lainnya. Kelas ini berisi metode untuk algoritme kerangka pengumpulan, seperti pencarian biner, pengurutan, pengocokan, pembalikan, dll.

Apa itu antarmuka Comparable dan Comparator?

Java menyediakan antarmuka Comparable yang harus diimplementasikan oleh setiap kelas kustom jika kita ingin menggunakan metode penyortiran Array atau Koleksi. Antarmuka yang dapat dibandingkan memiliki metode bandingkanTo (T obj) yang digunakan oleh metode penyortiran. Kita harus mengganti metode ini sedemikian rupa sehingga mengembalikan bilangan bulat negatif, nol, atau bilangan bulat positif jika objek "ini" lebih kecil dari, sama dengan, atau lebih besar dari objek yang diteruskan sebagai argumen.

Namun, dalam sebagian besar skenario kehidupan nyata, kami ingin menyortir berdasarkan parameter yang berbeda. Misalnya, sebagai CEO, saya ingin mengurutkan karyawan berdasarkan Gaji, seorang SDM ingin mengurutkan mereka berdasarkan usianya. Ini adalah situasi di mana kita perlu menggunakan

Antarmuka pembanding karena implementasi metode Comparable.compareTo (Object o) hanya dapat mengurutkan berdasarkan satu bidang dan kita tidak dapat memilih bidang tempat kita ingin mengurutkan Objek. Metode perbandingan antarmuka pembanding (Object o1, Object o2) perlu diimplementasikan yang membutuhkan dua argumen Object, itu harus diimplementasikan sedemikian rupa sehingga mengembalikan int negatif jika argumen pertama kurang dari yang kedua dan mengembalikan nol jika sama dan positif int jika argumen pertama lebih besar dari yang kedua.

Apa perbedaan antara Interface/antarmuka Comparable dan Comparator?

Antarmuka Comparable dan Comparator digunakan untuk mengurutkan koleksi atau larik objek. Antarmuka yang dapat dibandingkan digunakan untuk menyediakan pengurutan alami objek dan kita dapat menggunakannya untuk menyediakan pengurutan berdasarkan logika tunggal. Antarmuka pembanding digunakan untuk menyediakan algoritme yang berbeda untuk pengurutan dan kita dapat memilih pembanding yang ingin kita gunakan untuk mengurutkan koleksi objek yang diberikan.

Bagaimana kita bisa mengurutkan daftar Objek?

Jika kita perlu mengurutkan array Objects, kita bisa menggunakan Arrays.sort(). Jika kita perlu mengurutkan daftar objek, kita bisa menggunakan Collections.sort(). Kedua kelas ini memiliki kelebihan metode sort() untuk pengurutan natural (menggunakan Comparable) atau pengurutan berdasarkan kriteria (menggunakan

Comparator). Koleksi secara internal menggunakan metode pengurutan Array, sehingga keduanya memiliki kinerja yang sama kecuali Koleksi memerlukan waktu beberapa saat untuk mengonversi daftar ke larik.

Saat meneruskan Collection sebagai argumen ke suatu fungsi, bagaimana kita dapat memastikan bahwa fungsi tersebut tidak dapat mengubahnya?

Kita bisa membuat koleksi hanya-baca menggunakan metode Collections.unmodifiableCollection (Collection c) sebelum meneruskannya sebagai argumen, ini akan memastikan bahwa setiap operasi untuk mengubah koleksi akan membuangUnsupportedOperationException.

Bagaimana cara membuat koleksi tersinkronisasi dari koleksi yang diberikan?

Kita bisa menggunakan Collections.synchronizedCollection (Collection c) untuk mendapatkan koleksi tersinkronisasi (aman-thread) yang didukung oleh koleksi yang ditentukan.

Apa algoritma umum yang diterapkan dalam Collections Framework?

Java Collections Framework menyediakan implementasi algoritma yang umum digunakan seperti pengurutan dan pencarian. Kelas koleksi berisi implementasi metode ini. Sebagian besar algoritme ini berfungsi pada Daftar tetapi beberapa di antaranya dapat diterapkan untuk semua jenis koleksi.

Beberapa di antaranya adalah mengurutkan, mencari, mengocok, nilai min-max.

Apa itu notasi Big-O? Berikan beberapa contoh? Notasi Big-O menggambarkan kinerja suatu algoritma dalam hal jumlah elemen dalam struktur data. Karena class Collection sebenarnya adalah struktur data, kami biasanya cenderung menggunakan notasi Big-O untuk memilih implementasi collection yang akan digunakan berdasarkan waktu, memori, dan performa.

Contoh 1: ArrayList get (indeks i) adalah operasi waktu konstan dan tidak bergantung pada jumlah elemen dalam daftar. Jadi performanya dalam notasi Big-O adalah O (1).

Contoh 2: Pencarian linier pada kinerja array atau daftar adalah O (n) karena kita perlu mencari seluruh daftar elemen untuk menemukan elemen.

Apa praktik terbaik yang terkait dengan Java Collections Framework?

Memilih jenis koleksi yang tepat berdasarkan kebutuhan, misalnya jika ukurannya tetap, kita mungkin ingin menggunakan Array di atas ArrayList. Jika kita harus mengulang Peta dalam urutan penyisipan, kita perlu menggunakan TreeMap. Jika kita tidak ingin duplikat, kita harus menggunakan Set.

Beberapa kelas koleksi memungkinkan untuk menentukan kapasitas awal, jadi jika kita memiliki perkiraan jumlah elemen yang akan kita simpan, kita dapat menggunakannya untuk menghindari pengulangan atau pengubahan ukuran.

Menulis program dalam hal antarmuka bukan implementasi, ini memungkinkan kita untuk mengubah implementasi dengan mudah di lain waktu. Selalu gunakan Generik untuk keamanan tipe dan hindari ClassCastException saat runtime.

Gunakan kelas yang tidak dapat diubah yang disediakan oleh JDK sebagai kunci dalam Peta untuk menghindari implementasi hashCode() dan sama dengan() untuk kelas khusus kita.

Gunakan kelas utilitas Koleksi sebanyak mungkin untuk algoritme atau untuk mendapatkan koleksi yang hanya bisa dibaca, disinkronkan, atau kosong daripada menulis implementasi sendiri. Ini akan meningkatkan penggunaan kembali kode dengan stabilitas yang lebih besar dan kemudahan perawatan yang rendah.

Apa itu Antrian Prioritas Java?

PriorityQueue adalah antrian tak terbatas berdasarkan tumpukan prioritas dan elemen diurutkan dalam urutan aslinya atau kami dapat menyediakan Pembanding untuk memesan pada saat pembuatan. PriorityQueue tidak mengizinkan nilai nol dan kami tidak dapat menambahkan objek apa pun yang tidak memberikan pengurutan alami atau kami tidak memiliki pembanding untuk pengurutan. Java PriorityQueue tidak thread-safe dan menyediakan waktu O (log (n)) untuk operasi enqueing dan dequeing.

Mengapa kita tidak bisa menulis kode sebagai List <Number> numbers = new ArrayList <Integer>() ;?

Generik tidak mendukung pengetikan karena akan menyebabkan masalah dalam mencapai keamanan jenis.

Itulah mengapa Daftar <T> tidak dianggap sebagai subtipe dari Daftar <S> dengan S adalah tipe super dari T. Untuk memahami mengapa tidak diizinkan, mari kita lihat apa yang bisa terjadi jika telah didukung.

```
List<Long> listLong = ArrayList baru<Long>();
listLong.add(Long.valueOf(10)); List<Number>
listNumbers = listLong; // compiler error
 listNumbers.add(Double.valueOf(1.23));
```

Seperti yang Anda lihat dari kode di atas bahwa IF generics akan mendukung subtipe, kita dapat dengan mudah menambahkan Double ke daftar Long yang akan menyebabkan ClassCastException saat runtime saat melintasi daftar Long.

Mengapa kita tidak dapat membuat array umum? atau tulis kode sebagai List <Integer> [] array = new ArrayList <Integer> [10];

Kami tidak diizinkan membuat larik generik karena larik membawa informasi jenis elemennya pada waktu proses. Informasi ini digunakan pada waktu proses untuk menampilkan ArrayStoreException jika jenis elemen tidak cocok dengan jenis yang ditentukan. Karena informasi tipe generik dihapus saat runtime oleh Type Erasure, pemeriksaan penyimpanan array akan diteruskan di tempat yang seharusnya gagal. Mari kita pahami ini dengan kode contoh sederhana.

```
List<Integer>[] intList = new List<Integer>[5]; // compile error
Object[] objArray = intList;
List<Double> doubleList = new ArrayList<Double>();
doubleList.add(Double.valueOf(1.23));
```

objArray[0] = doubleList; // ini seharusnya gagal tetapi akan lolos karena pada saat runtime intList dan doubleList keduanya hanya Daftar

Array adalah kovarian secara alami yaitu S [] adalah subtipe dari T [] setiap kali S adalah subtipe dari T tetapi generik tidak mendukung kovarian atau sub-pengetikan seperti yang kita lihat di pertanyaan terakhir. Jadi jika kita diizinkan untuk membuat array umum, karena penghapusan tipe kita tidak akan mendapatkan pengecualian penyimpanan array meskipun kedua jenis tidak terkait.

Pegecualian Pertanyaan Dan Jawaban Wawancara Java

Java menyediakan pendekatan yang kuat dan berorientasi objek untuk menangani skenario pengecualian yang dikenal sebagai Exception handling Java.

Apa itu Pengecualian di java?

Pengecualian adalah peristiwa kesalahan yang dapat terjadi selama pelaksanaan program dan mengganggu stream normalnya. Pengecualian dapat muncul dari berbagai jenis situasi seperti data yang salah dimasukkan oleh pengguna, kegagalan perangkat keras, kegagalan koneksi jaringan dll. Setiap kali terjadi kesalahan saat menjalankan pernyataan java, objek pengecualian dibuat dan kemudian JRE mencoba untuk menemukan penangan pengecualian untuk menangani pengecualian . Jika penangan pengecualian yang sesuai ditemukan maka objek pengecualian diteruskan ke kode penangan untuk memproses pengecualian, yang dikenal sebagai menangkap pengecualian. Jika tidak ada penangan yang ditemukan maka aplikasi akan melempar pengecualian ke lingkungan runtime dan JRE menghentikan program.

Kerangka kerja exception handling Java digunakan untuk menangani kesalahan waktu proses saja, kesalahan waktu kompilasi tidak ditangani oleh kerangka kerja exception handling.

Apa KeywordException handling di Java? Ada empat keyword yang digunakan dalam exception handling java.

throw: Kadang-kadang kita secara eksplisit ingin membuat objek pengecualian dan kemudian membuangnya untuk menghentikan pemrosesan normal program. keyword throw digunakan untuk membuang pengecualian ke runtime untuk menanganinya.

throws: Saat kita melempar pengecualian yang dicentang dalam sebuah metode dan tidak menanganinya, maka kita perlu menggunakan keyword lemparan dalam tanda tangan metode agar program pemanggil mengetahui pengecualian yang mungkin dilemparkan oleh metode tersebut. Metode pemanggil mungkin menangani pengecualian ini atau menyebarkannya ke metode pemanggil menggunakan keyword lemparan. Kita bisa menyediakan beberapa pengecualian dalam klausa throws dan juga bisa digunakan dengan metode main().

try-catch: Kami menggunakan blok try-catch untuk exception handling dalam kode kami. try adalah awal dari blok dan catch berada di akhir blok try untuk menangani pengecualian. Kita dapat memiliki beberapa catch blockan dengan mencoba dan blok coba-tangkap juga dapat bersarang. catch block membutuhkan parameter yang harus bertipe Exception. Akhirnya: finally block adalah opsional dan hanya dapat digunakan dengan blok coba-tangkap. Karena pengecualian menghentikan proses eksekusi, kita mungkin memiliki beberapa sumber daya terbuka yang tidak akan ditutup, sehingga kita dapat menggunakan finally block.

Finally : blok selalu dieksekusi, apakah pengecualian terjadi atau tidak.

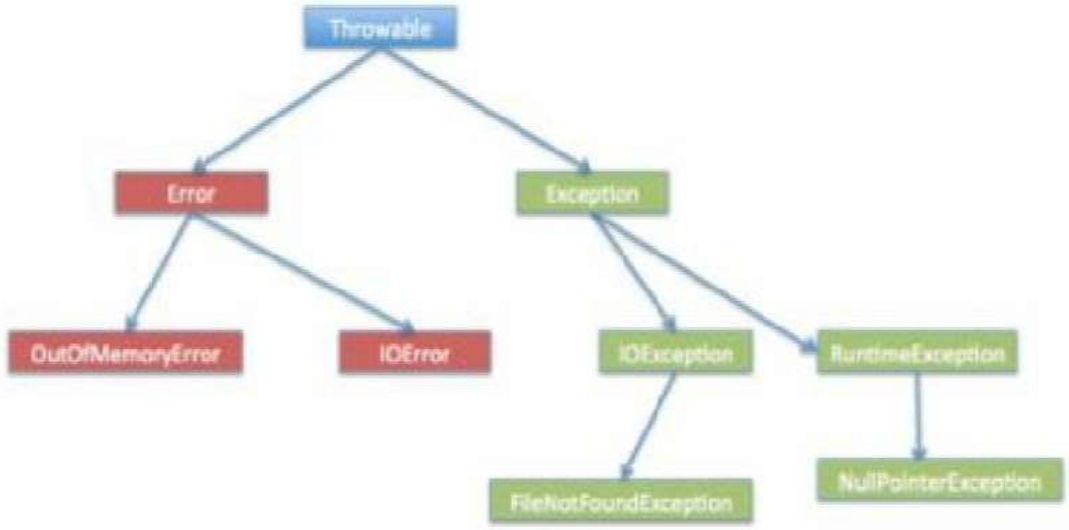
Jelaskan Hirarki Pengecualian Java?

Pengecualian Java bersifat hierarkis dan peinheritance digunakan untuk mengkategorikan berbagai jenis pengecualian. Throwable adalah kelas induk Hierarki Pengecualian Java dan memiliki dua objek turunan - Kesalahan dan Pengecualian. Pengecualian dibagi lagi menjadi pengecualian yang diperiksa dan pengecualian waktu proses.

Kesalahan adalah skenario luar biasa yang berada di luar cakupan aplikasi dan tidak mungkin diantisipasi dan dipulihkan darinya, misalnya kegagalan perangkat keras, kerusakan JVM, atau kesalahan kehabisan memori.

Pengecualian Tercentang adalah skenario luar biasa yang dapat kita antisipasi dalam program dan mencoba memulihkannya, misalnya FileNotFoundException. Kita harus menangkap pengecualian ini dan memberikan pesan yang berguna kepada pengguna dan mencatatnya dengan benar untuk tujuan debugging. Pengecualian adalah kelas induk dari semua Pengecualian yang Dicentang.

Pengecualian Waktu Proses disebabkan oleh pemrograman yang buruk, misalnya mencoba mengambil elemen dari Array. Kita harus memeriksa panjang array terlebih dahulu sebelum mencoba mengambil elemen jika tidak maka ArrayIndexOutOfBoundException akan muncul saat runtime. RuntimeException adalah kelas induk dari semua pengecualian waktu proses.



Apa metode penting dari Kelas Pengecualian Java?

Pengecualian dan semua subkelasnya tidak menyediakan metode khusus apa pun dan semua metode ditentukan dalam kelas dasar Throwable.

String getMessage() - Metode ini mengembalikan pesan String of Throwable dan pesan dapat diberikan sambil membuat pengecualian melalui konstruktornya.

String getLocalizedMessage() - Metode ini disediakan sehingga subclass bisa menimpanya untuk memberikan pesan khusus lokal ke program pemanggil. Implementasi kelas yang dapat dilempar dari metode ini cukup menggunakan metode getMessage() untuk mengembalikan pesan pengecualian.

sinkronisasi Throwable getCause() - Metode ini mengembalikan penyebab pengecualian atau null id penyebabnya tidak diketahui.

String toString() - Metode ini mengembalikan informasi tentang Throwable dalam format String, String yang dikembalikan berisi nama kelas Throwable dan pesan lokal.

void printStackTrace() - Metode ini mencetak informasi jejak tumpukan ke stream kesalahan standar, metode ini kelebihan beban dan kita dapat meneruskan PrintStream atau PrintWriter sebagai argumen untuk menulis informasi jejak tumpukan ke file atau stream.

Jelaskan Fitur Java 7 ARM dan blok multi-catch?

Jika Anda menangkap banyak pengecualian dalam satu blok percobaan, Anda akan melihat bahwa kode catch block terlihat sangat jelek dan sebagian besar terdiri dari kode yang berlebihan untuk mencatat kesalahan, mengingat ini Java 7 salah satu fiturnya adalah blok multi-tangkap di mana kita bisa menangkap beberapa pengecualian dalam satu catch blockan. Catch block dengan fitur ini terlihat seperti di bawah ini:

```
catch(IOException | SQLException | Exception ex){
 logger.error(ex);
throw new MyException(ex.getMessage());
}
```

Sering kali, kami menggunakan last block hanya untuk menutup sumber daya dan terkadang kami lupa menutupnya dan mendapatkan pengecualian waktu proses saat sumber daya habis. Pengecualian ini sulit untuk di-debug dan kami mungkin perlu melihat ke setiap tempat di mana kami menggunakan jenis sumber daya tersebut untuk memastikan kami menutupnya. Jadi java 7 salah satu perbaikannya adalah try-with-resources di mana kita dapat membuat sumber daya dalam pernyataan try itu sendiri dan menggunakannya di dalam blok coba-tangkap. Ketika eksekusi keluar dari blok coba-tangkap, lingkungan runtime secara otomatis menutup sumber daya ini. Contoh blok trycatch dengan perbaikan ini adalah:

```
try (MyResource mr = new MyResource()) {
System.out.println("MyResource created in try-with-resources");
} catch (Exception e) {
e.printStackTrace();
}
```

Apa perbedaan antara Pengecualian yang Dicentang dan Tidak Dicentang di Java?

Pengecualian yang Dicentang harus ditangani dalam kode menggunakan blok coba-tangkap

atau cara lain main() harus menggunakan keyword throws untuk memberi tahu JRE tentang pengecualian ini yang mungkin terlempar dari program. Tidak dicentang

Pengecualian tidak diperlukan untuk ditangani dalam program atau menyebutkannya dalam klausul lemparan. Exception adalah kelas super dari semua pengecualian yang dicentang sedangkan RuntimeException adalah kelas super dari semua pengecualian yang tidak dicentang.

Pengecualian yang dicentang adalah skenario kesalahan yang tidak disebabkan oleh program, misalnya FileNotFoundException dalam membaca file yang tidak ada, sedangkan pengecualian yang tidak dicentang sebagian besar disebabkan oleh pemrograman yang buruk, misalnya NullPointerException saat menjalankan metode pada referensi objek tanpa memastikan bahwa itu bukan nol.

Apa perbedaan antara keyword lempar dan lempar di java?

keyword throws digunakan dengan tanda tangan metode untuk menyatakan pengecualian yang mungkin dilemparkan metode sedangkan keyword throw digunakan untuk mengganggu stream program dan menyerahkan objek pengecualian ke runtime untuk menanganinya.

Bagaimana cara menulis pengecualian khusus di Java?

Kita dapat memperluas kelas Exception atau subkelasnya yang mana pun untuk membuat kelas pengecualian kustom kita. Kelas pengecualian khusus dapat memiliki variabel dan metode sendiri yang dapat kita gunakan untuk meneruskan kode kesalahan atau informasi terkait pengecualian lainnya ke penangan pengecualian.

Contoh sederhana dari pengecualian khusus ditampilkan di bawah ini.

```
MyException.java
package com.journaldev.exceptions;
import java.io.IOException;
public class MyException extends IOException {
private static final long serialVersionUID = 4664456874499611218L;
private String errorCode="Unknown_Exception";
public MyException(String message, String errorCode){
super(message);
this.errorCode=errorCode;
}    public String getErrorCode(){
return this.errorCode;  } }
```

Apa itu OutOfMemoryError di Java?

OutOfMemoryError di Java adalah subkelas java.lang.VirtualMachineError dan dilempar oleh JVM saat heap kehabisan memori. Kami dapat memperbaiki kesalahan ini dengan menyediakan lebih banyak memori untuk menjalankan aplikasi java melalui opsi java.

$> java Programku -Xms1024m -Xmx1024m -XX: PermSize = 64M-XX: MaxPermSize = 256m

Apa sajakah skenario berbeda yang menyebabkan "Exception in thread main"?

Beberapa skenario pengecualian utas utama umum adalah:

Pengecualian di thread main java.lang.UnsupportedClassVersionError: Pengecualian ini muncul saat kelas java Anda dikompilasi dari versi JDK lain dan Anda mencoba menjalankannya dari versi java lain.

Pengecualian di thread main java.lang.NoClassDefFoundError: Ada dua varian pengecualian ini. Yang pertama adalah tempat Anda memberikan nama lengkap kelas dengan ekstensi .class. Skenario kedua adalah ketika Kelas tidak ditemukan.

Pengecualian di thread main java.lang.NoSuchMethodError: main: Pengecualian ini muncul saat Anda mencoba menjalankan kelas yang tidak memiliki metode utama.

Pengecualian di thread "main" java.lang.ArithmeticException: Setiap kali ada pengecualian yang dilemparkan dari metode utama, ia mencetak pengecualian adalah konsol. Bagian pertama menjelaskan bahwa pengecualian dilempar dari metode utama, bagian kedua mencetak nama kelas pengecualian dan kemudian setelah titik dua, mencetak pesan pengecualian.

Apa perbedaan antara final, akhirnya, dan finalisasi di Java?

final dan terakhir adalah keyword di java sedangkan finalize adalah metode. keyword terakhir dapat digunakan dengan variabel kelas sehingga mereka tidak dapat ditugaskan kembali, dengan kelas untuk menghindari perluasan oleh kelas dan dengan metode untuk menghindari penimpaan oleh subkelas, akhirnya keyword digunakan dengan blok coba-tangkap untuk memberikan pernyataan yang akan selalu dieksekusi bahkan jika beberapa pengecualian muncul, biasanya akhirnya digunakan untuk menutup sumber daya. metode finalize() dijalankan oleh Garbage Collector sebelum objek dimusnahkan, ini cara yang bagus untuk memastikan semua sumber daya global ditutup. Dari ketiganya, hanya akhirnya yang terkait dengan exception handling java.

Apa yang terjadi jika pengecualian dilemparkan oleh metode utama?

Ketika pengecualian dilemparkan oleh metode main(), Java Runtime menghentikan program dan mencetak pesan pengecualian dan pelacakan tumpukan di konsol sistem.

Bisakah kita memiliki catch block kosong?

Kita dapat memiliki catch block kosong tetapi itu adalah contoh pemrograman terburuk. Kita tidak boleh memiliki blok catch kosong karena jika pengecualian ditangkap oleh blok itu, kita tidak akan memiliki informasi tentang pengecualian tersebut dan akan menjadi mimpi buruk untuk men-debugnya. Setidaknya harus ada pernyataan logging untuk mencatat detail pengecualian di konsol atau file log.

Sediakan beberapa Praktik Terbaik Exception handling Java?

Beberapa praktik terbaik yang terkait dengan Exception handling Java adalah:

Gunakan Pengecualian Khusus untuk kemudahan debugging.Lempar Pengecualian Lebih Awal (Gagal-Cepat) dalam program.

Catch Exception di akhir program, biarkan pemanggil menangani pengecualian tersebut.

Gunakan fitur Java 7 ARM untuk memastikan sumber daya ditutup atau gunakan finally block untuk menutupnya dengan benar.

Selalu catat pesan pengecualian untuk tujuan debugging. Gunakan blok multi-catch untuk menutup lebih bersih.

Gunakan pengecualian khusus untuk menampilkan satu jenis pengecualian dari API aplikasi Anda.

Ikuti konvensi penamaan, selalu diakhiri dengan Exception.

Dokumentasikan Pengecualian yang Dilempar dengan metode menggunakan @throws di javadoc.Pengecualian itu mahal, jadi buang hanya jika memang masuk akal. Jika tidak, Anda dapat menangkapnya dan memberikan respons nol atau kosong.

BAB 21
WAWANCARA CORE JAVA

Apa perbedaan antara JDK, JRE dan JVM?

JVM- JVM adalah singkatan dari Java Virtual Machine, ini adalah mesin abstrak yang menyediakan lingkungan runtime tempat bytecode java dapat dijalankan. JVM tersedia untuk banyak platform perangkat keras dan perangkat lunak (jadi JVM bergantung pada bentuk pelat).

JRE - JRE adalah singkatan dari Java Runtime Environment. Ini adalah implementasi JVM dan ada secara fisik.

JDK - JDK adalah singkatan dari Java Development Kit. Itu ada secara fisik. Ini berisi alat pengembangan JRE +.

Berapa banyak jenis area memori yang dialokasikan oleh JVM?

Banyak jenis:

- *Kelas (Metode) Area*
- *Tumpukan*
- *Tumpukan*
- *Daftar Penghitung Program*
- *Tumpukan Metode Asli*

Apa itu kompiler JIT?

Kompiler Just-In-Time (JIT): Digunakan untuk meningkatkan kinerja. JIT mengkompilasi bagian dari kode byte yang memiliki fungsi serupa pada saat yang sama, dan karenanya mengurangi jumlah waktu yang dibutuhkan untuk kompilasi. Di sini istilah "kompilator" mengacu pada penerjemah dari set instruksi mesin virtual Java (JVM) ke set instruksi dari CPU tertentu.

Apa itu platform?

Platform pada dasarnya adalah lingkungan perangkat keras atau perangkat lunak tempat program berjalan. Ada dua jenis platform berbasis perangkat lunak dan berbasis perangkat keras. Java menyediakan platform berbasis perangkat lunak.

Apa perbedaan utama antara platform Java dan platform lain?

Platform Java berbeda dari kebanyakan platform lain dalam arti bahwa itu adalah platform berbasis perangkat lunak yang berjalan di atas platform berbasis perangkat keras lainnya. Ini memiliki dua komponen:

Lingkungan Waktu Proses

API (Antarmuka Pemrograman Aplikasi)

Apa yang memberi Java sifat 'tulis sekali dan jalankan di mana saja'?

Bytecode. Java dikompilasi menjadi kode byte yang merupakan bahasa perantara antara kode sumber dan kode mesin. Kode byte ini tidak spesifik platform dan karenanya dapat diumpankan ke platform apa pun.

Apa itu classloader?

Classloader adalah subsistem JVM yang digunakan untuk memuat class dan interface. Ada banyak jenis classloader, mis. Pemuat kelas bootstrap, Pemuat kelas ekstensi, Pemuat system class, pemuat kelas plugin, dll.

Apakah nama file .java kosong adalah nama file sumber yang valid?

Ya, simpan file java Anda hanya dengan .java, kompilasi dengan javac .java dan jalankan dengan java yourclassname Mari kita ambil contoh sederhana:

```
//save by .java only
class A{
public static void main(String args[]){
System.out.println("Hello java");
}
}
//compile by javac .java
//run by    java A
```

compile it by **javac .java**
run it by **java A**

Apakah keyword delete, next, main, exit, atau null di java?

Tidak.

Jika saya tidak memberikan argumen apa pun pada baris perintah, maka array String metode Utama akan kosong atau null?

Ini kosong. Tapi tidak nol.

Bagaimana jika saya menulis kehampaan publik statis dan bukan kehampaan statis publik?

Program mengkompilasi dan berjalan dengan baik.

Berapa nilai default dari variabel lokal?

Variabel lokal tidak diinisialisasi ke nilai default apa pun, baik primitif maupun referensi objek.

Core Java – Konsep Oop : Pertanyaan Wawancara Oop Awal

Ada lebih dari 50 pertanyaan wawancara OOP (Pemrograman Berorientasi Objek dan Sistem).

Tetapi mereka telah dikategorikan dalam banyak bagian seperti pertanyaan wawancara konstruktor, pertanyaan wawancara statis, pertanyaan Wawancara Inheritance, pertanyaan wawancara abstraksi, pertanyaan wawancara polimorfisme, dll. Untuk pemahaman yang lebih baik.

Apa perbedaan antara bahasa pemrograman berorientasi objek dan bahasa pemrograman berbasis objek?

Bahasa pemrograman berbasis objek mengikuti semua fitur OOP kecuali Inheritance. Contoh bahasa pemrograman berbasis objek adalah JavaScript, VBScript, dll.

Berapakah nilai awal referensi objek yang didefinisikan sebagai variabel instan?

Referensi objek semuanya diinisialisasi ke null di Java.

Core Java – Konsep Oop : Pertanyaan Pembuat Wawancara

Apa itu konstruktor?

Pembuat adalah seperti metode yang digunakan untuk menginisialisasi keadaan suatu objek. Itu dipanggil pada saat pembuatan objek.

Apa tujuan dari konstruktor default?

Konstruktor default memberikan nilai default ke objek. Kompilator java membuat konstruktor default hanya jika tidak ada konstruktor di kelas. Selengkapnya ...

Apakah konstruktor mengembalikan nilai apa pun?

Jawab: ya, itu adalah instance saat ini (Anda tidak dapat menggunakan tipe pengembalian namun mengembalikan nilai). Lebih jelasnya ...

Apakah konstruktor diwariskan?

Tidak, konstruktor tidak diwariskan.

Bisakah Anda membuat konstruktor final?

Tidak, konstruktor tidak bisa final.

Core Java – Konsep Oop : Keyword Statis Pada Pertanyaan Wawancara

Apa itu variabel statis?

variabel statis digunakan untuk merujuk properti umum dari semua objek (yang tidak unik untuk setiap objek) mis. nama perusahaan karyawan, nama mahasiswa, dll. Variabel statis hanya mendapat memori satu kali di area kelas pada saat pemuatan kelas.

Apa itu metode statis?

Metode statis milik kelas, bukan objek kelas. Metode statis dapat dipanggil tanpa perlu membuat instance kelas. metode statis dapat mengakses anggota data statis dan dapat mengubah nilainya.

Mengapa metode utama statis?

Karena objek tidak perlu memanggil metode statis jika merupakan metode non-statis, maka jvm membuat objek terlebih dahulu kemudian memanggil metode main() yang akan menimbulkan masalah alokasi memori tambahan. Selengkapnya ...

Apa itu blok statis?

Digunakan untuk menginisialisasi anggota data statis. Itu dieksekusi sebelum metode utama pada saat pemuatan kelas.

Bisakah kita menjalankan program tanpa metode main()?

Jawaban : Ya, salah satunya adalah blok statis.

Bagaimana jika pengubah statis dihapus dari tanda tangan metode utama?

Kompilasi program. Tetapi pada saat runtime muncul kesalahan "NoSuchMethodError".

Apa perbedaan antara metode statis (kelas) dan metode instance?

Metode Statis atau Class	Metode Instan
Sebuah metode i.s. dideklarasikan sebagai statis dikenal sebagai metode statis.	Metode yang tidak dideklarasikan sebagai statis dikenal sebagai metode instance.
Objek tidak diperlukan untuk memanggil metode statis.	Objek diperlukan untuk memanggil metode instance.
Anggota non-statis (instance) tidak dapat diakses dalam konteks statis (metode statis, blok statis, dan kelas bertingkat statis) secara langsung.	variabel statis dan non-statis keduanya dapat diakses dalam metode instance.
Misalnya: public static int cube (int n) {return n * n * n;}	Misalnya: public void msg() {...}.

Core Java – Konsep Oop : Pertanyaan Wawancara Inheritance

Apa ini di java?

Ini adalah keyword yang mengacu pada objek saat ini.

Apa itu Inheritance?

Peinheritance adalah mekanisme di mana satu objek memperoleh semua properti dan perilaku objek lain dari kelas lain. Ini mewakili hubungan IS-A. Ini digunakan untuk Resusabilitas Kode dan Penimpaan Metode.

Kelas mana yang merupakan superclass untuk setiap kelas?

Kelas objek.

Mengapa multiple inheritance tidak didukung di java?

Untuk mengurangi kerumitan dan menyederhanakan bahasa, multiple inheritance tidak didukung di java untuk class.

Apakah komposisi itu?

Memegang referensi kelas lain di dalam kelas lain disebut komposisi.

Apa perbedaan antara Agregasi dan Komposisi?

Agregasi mewakili hubungan yang lemah sedangkan komposisi mewakili hubungan yang kuat. Contoh: sepeda memiliki indikator (agregasi) tetapi sepeda memiliki mesin (kompos).

Mengapa Java tidak mendukung pointer?

Pointer adalah variabel yang mengacu pada alamat memori. Mereka tidak digunakan di java karena tidak aman (tidak aman) dan rumit untuk dipahami.

Apa itu super di java?

Ini adalah keyword yang merujuk ke objek kelas induk langsung.

Bisakah Anda menggunakan this() dan super() keduanya dalam konstruktor?

Tidak. Karena super() atau this() harus menjadi pernyataan pertama.

Apa itu kloning objek?

Kloning objek digunakan untuk membuat salinan yang tepat dari suatu objek.

Core Java – Konsep Oop : Metode Pertanyaan Wawancara Yang Berlebihan

Apa metode overloading?

Jika kelas memiliki beberapa metode dengan nama yang sama tetapi parameter berbeda, itu dikenal sebagai Metode Overloading. Ini meningkatkan keterbacaan program.

Mengapa metode overloading tidak dimungkinkan dengan mengubah tipepengembalian di java?

Karena ambiguitas.

Bisakah kita membebani metode main()?

Ya, Anda bisa memiliki banyak metode main() di kelas dengan membebani metode utama.

Core Java – Konsep Oop : Metode Mengesampingkan Pertanyaan Wawancara

Apa metode overriding:

Jika sebuah subclass menyediakan implementasi spesifik dari sebuah metode yang telah disediakan oleh kelas induknya, itu dikenal sebagai Method Overriding. Ini digunakan untuk polimorfisme runtime dan untuk menyediakan implementasi spesifik dari metode tersebut.

Bisakah kita mengganti metode statis?

Tidak, Anda tidak dapat mengganti metode statis karena mereka adalah bagian dari kelas, bukan objek.

Mengapa kami tidak dapat mengganti metode statis?

Karena metode statis adalah bagian dari kelas dan terikat dengan kelas sedangkan metode instance terikat dengan objek dan statis mendapat memori di area kelas dan instance mendapat memori di heap.

Bisakah kita mengganti metode kelebihan beban?

Iya.

Perbedaan antara metode Overloading dan Overriding.

Metode Overloading	Metode Overriding
Metode Overloading meningkatkan keterbacaan program.	Metode Overriding mengimplementasi dari metode yang telah disediakan oleh kelas supernya.
Metode Overloading terjadi di dalam kelas.	Metode Overriding Penimpaan metode terjadi di dua kelas yang memiliki hubungan IS-A.
Dalam hal ini, parameter harus berbeda	Dalam hal ini, parameter harus sama.

Bisakah Anda memiliki fungsi virtual di Java?

Ya, semua fungsi di Java adalah virtual secara default.

Apa itu tipe pengembalian kovarian?

Sekarang, sejak java5, dimungkinkan untuk mengganti metode apa pun dengan mengubah tipe kembalian jika tipe kembalian dari metode subclass overriding adalah tipe subclass. Ini dikenal sebagai tipe pengembalian kovarian.

Core Java – Konsep Oop : Keyword Terakhir Pertanyaan Wawancara

Apa variabel terakhir?

Jika Anda menjadikan variabel apa pun sebagai final, Anda tidak dapat mengubah nilai variabel akhir (Ini akan menjadi konstan).

Apa metode terakhir?

Metode terakhir tidak dapat diganti.

Apakah kelas terakhir itu?

Kelas terakhir tidak bisa

Apa itu variabel akhir kosong?

Variabel terakhir, tidak diinitalkan pada saat deklarasi, dikenal sebagai variabel final kosong.

Bisakah kita menginisialisasi variabel akhir kosong?

Ya, hanya dalam konstruktor jika non-statis. Jika itu adalah variabel final kosong statis, itu dapat diinisialisasi hanya di blok statis.

Bisakah Anda mendeklarasikan metode utama sebagai final?

Ya, seperti, public static final void main (String [] args) {}.

Core Java – Konsep Oop : Pertanyaan Wawancara Polimorfisme, Abstraksi Dan Paket

Apa itu Runtime Polymorphism?

Polimorfisme waktu proses atau pengiriman metode dinamis adalah proses di mana panggilan ke metode yang diganti diselesaikan pada waktu proses daripada pada waktu kompilasi. Dalam proses ini, metode yang diganti dipanggil melalui variabel referensi kelas super. Penentuan metode pemanggilan didasarkan pada objek yang dirujuk oleh variabel referensi.

Dapatkah Anda mencapai Polimorfisme Runtime oleh anggota data?

Tidak.

Apa perbedaan antara Pengikatan Statis dan Pengikatan Dinamis?

Dalam kasus jenis pengikatan statis objek ditentukan pada waktu kompilasi sedangkan dalam jenis objek pengikatan dinamis ditentukan saat runtime.

Core Java – Konsep Oop : Pertanyaan Wawancara Abstraksi

Apa itu abstraksi?

Abstraksi adalah proses menyembunyikan detail implementasi dan hanya menampilkan fungsionalitas kepada pengguna. Abstraksi memungkinkan Anda fokus pada apa yang dilakukan objek alih-alih bagaimana melakukannya.

Apa perbedaan antara Abstraksi dan Enkapsulasi?

Abstraksi menyembunyikan detail implementasi sedangkan enkapsulasi membungkus kode dan data menjadi satu unit.

Apa itu kelas abstrak?

Kelas yang dideklarasikan sebagai abstrak dikenal sebagai kelas abstrak. Ini perlu diperpanjang dan metodenya diterapkan. Itu tidak bisa dipakai.

Bisakah ada metode abstrak tanpa kelas abstrak?

Tidak, jika ada metode abstrak di kelas, kelas itu harus abstrak.

Bisakah Anda menggunakan abstrak dan final keduanya dengan sebuah metode?

Tidak, karena metode abstrak perlu diganti sedangkan Anda tidak dapat mengganti metode akhir.

Apakah mungkin untuk membuat instance kelas abstrak?

Tidak, kelas abstrak tidak pernah dapat dibuat instance-nya.

Apa itu Interface/antarmuka?

Interface/Antarmuka adalah cetak biru kelas yang memiliki konstanta statis dan metode abstrak. Dapat digunakan untuk mencapai abstraksi penuh dan peinheritance ganda.

Bisakah Anda mendeklarasikan metode antarmuka statis?

Tidak, karena metode antarmuka abstrak secara default, dan keyword statis dan abstrak tidak dapat digunakan bersama.

Bisakah Antarmuka menjadi final?

Tidak, karena implementasinya disediakan oleh kelas lain.

Apa itu antarmuka penanda?

Antarmuka yang tidak memiliki anggota dan metode data dikenal sebagai antarmuka penanda, misalnya Serializable, Cloneable, dll.

Apa perbedaan antara Kelas Abstrak dan Antarmuka?

Kelas Abstrak	Interface
Kelas abstrak dapat memiliki isi metode (metode non-abstrak).	Antarmuka hanya memiliki metode abstrak
Kelas abstrak dapat memiliki variabel instan.	Antarmuka tidak boleh memiliki variabel contoh.
Kelas abstrak dapat memiliki konstruktor	Antarmuka tidak boleh memiliki konstruktor.
Kelas abstrak dapat memiliki metode statis.	Antarmuka tidak boleh memiliki metode statis.
Anda dapat memperluas satu kelas abstrak.	Anda dapat menerapkan banyak antarmuka.

Bisakah kita mendefinisikan pengubah privat dan dilindungi untuk variabel dalam *antarmuka?*

Tidak, mereka secara implisit bersifat publik.

Kapan referensi objek dapat dikirim ke referensi antarmuka?

Referensi objek dapat dikirim ke referensi antarmuka saat objek mengimplementasikan antarmuka yang direferensikan.

Core Java – Konsep Oop : Paket Pertanyaan Wawancara

Apa itu paket?

Paket adalah sekelompok jenis antarmuka kelas dan sub-paket yang serupa. Ini memberikan perlindungan akses dan menghilangkan tabrakan penamaan.

Apakah saya perlu mengimpor paket java.lang sewaktu-waktu? Kenapa?

Tidak. Ini secara default dimuat secara internal oleh JVM.

Bisakah saya mengimpor paket/kelas yang sama dua kali? Akankah JVM memuat paket dua kali saat runtime?

Seseorang dapat mengimpor paket yang sama atau kelas yang sama beberapa kali. Baik compiler maupun JVM tidak mengeluhkannya, tetapi JVM akan memuat kelas secara internal hanya sekali tidak peduli berapa kali Anda mengimpor kelas yang sama.

Apa itu impor statis?

Dengan impor statis, kita dapat mengakses anggota statis kelas secara langsung, tidak ada yang harus memenuhi syarat dengan nama kelas.

BAB 22
EXCEPTION JAVA DAN PERTANYAAN WAWANCARA

Apa itu Exception handling?

Exception Handling adalah mekanisme untuk menangani error runtime. Ini terutama digunakan untuk menangani pengecualian yang dicentang.

Apa perbedaan antara Pengecualian yang Dicentang dan Pengecualian yang Tidak Dicentang?

Exception yang Dicentang:

Kelas-kelas yang memperluas kelas Throwable kecuali RuntimeException dan Error dikenal sebagai pengecualian yang diperiksa misalnya, IOException, SQLException, dll. Pengecualian yang dicentang diperiksa pada waktu kompilasi.

Exception yang Tidak Dicentang:

Kelas yang memperluas RuntimeException dikenal sebagai pengecualian yang tidak dicentang, mis. ArithmeticException, NullPointerException, dll. Pengecualian yang tidak dicentang tidak diperiksa pada waktu kompilasi.

Apa kelas dasar untuk Error dan Exception?

Dapat dibuang.

Apakah perlu bahwa setiap blok percobaan harus diikuti oleh catch block?

Tidak perlu bahwa setiap blok percobaan harus diikuti oleh catch block. Ini harus diikuti oleh catch blockan ATAU blok akhirnya. Dan pengecualian apa pun yang mungkin dilemparkan harus dideklarasikan dalam klausa lemparan metode.

Apa finally block?

Finally block adalah blok yang selalu dieksekusi.

Bisakah finally block digunakan tanpa tangkapan?

Ya, dengan coba blokir. akhirnya harus diikuti dengan mencoba atau menangkap.

Apakah ada kasus dimana akhirnya tidak akan dieksekusi?

finally block tidak akan dijalankan jika program keluar (baik dengan memanggil System.exit() atau dengan menyebabkan kesalahan fatal yang menyebabkan proses dibatalkan).

Apa perbedaan antara lemparan dan lemparan?

Throw keyword	throws keyword
Throw digunakan untuk melempar pengecualian secara eksplisit.	Throws digunakan untuk mendeklarasikan pengecualian

pengecualian yang dicentang tidak dapat disebarkan hanya dengan throw.	pengecualian yang diperiksa dapat disebarkan dengan Throws
Throw diikuti dengan sebuah instance	Throws diikuti oleh kelas
Thow digunakan dalam metode ini.	Throws digunakan dengan tanda tangan metode
Anda tidak dapat membuat banyak pengecualian	Anda dapat mendeklarasikan beberapa pengecualian, mis. public void method() melempar IOException, SQLException.

Bisakah pengecualian dicabut kembali?

Iya.

Dapatkah metode subclass overriding mendeklarasikan pengecualian jika metode kelas induk tidak menampilkan pengecualian?

Ya tapi hanya pengecualian yang tidak dicentang tidak dicentang.

Apa itu propagasi pengecualian?

Meneruskan objek pengecualian ke metode pemanggilan dikenal sebagai propagasi pengecualian.

Core Java : Pernyataan Wawancara Penanganan String

Ada diberikan daftar pertanyaan wawancara penanganan string dengan jawaban singkat dan tajam. Jika Anda tahu pertanyaan wawancara penanganan string.

Apa yang dimaksud dengan immutable dalam istilah String?

Arti sederhana dari kekekalan tidak dapat diubah atau tidak dapat diubah. Setelah objek string dibuat, nilainya tidak dapat diubah

Mengapa objek string tidak dapat diubah di java?

Karena java menggunakan konsep string literal. Misalkan ada 5 variabel referensi, semuanya mengacu pada satu objek "sachin". Jika satu variabel referensi mengubah nilai objek, maka akan terpengaruh ke semua variabel referensi. Itulah mengapa objek string tidak dapat diubah di java.

Berapa banyak cara kita dapat membuat objek string?

Ada dua cara untuk membuat objek string, dengan string literal dan dengan keyword baru.

Berapa banyak objek yang akan dibuat dalam kode berikut?

```
String s1="Welcome";
String s2="Welcome";
String s3="Welcome";
```

Mengapa java menggunakan konsep string literal?

Untuk membuat Java lebih hemat memori (karena tidak ada objek baru yang dibuat jika sudah ada dalam kumpulan konstan string).

Berapa banyak objek yang akan dibuat dalam kode berikut?

```
String s = new String("Welcome");
```

Dua objek, satu di kumpulan konstan string dan lainnya di non-kumpulan (heap).

Apa perbedaan mendasar antara objek string dan stringbuffer?

String adalah objek yang tidak bisa diubah. StringBuffer adalah objek yang bisa berubah.

Apa perbedaan antara StringBuffer dan StringBuilder?

StringBuffer disinkronkan sedangkan StringBuilder tidak disinkronkan.

Bagaimana cara membuat kelas yang tidak dapat diubah di java?

Kita dapat membuat kelas yang tidak dapat diubah sebagai kelas String dengan mendefinisikan kelas akhir dan

Apa tujuan metode toString() di java?

Metode toString() mengembalikan representasi string dari objek apa pun. Jika Anda mencetak objek apa pun, compiler java secara internal memanggil metode toString() pada objek tersebut. Jadi menimpa metode toString(), mengembalikan keluaran yang diinginkan, dapat berupa status objek, dll. Bergantung pada implementasi Anda.

BAB 23
CORE JAVA – KONSEP OOP
PERTANYAAN WAWANCARA TENTANG INTERFACE/ANTARMUKA

Apa itu nested class?

Kelas yang dideklarasikan di dalam kelas lain dikenal sebagai nested class. Ada 4 tipe inner class anggota kelas, inner class lokal, inner class tanpa nama dan nested class statis.

Apakah ada perbedaan antara nested class dan inner class?

Ya, inner class adalah nested class non-statis, yaitu inner class adalah bagian dari kelas bertingkat.

Bisakah kita mengakses variabel lokal non-final, di dalam inner class lokal?

Tidak, variabel lokal harus konstan jika Anda ingin mengaksesnya di inner class lokal.

Apa itu ested interface?

Setiap antarmuka yang dideklarasikan di dalam antarmuka atau kelas, dikenal sebagai ested interface. Ini statis secara default.

Bisakah kelas memiliki antarmuka?

Ya, ini dikenal sebagai ested interface.

Bisakah Antarmuka memiliki kelas?

Ya, mereka statis secara implisit.

Garbage Collection Java, Arus I/O, Serialisasi Dan Pertanyaan Wawancara Jaringan

Apa itu Garbage Collection?

Garbage Collection adalah proses mengambil kembali objek yang tidak digunakan waktu proses. Ini dilakukan untuk manajemen memori.

Apa itu gc()?

gc() adalah daemon thread.gc() metode yang didefinisikan di system class yang digunakan untuk mengirim permintaan ke JVM untuk melakukan Garbage Collection.

Apa tujuan dari metode finalize()?

metode finalize() dipanggil tepat sebelum objek dikumpulkan sampah. Ini digunakan untuk melakukan pemrosesan pembersihan.

Kelas yang dideklarasikan di dalam kelas lain dikenal sebagai nested class. Ada 4 tipe inner class anggota kelas, inner class lokal, inner class tanpa nama dan nested class statis.

Apakah ada perbedaan antara nested class dan inner class?

Ya, inner class adalah nested class non-statis, yaitu inner class adalah bagian dari kelas bertingkat.

Bisakah kita mengakses variabel lokal non-final, di dalam inner class lokal?

Tidak, variabel lokal harus konstan jika Anda ingin mengaksesnya di inner class lokal.

Apa itu nersted interface?

Setiap antarmuka yang dideklarasikan di dalam antarmuka atau kelas, dikenal sebagai nersted interface. Ini statis secara default.

Bisakah kelas memiliki interface/antarmuka?

Ya, ini dikenal sebagai nersted interface.

Bisakah Antarmuka memiliki kelas?

Ya, mereka statis secara implisit.

Dapatkah objek yang tidak direfrensikan direfrensikan lagi?

Iya.

Thread jenis apakah yang dimaksud dengan thread Pengumpul Sampah?

Utas daemon.

Apa perbedaan antara final, akhirnya, dan finalisasi?

final: adalah keyword, final dapat berupa variabel, metode, atau kelas. Anda, tidak dapat mengubah nilai variabel akhir, tidak dapat mengganti metode akhir, tidak dapat mewarisi kelas akhir.
finally: finally block digunakan dalam exception handling. finally block selalu dieksekusi. finalize : metode finalize digunakan dalam Garbage Collection.
metode finalize: dipanggil tepat sebelum objek dikumpulkan sampah. Metode finalize. bisa digunakan untuk melakukan pemrosesan pembersihan apa pun.

Apa tujuan dari kelas Runtime?

Tujuan dari kelas Runtime adalah untuk menyediakan akses ke sistem runtime Java.

Bagaimana Anda akan menjalankan proses eksternal di Java?

Dengan metode Runtime.getRuntime(). Exec (?).

Pertanyaan Wawancara I/O

Apa perbedaan antara hierarki kelas Reader/Writer dan hierarki kelas InputStream/OutputStream?

Hierarki kelas Reader/Writer berorientasi pada karakter, dan hierarki kelas InputStream/OutputStream berorientasi pada byte.

Apa itu filter I/O?

Filter I/O adalah objek yang membaca dari satu stream dan menulis ke stream lain, biasanya mengubah data dengan cara tertentu saat diteruskan dari satu stream ke stream lainnya.

Serialisasi Pertanyaan Wawancara

Apa itu serialisasi?

Serialisasi adalah proses menulis status objek ke dalam stream byte, yang terutama digunakan untuk melakukan perjalanan status objek di jaringan.

Apa itu Deserialization?

Deserialization adalah proses merekonstruksi objek dari status serialisasi. Ini adalah operasi kebalikan dari serialisasi.

Apa itu keyword sementara?

Jika Anda mendefinisikan anggota data sebagai sementara, itu tidak akan diserialkan.

Apa itu Dapat Dieksternalisasikan?

Antarmuka yang dapat dieksternalisasi digunakan untuk menulis keadaan objek ke dalam stream byte dalam format terkompresi. Ini bukan antarmuka penanda.

Apa perbedaan antara antarmuka Serializalble dan Externalizable?

Serializable adalah antarmuka penanda tetapi Externalizable bukan antarmuka penanda. Saat Anda menggunakan antarmuka Serializable, kelas Anda akan diserialisasi secara otomatis secara default. Tetapi Anda dapat mengganti dua metode writeObject() dan readObject() untuk mengontrol proses serailisasi objek yang lebih kompleks. Saat Anda menggunakan antarmuka Eksternal, Anda memiliki kendali penuh atas proses serialisasi kelas Anda.

Pertanyaan Wawancara Jaringan

Bagaimana cara mengubah alamat IP numerik seperti 192.18.97.39 menjadi nama host seperti java.sun.com?

Oleh InetAddress.getByName ("192.18.97.39"). GetHostName() di mana 192.18.97.39 adalah alamat IP.

Pertanyaan Wawancara Refleksi

Apakah refleksi itu?

Refleksi adalah proses memeriksa atau memodifikasi perilaku runtime kelas saat runtime. Ini digunakan dalam:

IDE (Integreted Development Environment) mis. Eclipse, MyEclipse, NetBeans.

Debugger

Alat Uji dll.

Bisakah Anda mengakses metode privat dari luar kelas?

Ya, dengan mengubah perilaku runtime kelas jika kelas tersebut tidak diamankan.

BAB 24
JAVA MISCELLANEOUS
PERTANYAAN WAWANCARA AWT DAN SWING

Apa itu wrapperclasses?

Wrapperclasses adalah kelas yang memungkinkan tipe primitif untuk diakses sebagai objek.

Apa metode asli?

Metode asli adalah metode yang diimplementasikan dalam bahasa selain Java.

Apa tujuan system class?

Tujuan system class adalah untuk menyediakan akses ke sumber daya sistem.

Apa yang terlintas dalam pikiran ketika seseorang menyebutkan salinan dangkal di java?

Kloning objek.

Apa itu kelas singleton?

Kelas Singleton berarti bahwa pada waktu tertentu hanya ada satu contoh kelas yang ada, dalam satu JVM.

Pertanyaan Wawancara Awt Dan Swing

Penampung mana yang menggunakan tata letak perbatasan sebagai tata letak defaultnya?

Kelas Window, Frame dan Dialog menggunakan tata letak perbatasan sebagai tata letak default mereka.

Penampung mana yang menggunakan FlowLayout sebagai tata letak defaultnya?

Kelas Panel dan Applet menggunakan FlowLayout sebagai tata letak defaultnya.

Apa itu komponen tak tertandingi?

Komponen tak tertandingi disebut komponen ringan.

perbedaan antara Scrollbar dan ScrollPane?

Scrollbar adalah Komponen, tetapi bukan Penampung. ScrollPane adalah Container. ScrollPane menangani acara sendiri dan melakukan penggulirannya sendiri.

Apa itu komponen ringan?

Komponen ringan adalah komponen yang tidak sesuai dengan panggilan asli untuk mendapatkan unit grafis. Mereka membagikan unit grafis komponen induknya untuk membuatnya. Misalnya komponen Swing.

Apa itu komponen kelas berat?

Untuk setiap panggilan paint, akan ada panggilan asli untuk mendapatkan unit grafis. Misalnya, AWT.

Apa itu applet?

Applet adalah program java kecil yang berjalan di dalam browser dan menghasilkan konten dinamis.

Bisakah Anda menulis kelas Java yang bisa digunakan baik sebagai applet maupun aplikasi?

Iya. Tambahkan metode main() ke applet.

Pertanyaan Wawancara Internasionalisasi

Apa itu Lokal?

Objek Lokal mewakili wilayah geografis, politik, atau budaya tertentu.

Bagaimana Anda akan memuat lokasi tertentu?

Dengan metode ResourceBundle.getBundle (?).

Pertanyaan Wawancara Java Bean

Apa itu JavaBean?

JavaBean adalah komponen perangkat lunak yang dapat digunakan kembali yang ditulis dalam bahasa pemrograman Java, dirancang untuk dimanipulasi secara visual oleh lingkungan pengembangan perangkat lunak, seperti JBuilder atau VisualAge untuk Java.

Pertanyaan Wawancara Rmi

Dapatkah aplikasi berbasis RMI dan Corba berinteraksi?

Ya mereka bisa. RMI tersedia dengan IIOP sebagai protokol transport daripada JRMP.

Pertanyaan Wawancara Multitrheading Java

Multithreading dan Synchronization dianggap sebagai bab khas dalam pemrograman java. Di perusahaan pengembang game, pertanyaan wawancara terkait mulithreading sering ditanyakan. Daftar pertanyaan wawancara multithreading java yang sering diajukan diberikan di bawah ini.

Apa itu multithreading?

Multithreading adalah proses menjalankan beberapa utas secara bersamaan. Keuntungan utamanya adalah:

- Threads share ruang alamat yang sama.
- Thread-nya ringan.
- Biaya komunikasi antar proses rendah.

Apa itu thread/utas?

Utas adalah subproses ringan, jalur eksekusi terpisah, dan disebut jalur eksekusi terpisah karena setiap utas berjalan dalam bingkai tumpukan terpisah.

Apa perbedaan antara penjadwalan preemptive dan pembagian waktu?

Di bawah penjadwalan preemptive, tugas prioritas tertinggi dijalankan hingga memasuki status menunggu atau mati atau tugas dengan prioritas lebih tinggi muncul. Di bawah pembagian waktu, tugas dijalankan untuk bagian waktu yang telah ditentukan dan kemudian memasukkan kembali kumpulan tugas yang siap. Penjadwal kemudian menentukan tugas mana yang harus dijalankan selanjutnya, berdasarkan prioritas dan faktor lainnya.

Apa metode join()?

Metode join() menunggu utas mati. Dengan kata lain, ini menyebabkan utas yang sedang berjalan berhenti dijalankan hingga utas yang bergabung dengannya menyelesaikan tugasnya.

Apa perbedaan antara metode wait() dan sleep()?

Wait	Sleep
Metode wait didefinisikan pada Object class	*1. Metode sleep* diidefinisikan pada Thread class
Metode wait merilis lock	*2. Metode sleep* tidak merilis lock

Apakah mungkin memulai utas dua kali?

Tidak, tidak ada kemungkinan untuk memulai utas dua kali. Jika ya, itu akan membuat pengecualian.

Bisakah kita memanggil metode run() daripada start()?

ya, tetapi ini tidak akan berfungsi sebagai utas melainkan akan berfungsi sebagai objek normal sehingga tidak akan ada peralihan konteks antar utas.

Bagaimana dengan untaian daemon?

Utas daemon pada dasarnya adalah utas prioritas rendah yang menyediakan dukungan latar belakang untuk utas pengguna. Ini menyediakan layanan ke utas pengguna.

Bisakah kita menjadikan utas pengguna sebagai utas daemon jika utas dimulai?

Tidak, jika Anda melakukannya, IllegalThreadStateException akan muncul

Apa itu hook shutdown?

Pengait shutdown pada dasarnya adalah utas yaitu dipanggil secara implisit sebelum JVM dimatikan. Jadi kita bisa menggunakannya untuk melakukan pembersihan sumber daya.

Kapan kita harus menghentikan sebuah utas?

Kita harus menghentikan sebuah utas jika kita ingin menghentikan kondisi tidur atau menunggu utas.

Apa itu sinkronisasi?

Sinkronisasi adalah kemampuan untuk mengontrol akses beberapa utas ke sumber daya bersama.

Untuk mencegah gangguan thread.

Untuk mencegah masalah konsistensi.

Apa tujuan blok sinkron?

Blok tersinkronisasi digunakan untuk mengunci objek untuk sumber daya bersama.

Cakupan blok tersinkronisasi lebih kecil dari metode.

Bisakah objek Java dikunci untuk penggunaan eksklusif oleh utas tertentu?

Iya. Anda dapat mengunci objek dengan meletakkannya di blok "tersinkronisasi". Objek terkunci tidak dapat diakses ke utas apa pun selain utas yang secara eksplisit mengklaimnya.

Apa itu sinkronisasi statis?

Jika Anda membuat metode statis apa pun sebagai disinkronkan, kunci akan berada di kelas, bukan pada objek.

Apa perbedaan antara notify() dan notifyAll()?

Notify() digunakan untuk membuka blokir satu thread menunggu sedangkan metode notifyAll() digunakan untuk membuka blokir semua thread dalam keadaan menunggu.

Apa itu deadlock?

Deadlock adalah situasi ketika dua utas menunggu satu sama lain untuk melepaskan sumber daya. Setiap utas menunggu sumber daya yang dipegang oleh utas menunggu lainnya.

BAB 25
20 PERTANYAAN WAWANCARA JAVA COLLECTION

Di Java, pertanyaan wawancara pengumpulan paling banyak ditanyakan oleh pewawancara. Berikut adalah daftar pertanyaan wawancara koleksi yang paling sering ditanyakan dengan jawaban.

Apa perbedaan antara ArrayList dan Vector?

No.	Array List	Vektor
1	ArrayList tidak disinkronisaikan	Vektor disinkronkan
2	ArrayList bukan legacy class	Vektor adalah legacy class
3	meningkatkan ukurannya sebesar 50% dari ukuran array.	Vektor meningkatkan ukurannya dengan menggandakan ukuran array.

Apa perbedaan antara ArrayList dan LinkedList?

No.	Array List	LinkedList
1	ArrayList meggunakan dynamic array	LinkedList menggunakan dua kali lipat linkedlist
2	ArrayList tidak efisien untuk memanipulasi karena banyak pergeseran diperlukan	LinkedList efisien untuk memanipulasi
3	ArrayList lebih baik menyimpan dan mengambil data.	LinkedList jauh lebihbaik dalam memanipulasi data

Apa perbedaan antara Iterator dan ListIterator?

Iterator melintasi elemen ke arah depan saja sedangkan ListIterator melintasi elemen ke arah depan dan belakang.

No.	Iterator	ListIterator
1	Iterator hanya melintasi elemen ke arah depan.	ListIterator melintasi kedua elemen ke belakang dan ke depan.
2	Iterator dapat digunakan dalam List, Set dan Queue.	ListIteraror hanya dapat digunakan dalam Daftar

Apa perbedaan antara Iterator dan Enumeration?

No.	Iterator	Enumerasi

1	Iterator dapat melintasi elemen legacy dan nonlegacy.	Enumerasi hanya dapat melintasi elemen lama.
2	Iterator cepat gagal.	Enumerasi tidak cepat gagal.
3	Iterator lebih lambat dari Enumerasi.	Enumerasi lebih cepat dari Iterator.

Apa perbedaan antara List dan Set?

Daftar dapat berisi elemen duplikat sedangkan Set hanya berisi elemen unik.

Apa perbedaan antara HashSet dan TreeSet?

HashSet tidak mempertahankan urutan sedangkan TreeSet mempertahankan urutan naik.

Apa perbedaan antara Set dan Map?

Set berisi nilai hanya sedangkan Map berisi kunci dan nilai keduanya.

Apa perbedaan antara HashSet dan HashMap?

HashSet hanya berisi nilai sedangkan HashMap berisi entri (kunci, nilai). HashSet dapat diulangi tetapi HashMap perlu diubah menjadi Set untuk diiterasi.

Apa perbedaan antara HashMap dan TreeMap?

HashMap tidak mempertahankan urutan tetapi TreeMap mempertahankan urutan naik.

Apa perbedaan antara HashMap dan Hashtable?

No.	HaskMap	Hashtable
1	HashMap tidak disinkronkan.	Hashtable disinkronkan.
2	HashMap dapat berisi satu kunci nol dan beberapa nilai nol.	Hashtable tidak boleh berisi kunci nol atau nilai nol.

Apa perbedaan antara Koleksi dan Koleksi?

Koleksi adalah antarmuka sedangkan Koleksi adalah kelas. Antarmuka Koleksi menyediakan fungsionalitas normal dari struktur data ke Daftar, Set, dan Antrean. Tapi, kelas Koleksi adalah untuk mengurutkan dan menyinkronkan elemen koleksi.

Apa perbedaan antara Comparable dan Comparator?

No.	Comparable	Comparator
1	Comparable hanya menyediakan satu jenis urutan.	Comparator menyediakan beberapa jenis urutan.
2	Ini menyediakan satu metode bernama CompareTo.	Ini menyediakan satu metode bernama Compare.
3	Itu ditemukan dalam paket java.lang.	itu ditemukan dalam paket java.util.
4	Jika kita mengimplementasikan	Kelas sebenarnya tidak diubah

	antarmuka Comparable, kelas aktual akan dimodifikasi.	

Apa keuntungan dari file Properties?

Jika Anda mengubah nilai di file properti, Anda tidak perlu mengompilasi ulang kelas java. Jadi, itu membuat aplikasi mudah dikelola.

Apa metode hashCode()?

Metode hashCode() mengembalikan nilai kode hash (bilangan bulat). Metode hashCode() mengembalikan bilangan bulat yang sama, jika dua kunci (dengan memanggil metode equals()) sama. Namun, ada kemungkinan dua nomor kode hash dapat memiliki kunci yang berbeda atau sama.

Mengapa kita mengganti metode equals()?

Metode sama digunakan untuk memeriksa apakah dua objek sama atau tidak. Ini perlu diganti jika kita ingin memeriksa objek berdasarkan properti. Misalnya Employee adalah kelas yang memiliki 3 anggota data: id, nama dan gaji. Tapi, kami ingin memeriksa kesetaraan objek karyawan berdasarkan gaji. Kemudian, kita perlu mengganti metode equals().

Bagaimana cara menyinkronkan elemen List, Set, dan Map?

Ya, kelas Koleksi menyediakan metode untuk membuat elemen Daftar, Set, atau Peta sebagai disinkronkan:

public static List synchronizedList (Daftar l) {}
public static Set synchronizedSet (Set s) {}
public static SortedSet synchronizedSortedSet (SortedSet s) {}
Peta statis publik synchronizedMap (Peta m) {}
SortedMap public static synchronizedSortedMap (SortedMap m) {}

Apa keuntungan dari koleksi generik?

Jika kita menggunakan kelas generik, kita tidak membutuhkan typecasting. Ini aman untuk mengetik dan diperiksa pada waktu kompilasi.

Apa itu tabrakan hash di Hashtable dan bagaimana penanganannya di Java?

Dua kunci berbeda dengan nilai hash yang sama dikenal sebagai tabrakan hash. Dua entri berbeda akan disimpan dalam satu keranjang hash untuk menghindari tabrakan.

Apa itu kelas Kamus?

Kelas Dictionary menyediakan kemampuan untuk menyimpan pasangan nilai-kunci.

Berapa ukuran default faktor beban dalam koleksi berbasis hashing?

Ukuran default faktor beban adalah 0,75. Kapasitas default dihitung sebagai kapasitas awal * faktor beban. Misalnya, 16 * 0,75 = 12. Jadi, 12 adalah kapasitas default Map.

Pertanyaan Wawancara Jdbc

Daftar pertanyaan wawancara jdbc yang sering ditanyakan dengan jawaban diberikan di bawah ini.

Apa itu JDBC?

JDBC adalah Java API yang digunakan untuk menghubungkan dan mengeksekusi query ke database. JDBC API menggunakan driver jdbc untuk terhubung ke database.

Apa itu JDBC Driver?

Driver JDBC adalah komponen software yang memungkinkan aplikasi java untuk berinteraksi dengan database. Ada 4 tipe driver JDBC:

- Bridge driver JDBC-ODBC
- Driver Native-API (sebagian driver java)
- Network Protocol driver (sepenuhnya driver java)
- Thin driver (sepenuhnya driver java)

Apa langkah-langkah untuk terhubung ke database di java?

- Mendaftarkan kelas pengemudi
- Membuat koneksi
- Membuat pernyataan
- Menjalankan kueri
- Menutup koneksi

Apa saja komponen API JDBC?

Paket java.sql berisi antarmuka dan kelas untuk JDBC API.

Interface:

- Connection
- Statement
- PreparedStatement
- ResultSet
- ResultSetMetaData
- DatabaseMetaData
- CallableStatement etc.

Kelas:

- DriverManager
- Blob
- Clob

- Types
- SQLException etc.

Apa pernyataan JDBC?

Ada 3 pernyataan JDBC.

- Statement
- PreparedStatement
- CallableStatement

Apa perbedaan antara antarmuka Statement dan PreparedStatement?

Dalam kasus Pernyataan, kueri dipatuhi setiap kali sedangkan dalam kasus PreparedStatement, kueri dipatuhi hanya sekali. Jadi kinerja PreparedStatement lebih baik dari Statement

Bagaimana kita dapat menjalankan prosedur dan fungsi yang tersimpan?

Dengan menggunakan antarmuka pernyataan Callable, kita dapat menjalankan prosedur dan fungsi.

Apa peran kelas JDBC DriverManager?

Kelas DriverManager mengelola driver terdaftar. Ini dapat digunakan untuk mendaftarkan dan membatalkan pendaftaran driver. Ini menyediakan metode pabrik yang mengembalikan instance Connection.

Apa antarmuka Koneksi JDBC?

Antarmuka koneksi mempertahankan sesi dengan database. Ini dapat digunakan untuk manajemen transaksi. Ini menyediakan metode pabrik yang mengembalikan instance Statement, PreparedStatement, CallableStatement dan DatabaseMetaData.

Apa antarmuka JDBC ResultSet?

Objek ResultSet mewakili baris tabel. Ini dapat digunakan untuk mengubah penunjuk kursor dan mendapatkan informasi dari database.

Apa antarmuka JDBC ResultSetMetaData?

Antarmuka ResultSetMetaData mengembalikan informasi tabel seperti jumlah total kolom, nama kolom, jenis kolom, dll.

Apa antarmuka JDBC DatabaseMetaData?

Antarmuka DatabaseMetaData mengembalikan informasi dari database seperti nama pengguna, nama driver, versi driver, jumlah tabel, jumlah tampilan, dll.

Antarmuka mana yang bertanggung jawab untuk manajemen transaksi di JDBC?

Antarmuka Connection menyediakan metode untuk manajemen transaksi seperti commit(), rollback() dll.

Apa itu pemrosesan batch dan bagaimana melakukan pemrosesan batch di JDBC?

Dengan menggunakan teknik batch processing di JDBC, kita dapat mengeksekusi banyak query. Itu membuat kinerjanya cepat.

Bagaimana cara menyimpan dan mengambil gambar dari database?

Dengan menggunakan antarmuka PreparedStaement, kita dapat menyimpan dan mengambil gambar.

BAB 26
FITUR JAVA 8 UNTUK PENGEMBANG
LAMBDAS, FUNCTIONAL INTERFACE, ARUS API DAN WAKTU

Java 8 dirilis pada 18 Maret 2014, jadi inilah saat yang tepat untuk mencari fitur Java 8 untuk para pengembang. Dan tidak ada materi enogh yang tersedia untuk java 8. Tetapi di sini setelah banyak kerja keras saya menulis Beberapa fitur penting yang diperkenalkan di Java 8 yang saya nantikan adalah:

- *forEach() di antarmuka Iterable.*
- *default dan metode statis di Antarmuka.*
- *Functional Interface dan Lambda expression.*
- *Java Stream API untuk Operasi Data Massal pada Collections.*
- *Java Time API.*
- *Peningkatan Collection API.*
- *Peningkatan Concurrency API.*
- *Perbaikan Java IO.*
- *Perbaikan Core API Miscellaneous.*

Mari kita lihat sekilas fitur-fitur Java 8 ini. Saya akan memberikan beberapa potongan kode untuk pemahaman yang lebih baik, jadi jika Anda ingin menjalankan program di Java 8, Anda harus mengatur lingkungan Java 8 dengan mengikuti langkah-langkah berikut.

Unduh JDK8 dan instal. Instalasi sederhana seperti versi java lainnya. Instalasi JDK diperlukan untuk menulis, mengkompilasi dan menjalankan program di Java.

CATATAN: CURRENT ECLIPS IDE tidak mendukung Java8, jadi Anda harus mengunduhnya dari efxclipse.org Eclipse untuk Java 8. Ada beberapa versi berbeda untuk sistem Mac OS, Windows dan Linux dengan versi stabil, jadi unduh yang terbaru untuk kebanyakan fitur.

Saya baru saja memeriksa hari ini (28-Juli-2014) dan paket Eclipse Kepler 4.3.2 SR2 dapat digunakan untuk Java 8. Anda perlu mendownloadnya terlebih dahulu dan kemudian menginstal plugin "Java 8 support for Eclipse Kepler SR2" dari Eclipse Marketplace. Saya telah mencoba ini dan tampaknya berfungsi dengan baik.

forEach() di antarmuka Iterable.

Setiap kali kita perlu melintasi Koleksi, kita perlu membuat Iterator yang seluruh tujuannya adalah untuk mengulang dan kemudian kita memiliki logika bisnis dalam satu lingkaran untuk masing-masing elemen dalam Collections. Kita mungkin mendapatkan ConcurrentModificationException jika iterator tidak digunakan dengan benar.

Java 8 telah memperkenalkan metode FOREACH di antarmuka java.lang.Iterable sehingga saat menulis kode kami fokus pada logika bisnis saja. Metode FOREACH mengambil objek java.util.function.Consumer sebagai argumen,

sehingga membantu dalam memiliki logika bisnis kita di lokasi terpisah yang dapat kita gunakan kembali. Mari kita lihat penggunaan forEach dengan contoh sederhana.

Java8ForEachExample.java

```
package com.journaldev.java8.foreach;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.function.Consumer;
import java.lang.Integer;
public class Java8ForEachExample {

public static void main(String[] args) {
//creating sample Collection
List<Integer> myList = new ArrayList<Integer>();
for(int i=0; i<10; i++) myList.add(i);

//traversing using Iterator
Iterator<Integer> it = myList.iterator();
while(it.hasNext()){
Integer i = it.next();
System.out.println("Iterator Value::"+i);
}
//melintasi metode forEach dari Iterable dengan anonim class
myList.forEach(new Consumer<Integer>() {

public void accept(Integer t) {
System.out.println("forEach anonymous class Value::"+t);

}
});
//melintasi dengan implementasi antarmuka Konsumen
MyConsumer action = new MyConsumer();
myList.forEach(action);
}
}
//Implementasi konsumen yang dapat digunakan kembali class MyConsumer mengimplementasikan Consumer<Integer>{
public void menerima(Integer t) {
System.out.println("Consumer impl Value::"+t);
}
}
```

Jumlah baris mungkin bertambah tetapi metode forEach membantu memiliki logika untuk iterasi dan logika bisnis di tempat terpisah yang menghasilkan pemisahan perhatian yang lebih tinggi dan kode yang lebih bersih.

default dan metode statis di Antarmuka.

Jika Anda membaca detail metode forEach dengan cermat, Anda akan melihat bahwa metode ini ditentukan dalam antarmuka Iterable, tetapi kami tahu bahwa antarmuka tidak dapat memiliki isi metode. Dari Java 8, antarmuka ditingkatkan untuk memiliki metode dengan implementasi. Kita dapat menggunakan keyword 'DEFAULT' dan 'STATIC' untuk membuat antarmuka dengan implementasi metode.

Untuk setiap implementasi metode di antarmuka Iterable adalah:

```
default void forEach(Consumer<? super T> action) {
Objects.requireNonNull(action);
for (T t : this) {
action.accept(t);
}
}
```

Ingat:

Kami tahu bahwa Java tidak menyediakan banyak inheritance di Kelas karena itu mengarah ke Diamond problem. Jadi bagaimana itu akan ditangani dengan antarmuka sekarang, karena antarmuka sekarang mirip dengan kelas abstrak.

Solusinya adalah kompilator akan membuang pengecualian dalam skenario ini dan kita harus menyediakan logika implementasi di kelas yang mengimplementasikan antarmuka.

Interface Java

```
package com.journaldev.java8.defaultmethod;
@FunctionalInterface
public interface Interface1 {
void method1(String str);
default void log(String str){
System.out.println("I1 logging::"+str);
}
static void print(String str){
 System.out.println("Printing "+str);
}
//mencoba untuk menimpa Metode objek memberikan kesalahan waktu kompilasi sebagai
//"Metode default tidak dapat menggantikan metode dari java.lang.Object"  /
/  default String toString(){
//return "i1";
//  }
}
```

Interface2.java

```
package com.journaldev.java8.defaultmethod;

@FunctionalInterface
public interface Interface2 {
void method2();
default void log(String str){
System.out.println("I2 logging::"+str);
}

}
```

Perhatikan bahwa kedua antarmuka memiliki log metode umum() dengan logika implementasi.

MyClass.Java

```
package com.journaldev.java8.defaultmethod;
public class MyClass mengimplementasikan Interface1, Interface2 {

@Override
public void method2() {
}
@Override
public void method1(String str) {
}
//MyClass tidak akan dapat dikompilasi tanpa lognya sendiri

@Override
public void log(String str){
System.out.println("MyClass logging::"+str);
Interface1.print("abc");
}
}
```

Seperti yang Anda lihat bahwa Interface1 memiliki implementasi metode statis yang digunakan dalam implementasi metode MYCLASS.LOG(). Java 8 banyak menggunakan metode default dan statis dalam Collection API dan metode default ditambahkan sehingga kode kami tetap kompatibel dengan versi sebelumnya.

Jika ada inner class hierarki yang memiliki metode dengan tanda tangan yang sama, maka metode default menjadi tidak relevan. Karena setiap kelas yang mengimplementasikan antarmuka sudah memiliki Object sebagai superclass, jika kita memiliki metode default sama dengan(), hashCode() di antarmuka, itu akan menjadi tidak relevan. Itulah mengapa untuk kejelasan yang lebih baik, antarmuka tidak diizinkan untuk memiliki metode default kelas Object.

Untuk detail lengkap tentang perubahan antarmuka di Java 8, silakan baca Perubahan antarmuka Java 8 Di Bawah Ini.

Perubahan Antarmuka Java 8 - metode statis, metode default, Functional Interface

Salah satu perubahan desain terbesar di Java 8 adalah dengan konsep antarmuka. Sebelum Java 7, kita hanya bisa memiliki deklarasi metode di antarmuka. Tetapi dari Java 8, kita dapat memiliki metode default dan metode statis di antarmuka.

Merancang antarmuka selalu menjadi pekerjaan yang sulit karena jika kita ingin menambahkan metode tambahan di antarmuka, itu akan membutuhkan perubahan di semua kelas yang mengimplementasikan.

Seiring bertambahnya usia antarmuka, jumlah kelas yang mengimplementasikannya mungkin bertambah sehingga tidak mungkin untuk memperluas antarmuka. Itulah mengapa saat mendesain aplikasi, sebagian besar kerangka kerja menyediakan kelas implementasi dasar dan kemudian kami memperluasnya serta mengganti metode yang berlaku untuk aplikasi kami. Mari kita lihat metode antarmuka default dan statis serta alasan pendahuluannya.

Metode Default Antarmuka

Untuk membuat metode default di antarmuka, kita perlu menggunakan keyword "default" dengan tanda tangan metode. Sebagai contoh

Interface1.java

```
package com.journaldev.java8.defaultmethod;
public interface Interface1 {
void method1(String str);

default void log(String str){
System.out.println("I1 logging::"+str);
print(str);
}
}
```

Perhatikan bahwa log (String str) adalah metode default di Interface1. Sekarang ketika kelas akan mengimplementasikan Interface1, itu tidak wajib menyediakan implementasi untuk metode default. Fitur ini akan membantu kami dalam memperluas antarmuka dengan metode tambahan, yang kami butuhkan hanyalah menyediakan implementasi default. Katakanlah kita memiliki antarmuka lain dengan metode berikut:

Interface2.java

```
package com.journaldev.java8.defaultmethod;
public interface Interface2 {
void method2();
default void log(String str){
System.out.println("I2 logging::"+str);
```

```
}
{
```

Kita tahu bahwa Java tidak mengizinkan kita untuk memperluas beberapa kelas karena akan menghasilkan "Diamond Problem" di mana compiler tidak dapat memutuskan metode superclass mana yang akan digunakan. Dengan metode default, diamond problem akan muncul untuk antarmuka juga.

Karena jika sebuah kelas mengimplementasikan Interface1 danInterface2 dan tidak mengimplementasikan metode default umum, compiler tidak dapat memutuskan mana yang akan dipilih.

Memperluas beberapa antarmuka merupakan bagian integral dari Java, Anda akan menemukannya di kelas-kelas core java serta di sebagian besar aplikasi dan kerangka kerja perusahaan.

Jadi untuk memastikan, masalah ini tidak akan terjadi di antarmuka, itu diwajibkan untuk menyediakan implementasi untuk metode default umum. Jadi jika sebuah kelas mengimplementasikan kedua antarmuka di atas, ia harus menyediakan implementasi untuk metode log() jika tidak kompilator akan memunculkan kesalahan. Kelas sederhana yang mengimplementasikan Interface1 dan Interface2 adalah:

MyClass.java

```
package com.journaldev.java8.defaultmethod;
public class MyClass implements Interface1, Interface2 {
@Override
public void method2() {
}
@Override
public void method1(String str) {
}
@Override
public void log(String str){
System.out.println("MyClass logging::"+str);
Interface1.print("abc");
}
}
```

Poin penting tentang metode default antarmuka:

Metode default akan membantu kita dalam memperluas antarmuka tanpa takut merusak kelas implementasi.

Metode default telah menjembatani perbedaan antara antarmuka dan kelas abstrak.

Metode default akan membantu kita menghindari kelas utilitas, seperti semua metode kelas Koleksi dapat disediakan di antarmuka itu sendiri.

Metode default akan membantu kita dalam menghapus kelas implementasi dasar, kita dapat menyediakan implementasi default dan kelas implementasi dapat memilih kelas mana yang akan diganti.

Salah satu alasan utama memperkenalkan metode default adalah untuk menyempurnakan Collections API di Java 8 untuk mendukung Lambda expression.

Jika ada inner class hierarki yang memiliki metode dengan tanda tangan yang sama, maka metode default menjadi tidak relevan. Metode default tidak dapat menggantikan metode dari java.lang.Object. Alasannya sangat sederhana, karena Object adalah kelas dasar untuk semua kelas java. Jadi bahkan jika kita memiliki metode kelas Objek yang didefinisikan sebagai metode default dalam antarmuka, itu tidak akan berguna karena metode kelas Objek akan selalu digunakan. Itulah mengapa untuk menghindari kebingungan, kami tidak dapat memiliki metode default yang menggantikan metode kelas Object.

Metode default juga disebut sebagai Metode Pembela atau metode ekstensi Virtual.

Metode Interface Statis

Metode statis mirip dengan metode default kecuali kita tidak dapat menimpanya di kelas implementasi. Fitur ini membantu kami dalam menghindari hasil yang tidak diinginkan jika terjadi implementasi yang buruk di kelas anak. Mari kita lihat ini dengan contoh sederhana.

MyData.java

```
package com.journaldev.java8.staticmethod;
public interface MyData {
default void print(String str) {
if (!isNull(str))
System.out.println("MyData Print::" + str);
}
static boolean isNull(String str) {
System.out.println("Interface Null Check");
 return str == null ? true : "".equals(str) ? true : false;
}
}
```

Sekarang mari kita lihat kelas implementasi yang memiliki metode isNull() dengan implementasi yang buruk.

MyDataImpl.java

```
package com.journaldev.java8.staticmethod;
public class MyDataImpl implements MyData {
public boolean isNull(String str) {
System.out.println("Impl Null Check");
return str == null ? true : false;
}
```

```
public static void main(String args[]){
MyDataImpl obj = new MyDataImpl();
obj.print("");
obj.isNull("abc");
}
}
```

Perhatikan bahwa isNull (String str) adalah metode kelas sederhana, ini tidak menimpa metode antarmuka.

Misalnya, jika kita akan menambahkan anotasi @Override ke metode isNull(), itu akan menghasilkan kesalahan compiler. Sekarang ketika kita akan menjalankan aplikasi, kita mendapatkan keluaran sebagai berikut.

```
Interface Null Check
Impl Null Check
```

Jika kita membuat metode antarmuka dari statis menjadi default, kita akan mendapatkan output berikut.

```
Impl Null Check
MyData Print::
Impl Null Check
```

Metode statis hanya terlihat oleh metode antarmuka, jika kita menghapus metode isNull() dari kelasMyDataImpl, kita tidak akan dapat menggunakannya untuk objek MyDataImpl. Namun seperti metode statis lainnya, kita dapat menggunakan metode statis antarmuka menggunakan nama kelas. Misalnya, pernyataan yang valid adalah:

```
boolean result = MyData.isNull("abc");
```

Poin penting tentang metode statis antarmuka:

Metode statis antarmuka adalah bagian dari antarmuka, kami tidak dapat menggunakannya untuk objek kelas implementasi.

Metode statis antarmuka bagus untuk menyediakan metode utilitas, misalnya pemeriksaan nol, penyortiran koleksi, dll. Metode statis antarmuka membantu kita dalam memberikan keamanan dengan tidak mengizinkan kelas implementasi untuk menimpanya.

Kami tidak dapat mendefinisikan metode statis untuk metode kelas Objek, kami akan mendapatkan kesalahan kompiler sebagai "Metode statis ini tidak dapat menyembunyikan metode instance dari Objek". Ini karena tidak diizinkan di java,

karena Object adalah kelas dasar untuk semua kelas dan kita tidak dapat memiliki satu metode statis tingkat kelas dan metode instance lain dengan tanda tangan yang sama.

Kita dapat menggunakan metode antarmuka statis untuk menghapus kelas utilitas seperti Koleksi dan memindahkan semua metode statisnya ke antarmuka yang sesuai, yang akan mudah ditemukan dan digunakan.

Functional Interface

Sebelum saya menyimpulkan topik, saya ingin memberikan pengantar singkat tentang Functional Interface. Antarmuka dengan hanya satu metode abstrak dikenal sebagai Functional Interface.

Anotasi baru @FunctionalInterface telah diperkenalkan untuk menandai sebuah antarmuka sebagai Functional Interface. Anotasi @FunctionalInterface adalah fasilitas untuk menghindari penambahan metode abstrak yang tidak disengaja dalam Functional Interface. Ini opsional tetapi praktik yang baik untuk menggunakannya.

Functional Interface sudah lama ditunggu dan banyak dicari fitur Java 8 karena memungkinkan kita menggunakan Lambda expression untuk membuat instance-nya. Paket baru java.util.function dengan banyak Functional Interface ditambahkan untuk menyediakan tipe target untuk Lambda expression dan referensi metode.

Functional Interface dan Lambda expression.

Jika Anda melihat kode antarmuka di atas, Anda akan melihat anotasi @FunctionalInterface. Functional interface adalah konsep baru yang diperkenalkan di Java 8. Antarmuka dengan satu metode abstrak menjadi Functional interface.

Kita tidak perlu menggunakan anotasi @FunctionalInterface untuk menandai antarmuka sebagai Functional interface. Anotasi @FunctionalInterface adalah fasilitas untuk menghindari penambahan metode abstrak yang tidak disengaja dalam functional interface.

Anggap saja seperti anotasi @Override dan itu praktik terbaik untuk menggunakannya. java.lang.Runnablewith metode abstrak tunggal run() adalah contoh yang bagus dari functional interface.

Salah satu manfaat utama dari functional interface adalah kemungkinan untuk menggunakan Lambda expression untuk membuat instance-nya. Kita dapat membuat contoh antarmuka dengan kelas anonim tetapi kodenya terlihat besar.

```
Runnable r = new Runnable(){
@Override
public void run() {
System.out.println("My Runnable");
}};
```

Karena functional interface hanya memiliki satu metode, Lambda expression dapat dengan mudah menyediakan implementasi metode.

Kami hanya perlu memberikan argumen metode dan logika bisnis. Sebagai contoh, kita dapat menulis implementasi di atas menggunakan Lambda expression sebagai:

```
Runnable r1 =() -> {
System.out.println("My Runnable");
};
```

Jika Anda memiliki pernyataan tunggal dalam penerapan metode, kami juga tidak membutuhkan tanda kurung kurawal. Misalnya di atas kelas anonim Interface1 dapat dibuat menggunakan lambda sebagai berikut:

```
Interface1 i1 = (s) -> System.out.println(s);
i1.method1("abc");
```

Jadi Lambda expression adalah sarana untuk membuat kelas functional interface tanpa nama dengan mudah. Tidak ada manfaat runtime menggunakan Lambda expression, jadi saya akan menggunakannya dengan hati-hati karena saya tidak keberatan menulis beberapa baris kode tambahan.

Paket baru java.util.function telah ditambahkan dengan banyak functional interface untuk menyediakan tipe target untuk Lambda expression dan referensi metode.

Lambda expression adalah topik yang sangat besar, saya akan menulis + (Catatan) terpisah tentang itu di masa mendatang (Edisi Berikutnya Buku ini). Anda dapat membaca tutorial lengkap di bawah Tutorial Ekspresi Java 8 Lambda.

Java 8 Lambda Expressions dan Tutorial Functional interface

Java selalu menjadi bahasa Pemrograman Berorientasi Objek. Apa artinya segala sesuatu dalam pemrograman java berputar di sekitar Objek (kecuali beberapa tipe primitif untuk kesederhanaan). Kami tidak hanya memiliki fungsi di java, mereka adalah bagian dari Class dan kami perlu menggunakan class/object untuk menjalankan fungsi apa pun.

Jika kita melihat ke beberapa bahasa pemrograman lain seperti C++, JavaScript; mereka disebut bahasa pemrograman fungsional karena kita dapat menulis fungsi dan menggunakannya bila diperlukan. Beberapa bahasa ini mendukung Pemrograman Berorientasi Objek serta Pemrograman Fungsional.

Berorientasi objek tidak buruk, tetapi membawa banyak verbositas ke program. Misalnya, kita harus membuat instance Runnable. Biasanya kami melakukannya dengan menggunakan kelas anonim seperti di bawah ini.

```
Runnable r = new Runnable(){
@Override
public void run() {
System.out.println("My Runnable");
}
};
```

Jika Anda melihat kode di atas, bagian sebenarnya yang digunakan adalah kode di dalam metode run(). Sisa semua kode ini karena cara program java terstruktur.

Java 8 memberi kita konsep Fungsional Antarmuka dan Lambda expression untuk menghindari penulisan semua kode tidak berguna yang dapat dengan mudah kita hindari dengan membuat kompiler java kita cerdas.

Functional interface

Antarmuka dengan satu metode abstrak disebut Functional interface. @FunctionalInterfaceannotation ditambahkan sehingga kita dapat menandai sebuah antarmuka sebagai functional interface. Tidak wajib menggunakannya, tetapi praktik terbaiknya adalah menggunakannya dengan functional interface untuk menghindari penambahan metode tambahan secara tidak sengaja. Jika antarmuka dianotasi dengan anotasi @FunctionalInterface dan kami mencoba untuk memiliki lebih dari satu metode abstrak, itu akan memunculkan kesalahan kompiler.

Manfaat utama dari functional interface adalah kita dapat menggunakan Lambda expression untuk membuat instance-nya dan menghindari penggunaan implementasi kelas anonim yang besar.

Java 8 Collections API telah ditulis ulang dan Stream API baru disediakan yang menggunakan banyak functional interface. Java 8 telah mendefinisikan banyak functional interface dalam paket java.util.function, beberapa yang berguna adalah Konsumen, Pemasok, Fungsi, dan Predikat. Anda dapat menemukan lebih banyak detail tentang mereka di Contoh Streaming Java 8.

java.lang.Runnable adalah contoh yang bagus dari functional interface dengan run() metode abstrak tunggal.

Potongan kode di bawah ini memberikan beberapa panduan untuk functional interface:

```
interface Foo { boolean equals(Object obj); }
// Tidak berfungsi karena sama dengan sudah menjadi anggota implisit (Object class)
interface Comparator<T> {
boolean equals(Object obj);
int compare(T o1, T o2); }
// Fungsional karena Pembanding hanya memiliki satu antarmuka metode non-Objek abstrak Foo {  int m();
Object clone();
}
```

```
        // Tidak berfungsi karena metode Object.clone bukan antarmuka publik X {
        int m(Iterable<String> arg);
        }
        interface Y { int m(Iterable<String> arg);
        }
        interface Z extends X, Y {}
        // Fungsional: dua metode, tetapi keduanya memiliki antarmuka tanda tangan
yang sama X {
        Iterable m(Iterable<String> arg); }
        interface Y { Iterable<String> m(Iterable arg); }
        interface Z extends X, Y {}
         // Functional: Y.m is a subsignature & return-type-substitutable
        interface X { int m(Iterable<String> arg); }
        interface Y { int m(Iterable<Integer> arg); }
        interface Z extends X, Y {}
        // Tidak berfungsi: Tidak ada metode yang memiliki subsignature dari semua
antarmuka metode abstrak X {
        int m(Iterable<String> arg, Class c); }
        interface Y {
        int m(Iterable arg, Class<?> c); }
        interface Z extends X, Y {}
        // Tidak berfungsi: Tidak ada metode yang memiliki subsignature dari semua
antarmuka metode abstrak X { long m(); }
        interface Y { int m(); }
        interface Z extends X, Y {}
        // Kesalahan penyusun: tidak ada metode yang mengembalikan jenis Interface
yang dapat diganti Foo<T> {
        void m(T arg); }
        interface Bar<T> { void m(T arg); }
        interface FooBar<X, Y> extends Foo<X>, Bar<Y> {}
        // Kesalahan penyusun: tanda tangan berbeda, penghapusan yang sama
```

Lambda expression

Lambda expression adalah cara kita dapat memvisualisasikan pemrograman fungsional di dunia berorientasi objek java. Objek adalah dasar bahasa pemrograman java dan kami tidak akan pernah memiliki fungsi tanpa Objek, itulah sebabnya bahasa Java memberikan dukungan untuk menggunakan Lambda expression hanya dengan functional interface.

Karena hanya ada satu fungsi abstrak di functional interface, tidak ada kebingungan dalam menerapkan Lambda expression ke metode. Sintaks Lambda Expressions adalah (argumen) -> (body). Sekarang mari kita lihat bagaimana kita dapat menulis di atas Runnable anonim menggunakan Lambda expression.

```
        Runnable r1 =() -> System.out.println("My Runnable");
```

Mari kita coba memahami apa yang terjadi pada Lambda expression di atas.

Runnable adalah functional interface, itulah mengapa kita dapat menggunakan Lambda expression untuk membuat instance-nya.

Karena metode run() tidak membutuhkan argumen, Lambda expression kita juga tidak memiliki argumen. Sama seperti blok if-else, kita bisa menghindari tanda kurung kurawal ({}) karena kita memiliki satu pernyataan dalam tubuh metode.

Untuk beberapa pernyataan, kita harus menggunakan tanda kurung kurawal seperti metode lainnya.

Mengapa kita membutuhkan Lambda expression

Baris Kode yang Dikurangi -

Salah satu manfaat yang jelas dari menggunakan Lambda expression adalah bahwa jumlah kode berkurang, kita telah melihat betapa mudahnya kita dapat membuat instance dari functional interface menggunakan Lambda expression daripada menggunakan kelas anonim.

Dukungan Eksekusi Berurutan dan Paralel -

Manfaat lain menggunakan Lambda expression adalah kita dapat memanfaatkan dukungan operasi sekuensial dan paralel API Stream. Untuk menjelaskan ini, mari kita ambil contoh sederhana di mana kita perlu menulis metode untuk menguji apakah bilangan yang diteruskan adalah bilangan prima atau bukan. Biasanya kami akan menulis kode seperti di bawah ini. Kode tidak sepenuhnya dioptimalkan tetapi bagus untuk tujuan contoh, jadi bersabarlah dengan saya tentang ini.

```
//Traditional approach
private static boolean isPrime(int number) {
if(number < 2) return false;
for(int i=2; i<number; i++){
if(number % i == 0) return false;
}
return true;
}
```

Masalah dengan kode di atas adalah sifatnya yang berurutan, jika angkanya sangat besar maka akan memakan waktu yang cukup lama. Masalah lain dengan kode adalah ada begitu banyak titik keluar dan tidak dapat dibaca. Mari kita lihat bagaimana kita dapat menulis metode yang sama menggunakan Lambda expression dan stream API.

```
//Declarative approach
private static boolean isPrime(int number) {
return number > 1
&& IntStream.range(2, number - 1).noneMatch(
```

IntStream adalah urutan elemen bernilai int primitif yang mendukung operasi agregat sekuensial dan paralel. Ini adalah spesialisasi primitif dari Stream. Agar lebih mudah dibaca, kita juga bisa menulis metode seperti di bawah ini.

```
private static boolean isPrime(int number) {
IntPredicate isDivisible = index -> number % index == 0;
return number > 1
&& IntStream.range(2, number - 1).noneMatch(
isDivisible);
}
```

Jika Anda tidak terbiasa dengan IntStream, metode range() mengembalikan IntStream yang diurutkan secara berurutan dari startInclusive (inklusif) ke endExclusive (eksklusif) dengan langkah tambahan 1. metode noneMatch() mengembalikan apakah tidak ada elemen stream ini yang cocok dengan predikat yang diberikan .

Ini mungkin tidak mengevaluasi predikat pada semua elemen jika tidak perlu untuk menentukan hasil.

Meneruskan Perilaku ke dalam metode

Mari kita lihat bagaimana kita bisa menggunakan Lambda expression untuk meneruskan perilaku metode dengan contoh sederhana. Katakanlah kita harus menulis metode untuk menjumlahkan angka-angka dalam daftar jika cocok dengan kriteria yang diberikan. Kita dapat menggunakan Predikat dan menulis metode seperti di bawah ini.

```
public static int sumWithCondition(List<Integer> numbers,
Predicate<Integer> predicate) {
return numbers.parallelStream()
.filter(predicate)
.mapToInt(i -> i)
.sum();
}
```

Sampel yang digunakan:

```
//sum of all numbers
sumWithCondition(numbers, n -> true)
//sum of all even numbers sumWithCondition(numbers, i -> i%2==0)
//sum of all numbers greater than 5
sumWithCondition(numbers, i -> i>5)
```

Higher Efficiency dengan Laziness

Satu lagi keuntungan menggunakan Lambda expression adalah evaluasi malas, misalnya kita perlu menulis metode untuk menemukan bilangan ganjil maksimum dalam rentang 3 sampai 11 dan mengembalikan kuadratnya. Biasanya kami akan menulis kode untuk metode ini seperti ini:

```
private static int findSquareOfMaxOdd(List<Integer> numbers) {
int max = 0;
for (int i : numbers) {
if (i % 2 != 0 && i > 3 && i < 11 && i > max) {
max = i;
}
}
return max * max;
}
```

Program di atas akan selalu berjalan secara berurutan tetapi kita dapat menggunakan Stream API untuk mencapai ini dan mendapatkan keuntungan dari pencarian Kemalasan. Mari kita lihat bagaimana kita dapat menulis ulang kode ini dengan cara pemrograman fungsional menggunakan Stream API dan Lambda expression.

```
public static int findSquareOfMaxOdd(List<Integer> numbers) {
return numbers.stream()
.filter(NumberTest::isOdd)
//Predicate is functional interface and
.filter(NumberTest::isGreaterThan3)
//we are using lambdas to initialize it
.filter(NumberTest::isLessThan11)
//rather than anonymous inner classes
.max(Comparator.naturalOrder())
.map(i -> i * i)
.get();
}

public static boolean isOdd(int i) {
return i % 2 != 0;
}

public static boolean isGreaterThan3(int i){
return i > 3;
}

public static boolean isLessThan11(int i){
return i < 11;
}
```

Jika Anda terkejut dengan operator titik dua ganda (: :), ini diperkenalkan di Java 8 dan digunakan untuk referensi metode. Java Compiler menangani pemetaan argumen ke metode yang dipanggil. Ini adalah bentuk singkat dari Lambda expression i -> isGreaterThan3 (i) atau i -> NumberTest.isGreaterThan3 (i).

Contoh Lambda expression

Di bawah ini saya memberikan beberapa cuplikan kode untuk Lambda expression dengan komentar kecil yang menjelaskannya.

```
() -> {}  // Tanpa parameter; void result
() -> 42 // Tanpa parameter, expression body
() -> null          // Tanpa parameter, expression body
() -> { return 42; }        // Tanpa parameter, block body with return
() -> { System.gc(); }     // Tanpa parameter, void block body
// Badan blok yang kompleks dengan beberapa hasil
() -> {
 if (true) return 10;
else {
int result = 15;
for (int i = 1; i < 10; i++)
result *= i;
return result;
}
}
(int x) -> x+1    // Argumen tipe deklarasi tunggal
(int x) -> { return x+1; } // sama seperti di atas
(x) -> x+1          // Argumen tipe kesimpulan tunggal, sama seperti di bawah ini
x -> x+1            // Tanda kurung opsional untuk kasus tipe tereka tunggal
(String s) -> s.length() // Argumen tipe deklarasi tunggal
(Thread t) -> { t.start(); } // Argumen argumen tipe tunggal yang dideklarasikan
s -> s.length()  // Argumen tipe kesimpulan tunggal
t -> { t.start(); }          // Argumen tipe kesimpulan tunggal
(int x, int y) -> x+y       // Beberapa parameter tipe yang dideklarasikan
(x,y) -> x+y      // Beberapa parameter tipe-tereka
(x, final y) -> x+y         // Illegal: tidak dapat mengubah parameter tipe-tereka
(x, int y) -> x+y               // Illegal: tidak dapat mencampur tipe yang disimpulkan dan dideklarasikan
```

3 4 5 6 7	

Referensi Metode dan Pembuat

Referensi metode digunakan untuk merujuk ke metode tanpa memanggilnya; referensi konstruktor juga digunakan untuk merujuk ke konstruktor tanpa membuat instance baru dari kelas bernama atau tipe array.

Contoh referensi metode dan konstruktor:

	System::getProperty System.out::println "abc"::length ArrayList::new int[]::new

Sekian untuk Tutorial Functional interface dan Lambda expression, saya sangat menyarankan untuk menggunakannya karena sintaks ini baru untuk Java dan akan membutuhkan waktu untuk memahami dan menggunakannya dengan cara yang lebih baik.

4. Java Stream API untuk Data Massal

Java.util.stream baru telah ditambahkan di Java 8 untuk melakukan filter/map/reduce seperti operasi dengan koleksi. Stream API akan memungkinkan eksekusi sekuensial dan paralel.

Ini adalah salah satu fitur terbaik bagi saya karena saya banyak bekerja dengan Koleksi dan biasanya dengan Big Data, kami perlu memfilternya berdasarkan beberapa kondisi. Antarmuka koleksi telah diperpanjang dengan metode default STREAM() dan PARALLELSTREAM() untuk mendapatkan Stream untuk eksekusi sekuensial dan paralel. Mari kita lihat penggunaannya dengan contoh sederhana.

StreamExample.java
package com.journaldev.java8.stream; import java.util.ArrayList; import java.util.List; import java.util.stream.Stream;

```
public class StreamExample {
public static void main(String[] args) {

List<Integer> myList = new ArrayList<>();
for(int i=0; i<100; i++) myList.add(i);
//sequential stream
Stream<Integer> sequentialStream = myList.stream();

//parallel stream
Stream<Integer> parallelStream = myList.parallelStream();

//using lambda with Stream API, filter example
Stream<Integer> highNums = parallelStream.filter(p -> p > 90);
//using lambda in forEach
highNums.forEach(p -> System.out.println("High Nums parallel="+p));

Stream<Integer> highNumsSeq = sequentialStream.filter(p -> p > 90);
highNumsSeq.forEach(p -> System.out.println("High Nums sequential="+p));
}
}
```

Jika Anda akan menjalankan kode contoh di atas, Anda akan mendapatkan output seperti ini:

```
Tinggi Angka Paralel = 91
Tinggi Angka Paralel = 96
Tinggi Angka paralel = 93
Tinggi Angka paralel = 98
Tinggi Angka paralel = 94
Tinggi Angka paralel = 95
Tinggi angka paralel = 97
Tinggi Tinggi paralel = 92
Tinggi Angka paralel = 99
Sekuensial Angka Tinggi = 91
Sekuensial Angka Tinggi = 92
Sekuensial Angka Tinggi = 93
Sekuensial Angka Tinggi = 94
Sekuensial Angka Tinggi = 95
Sekuensial Angka Tinggi = 96
Sekuensial Angka Tinggi = 97
Sekuensial Angka Tinggi = 98
Sekuensial Angka Tinggi = 99
```

Perhatikan bahwa nilai pemrosesan paralel tidak berurutan, jadi pemrosesan paralel akan sangat membantu saat bekerja dengan koleksi besar. Topik ini tidak mencakup semua hal tentang Stream API, Anda dapat membaca semua tentang Stream API di Tutorial Contoh API Stream 8 Java.

Tutorial Contoh Java 8 Stream API Pada topik terakhir, kita melihat Perubahan Antarmuka Java 8 dan Functional interface serta Lambda expression.

Sekarang kita akan melihat salah satu API utama yang diperkenalkan di Java 8 - Java Stream API.

Ringkasan API Streaming.

Koleksi dan Stream.

Functional interface yang umum digunakan di Stream.

Fungsi dan BiFungsi.

Predikat dan BiPredicate.

Konsumen dan BiConsumer.

Pemasok.

java.util.Optional.

java.util.Spliterator.

Operasi Menengah dan Terminal.

Operasi Sirkuit Pendek.

Contoh Streaming Java.

Membuat Stream.

Mengonversi Stream ke Koleksi atau Larik.

Operasi Menengah Streaming.

Operasi Terminal Streaming.

Batasan Java Stream API.

Ringkasan API Streaming

Sebelum kita melihat Contoh Java 8 Stream API, mari kita lihat mengapa hal itu diperlukan. Misalkan kita ingin mengulang daftar bilangan bulat dan mencari tahu jumlah semua bilangan bulat yang lebih besar dari 10. Sebelum Java 8, pendekatan untuk melakukannya adalah:

```
0
        private static int sumIterator(List<Integer> list) {
        Iterator<Integer> it = list.iterator();
        int sum = 0;
        while (it.hasNext()) {
        int num = it.next();
        if (num > 10) {
        sum += num;
        }
        }
        return sum;
        }
```

1	

Ada tiga masalah utama dengan pendekatan di atas:

Kami hanya ingin mengetahui jumlah bilangan bulat tetapi kami juga harus memberikan bagaimana iterasi akan berlangsung, ini juga disebut iterasi eksternal karena program klien menangani algoritme untuk mengulangi daftar.

Program ini bersifat sekuensial, tidak mungkin kita dapat melakukan ini secara paralel dengan mudah.

Ada banyak kode untuk dilakukan bahkan untuk tugas yang sederhana.

Untuk mengatasi semua kekurangan di atas, Java 8 memperkenalkan Stream API. Kita bisa menggunakan API Stream untuk mengimplementasikan iterasi internal, itu lebih baik karena kerangka kerja java mengendalikan iterasi. Iterasi internal menyediakan beberapa fitur seperti eksekusi sekuensial dan paralel, pemfilteran berdasarkan kriteria yang diberikan, pemetaan, dll.

Sebagian besar argumen metode API Stream adalah functional interface, jadi Lambda expression bekerja sangat baik dengannya. Mari kita lihat bagaimana kita bisa menulis logika di atas dalam satu baris pernyataan.

```
private static int sumStream(List<Integer> list) {
return list.stream().filter(i -> i > 10).mapToInt(i -> i).sum();
}
```

Perhatikan bahwa program di atas menggunakan strategi iterasi kerangka kerja java, metode pemfilteran dan pemetaan dan akan meningkatkan efisiensi. Pertama-tama kita akan melihat konsep inti dari Stream API dan kemudian kita akan melihat beberapa contoh untuk memahami metode yang paling umum digunakan.

Koleksi dan Stream

Collection adalah struktur data dalam memori untuk menyimpan nilai dan sebelum kita mulai menggunakan collection, semua nilai seharusnya sudah diisi. Sedangkan Stream adalah struktur data yang dihitung sesuai permintaan.

Stream tidak menyimpan data, ia beroperasi pada struktur data sumber (kumpulan dan larik) dan menghasilkan data pipeline yang dapat kita gunakan dan melakukan operasi tertentu. Seperti kita dapat membuat stream dari daftar dan memfilternya berdasarkan suatu kondisi.

Operasi Stream menggunakan functional interface, yang membuatnya sangat cocok untuk pemrograman fungsional menggunakan Lambda expression. Seperti yang Anda lihat pada contoh di atas bahwa menggunakan Lambda expression membuat kode kita dapat dibaca dan dipendekkan.

Prinsip iterasi internal streaming membantu mencapai pencarian malas di beberapa operasi streaming. Misalnya pemfilteran, pemetaan, atau penghapusan

duplikat dapat diterapkan dengan malas, memungkinkan kinerja dan cakupan yang lebih tinggi untuk pengoptimalan.

Stream dapat dikonsumsi, jadi tidak ada cara untuk membuat referensi ke stream untuk penggunaan di masa mendatang. Karena datanya sesuai permintaan, streaming yang sama tidak dapat digunakan kembali beberapa kali.

Dukungan stream berurutan serta pemrosesan paralel, pemrosesan paralel dapat sangat membantu dalam mencapai kinerja tinggi untuk koleksi besar.

Semua antarmuka dan kelas API Stream ada dalam paket java.util.stream. Karena kita bisa menggunakan tipe data primitif seperti int, lama dalam koleksi menggunakan autoboxing dan operasi ini bisa memakan banyak waktu, ada kelas khusus untuk ini - IntStream, LongStream, dan DoubleStream.

Functional interface yang umum digunakan di Stream

Beberapa functional interface yang umum digunakan dalam metode Stream API adalah:

Function dan BiFunction:

Fungction mewakili fungsi yang mengambil satu tipe argumen dan mengembalikan tipe argumen lain. Fungsi adalah bentuk umum di mana T adalah jenis masukan untuk fungsi dan R adalah jenis hasil dari fungsi tersebut. Untuk menangani tipe primitif, ada antarmuka Fungsi tertentu

ToIntFunction, ToLongFunction, ToDoubleFunction, ToIntBiFunction, ToLongBiFunction, ToDoubleBiFunction, LongToIntFunction, LongToDoubleFunction, IntToLongFunction, IntToDoubleFunctionetc.

Beberapa metode Stream yang menggunakan Function atau spesialisasi primitifnya adalah:

Peta <R> Stream <R> (Fungsi <? Super T,? Extends R> mapper)

IntStream mapToInt (ToIntFunction <? Super T> mapper) - juga untuk stream spesifik primitif yang panjang dan ganda.

IntStream flatMapToInt (Function <? Super T,? Extends IntStream> mapper) - sama untuk long dan double

<A> A [] toArray (IntFunction <A []> generator)

<U> U kurangi (identitas U, BiFunction <U,? Super T, U> akumulator, penggabung BinaryOperator <U>)

Predikat dan BiPredicate:

Ini mewakili predikat terhadap elemen stream mana yang diuji. Ini digunakan untuk menyaring elemen dari stream. Sama seperti Function, ada antarmuka spesifik primitif untuk int, long dan double. Beberapa metode Stream- di mana spesialisasi Predicate atau BiPredicate digunakan adalah:

Stream<T> filter(Predicate<? super T> predicate)

boolean anyMatch(Predicate<? super T> predicate)

boolean allMatch(Predicate<? super T> predicate)

boolean noneMatch(Predicate<? super T> predicate)

Consumer dan BiConsumer:

Ini mewakili operasi yang menerima argumen input tunggal dan tidak mengembalikan hasil. Ini dapat digunakan untuk melakukan beberapa tindakan pada semua elemen stream. Beberapa metode Stream di mana Consumer, BiConsumer, atau antarmuka spesialisasi primitifnya digunakan adalah:

Stream<T> peek(Consumer<? super T> action)

void forEach(Consumer<? super T> action)

void forEachOrdered(Consumer<? super T> action)

Pemasok:

Pemasok mewakili suatu operasi di mana kami dapat menghasilkan nilai-nilai baru di stream. Beberapa metode di Stream yang membutuhkan argumen Pemasok adalah:

public static<T> Stream<T> generate(Supplier<T> s)

<R>Rcollect(Supplier<R>supplier,BiConsumer<R ? super T> accumulator,BiConsumer<R, R> combiner)

java.util.Optional

Opsional adalah objek kontainer yang mungkin berisi atau tidak mengandung nilai non-null. Jika sebuah nilai ada, isPresent() akan mengembalikan true dan get() akan mengembalikan nilainya. Operasi terminal stream mengembalikan objek opsional. Beberapa dari metode ini adalah:

Optional<T> reduce(BinaryOperator<T> accumulator)

Optional<T> min(Comparator<? super T> comparator)

Optional<T> max(Comparator<? super T> comparator)

Optional<T> findFirst() Optional<T> findAny()

java.util.Spliterator

Untuk mendukung eksekusi paralel di Stream API, antarmuka Spliterator digunakan. Metode spliterator trySplit mengembalikan Spliterator baru yang mengelola subset elemen Spliterator asli.

Operasi Menengah dan Terminal

Operasi Stream API yang mengembalikan Stream baru disebut operasi perantara. Seringkali, operasi ini bersifat malas, jadi mereka mulai memproduksi elemen stream baru dan mengirimkannya ke operasi berikutnya. Operasi menengah tidak pernah menjadi operasi produksi hasil akhir.

Operasi menengah yang umum digunakan adalah operasi filter dan map.Stream API yang mengembalikan hasil atau menghasilkan efek samping. Setelah metode terminal dipanggil pada stream, itu menghabiskan stream dan setelah itu kita tidak dapat menggunakan stream. Operasi terminal sifatnya bersemangat yaitu mereka memproses semua elemen di stream sebelum mengembalikan hasilnya.

Metode terminal yang umum digunakan adalah forEach, toArray, min, max, findFirst, anyMatch, allMatch, dll. Anda dapat mengidentifikasi metode terminal dari tipe kembalian, metode tersebut tidak akan pernah mengembalikan Stream.

Operasi Sirkuit Pendek

Operasi perantara disebut hubung singkat, jika dapat menghasilkan stream hingga untuk stream tak hingga. Misalnya limit() dan skip() adalah dua operasi perantara hubung singkat.

Operasi terminal disebut hubung singkat, jika dapat berakhir dalam waktu yang terbatas untuk stream tak terbatas. Misalnya anyMatch, allMatch, noneMatch, findFirst, dan findAny adalah operasi terminal hubung singkat.

Contoh Streaming Java

Saya telah membahas hampir semua bagian penting dari Java Stream API. Sangat menarik untuk menggunakan fitur API baru ini dan mari kita lihat beraksi dengan beberapa contoh.

Membuat Stream

Ada beberapa cara untuk membuat Stream dari array dan koleksi. Mari kita lihat ini dengan contoh sederhana.

Kita dapat menggunakan Stream.of() untuk membuat stream dari jenis data yang serupa. Misalnya, kita dapat membuat Stream dari integer dari sekelompok objek int atau Integer.

```
Stream<Integer> stream = Stream.of(1,2,3,4);
```

Kita bisa menggunakan Stream.of() dengan array Objects untuk mengembalikan stream. Perhatikan bahwa ini tidak mendukung autoboxing, jadi kami tidak dapat meneruskan larik tipe primitif.

```
Stream<Integer> stream = Stream.of(new Integer[]{1,2,3,4});
//works fine

Stream<Integer> stream1 = Stream.of(new int[]{1,2,3,4});
//Compile time error, Type mismatch: cannot convert from Stream<int[]> to Stream<Integer>
```

Kita bisa menggunakan Collection stream() untuk membuat stream sekuensial dan parallelStream() untuk membuat stream paralel.

```
List<Integer> myList = new ArrayList<>();
for(int i=0; i<100; i++) myList.add(i);
//sequential stream
Stream<Integer> sequentialStream = myList.stream();
//parallel stream
Stream<Integer> parallelStream = myList.parallelStream()
```

Kita bisa menggunakan metode Stream.generate() dan Stream.iterate() untuk membuat Stream.

```
Stream<String> stream1 = Stream.generate(() -> {return "abc";});
Stream<String> stream2 = Stream.iterate("abc", (i) -> i);
```

Menggunakan metode Arrays.stream() dan String.chars().

```
LongStream is = Arrays.stream(new long[]{1,2,3,4});
IntStream is2 = "abc".chars();
```

Mengonversi Stream ke Koleksi atau Larik

Ada beberapa cara untuk mendapatkan Koleksi atau Array dari Stream.

Kita bisa menggunakan metode Stream collect() untuk mendapatkan List, Map atau Set dari stream.

```
Stream<Integer> intStream = Stream.of(1,2,3,4);
ist<Integer> intList = intStream.collect(Collectors.toList());
System.out.println(intList);
//prints [1, 2, 3, 4] intStream = Stream.of(1,2,3,4);
//stream is closed, so we need to create it again
Map<Integer,Integer> intMap = intStream.collect(Collectors.toMap(i ->
i, i -> i+10));
System.out.println(intMap);
//prints {1=11, 2=12, 3=13, 4=14}
```

Kita bisa menggunakan metode stream toArray() untuk membuat array dari stream.

```
Stream<Integer> intStream = Stream.of(1,2,3,4);
Integer[] intArray = intStream.toArray(Integer[]::new);
System.out.println(Arrays.toString(intArray)); //prints [1, 2, 3, 4]
```

Operasi Menengah Streaming

Mari kita lihat contoh operasi menengah Stream yang umum digunakan.

Operasi Menengah Streaming- Mari kita lihat contoh operasi menengah Stream yang umum digunakan.

```
List<Integer> myList = new ArrayList<>();
for(int i=0; i<100;
i++) myList.add(i);
Stream<Integer> sequentialStream = myList.stream();
```

```
        Stream<Integer> highNums = sequentialStream.filter(p -> p > 90);
        //filter numbers greater than 90 System.out.print("High Nums greater
than 90=");
        highNums.forEach(p -> System.out.print(p+" "));
        //prints "High Nums greater than 90=91 92 93 94 95 96 97 98 99 "
```

Contoh stream map(): Kita bisa menggunakan map() untuk menerapkan fungsi ke stream. Mari kita lihat bagaimana kita dapat menggunakannya untuk menerapkan fungsi huruf besar ke daftar String.

```
        Stream<String> names = Stream.of("aBc", "d", "ef");
        System.out.println(names.map(s -> {
        return s.toUpperCase();
        }).collect(Collectors.toList()));
        //prints [ABC, D, EF]
```

Contoh stream sort(): Kita bisa menggunakan sort() untuk mengurutkan elemen stream dengan meneruskan argumen Comparator.

```
        Stream<String> names2 = Stream.of("aBc", "d", "ef", "123456");
        List<String> reverseSorted =
        names2.sorted(Comparator.reverseOrder()).collect(Collectors.toList());
        System.out.println(reverseSorted);
        // [ef, d, aBc, 123456]
        Stream<String> names3 = Stream.of("aBc", "d", "ef", "123456");
        List<String> naturalSorted = names3.sorted().collect(Collectors.toList());
        System.out.println(naturalSorted);
        //[123456, aBc, d, ef]
```

Stream flatMap() Contoh: Kita bisa menggunakan flatMap() untuk membuat stream dari daftar stream. Mari kita lihat contoh sederhana untuk menghilangkan keraguan ini.

```
        Stream<List<String>> namesOriginalList = Stream.of(
        Arrays.asList("Pankaj"),
        Arrays.asList("David", "Lisa"),
        Arrays.asList("Amit"));
        //flat the stream from List<String> to String stream
        Stream<String> flatStream = namesOriginalList
        .flatMap(strList -> strList.stream());
        flatStream.forEach(System.out::println);
```

Operasi Terminal Streaming

Mari kita lihat beberapa contoh operasi terminal.

Contoh pengurangan stream(): Kita bisa menggunakan reduce() untuk melakukan pengurangan elemen stream, menggunakan fungsi akumulasi asosiatif, dan mengembalikan Opsional. Mari kita lihat bagaimana kita bisa menggunakannya untuk mengalikan bilangan bulat dalam stream.

```
        Stream<Integer> numbers = Stream.of(1,2,3,4,5);
        Optional<Integer> intOptional = numbers.reduce((i,j) -> {return i*j;});
        if(intOptional.isPresent()) System.out.println("Multiplication =
"+intOptional.get()); //120
```

Contoh jumlah stream(): Kita dapat menggunakan operasi terminal ini untuk menghitung jumlah item dalam stream.

```
        Stream<Integer> numbers1 = Stream.of(1,2,3,4,5);
        System.out.println("Number of elements in
stream="+numbers1.count()); //5
```

Contoh Stream forEach(): Ini dapat digunakan untuk iterasi melalui stream. Kita bisa menggunakan ini sebagai pengganti iterator. Mari kita lihat cara menggunakannya untuk mencetak semua elemen stream.

```
        Stream<Integer> numbers2 = Stream.of(1,2,3,4,5);
        numbers2.forEach(i -> System.out.print(i+",")); //1,2,3,4,5,
```

Contoh stream match(): Mari kita lihat beberapa contoh metode pencocokan di Stream API.

```
        Stream<Integer> numbers3 = Stream.of(1,2,3,4,5);
        System.out.println("Stream contains 4? "+numbers3.anyMatch(i ->
i==4));
        //Stream contains 4? true Stream<Integer> numbers4 =
Stream.of(1,2,3,4,5);
        System.out.println("Stream contains all elements less than 10?
"+numbers4.allMatch(i -> i<10));
        //Stream contains all elements less than 10? true Stream<Integer>
numbers5 = Stream.of(1,2,3,4,5);
        System.out.println("Stream doesn't contain 10?
"+numbers5.noneMatch(i -> i==10));
        //Stream doesn't contain 10? true
```

Contoh stream findFirst(): Ini adalah operasi terminal hubung singkat, mari kita lihat bagaimana kita dapat menggunakannya untuk menemukan string pertama dari stream yang dimulai dengan D.

```
        Stream<String> names4 = Stream.of("Pankaj","Amit","David", "Lisa");
        Optional<String> firstNameWithD = names4.filter(i -> i.startsWith("D")).findFirst();
        if(firstNameWithD.isPresent()){
        System.out.println("First Name starting with D="+firstNameWithD.get()); //David
        }
```

Batasan Java Stream API

Stream API menghadirkan banyak hal baru untuk dikerjakan dengan daftar dan array, tetapi juga memiliki beberapa keterbatasan.

Lambda expression tanpa status: Jika Anda menggunakan stream paralel dan Lambda expression berstatus stateful, ini bisa menghasilkan respons acak. Mari kita lihat dengan program sederhana.

StatefulParallelStream.java

```
        package com.journaldev.java8.stream;
        import java.util.ArrayList;
        import java.util.Arrays;
        import java.util.List;
        import java.util.stream.Stream;
        public class StatefulParallelStream {
        public static void main(String[] args) {
        List<Integer> ss = Arrays.asList(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15);
        List<Integer> result = new ArrayList<Integer>();
        Stream<Integer> stream = ss.parallelStream();
0       stream.map(s -> {
        synchronized (result) {
1       if (result.size() < 10) {
        result.add(s);
2       }
        }
3       return s;
        }).forEach( e -> {});
4       System.out.println(result);
5
6
7
8
```

9	

Jika kami menjalankan program di atas, Anda akan mendapatkan hasil yang berbeda karena bergantung pada cara pengulangan streaming dan kami tidak menetapkan urutan apa pun untuk pemrosesan paralel. Jika kita menggunakan stream sekuensial, maka masalah ini tidak akan muncul.

Setelah digunakan, Stream tidak dapat digunakan nanti. Seperti yang Anda lihat pada contoh di atas bahwa setiap kali saya membuat stream.

Ada banyak metode dalam Stream API dan bagian yang paling membingungkan adalah metode yang kelebihan muatan. Itu membuat kurva belajar memakan waktu. Itu saja untuk Stream API di Java. Saya tidak sabar untuk menggunakan fitur ini dan membuat kode dapat dibaca dengan kinerja yang lebih baik melalui pemrosesan paralel.

Java Time API

Bekerja dengan Tanggal, Waktu dan Zona Waktu di java selalu sulit. Tidak ada pendekatan standar atau API di java untuk tanggal dan waktu di Java. Salah satu tambahan yang bagus di Java 8 adalah paket java.time yang akan merampingkan proses bekerja dengan waktu di java.

Hanya dengan melihat paket Java Time API, saya dapat merasakan bahwa ini akan sangat mudah digunakan. Ini memiliki beberapa sub-paket java.time.format yang menyediakan kelas untuk mencetak dan mengurai tanggal dan waktu dan java.time.zon memberikan dukungan untuk zona waktu dan aturannya.

API Waktu baru lebih memilih enum daripada konstanta bilangan bulat untuk bulan dan hari dalam seminggu. Salah satu kelas yang berguna adalah DateTimeFormatter untuk mengonversi objek datetime menjadi string.

Peningkatan Collection API

Kami telah melihat metode forEach() dan Stream API untuk koleksi. Beberapa metode baru yang ditambahkan di Collection API adalah:

Metode default Iterator untuk- EachRemaining (Tindakan konsumen) untuk melakukan tindakan yang diberikan untuk setiap elemen yang tersisa sampai semua elemen telah diproses atau tindakan tersebut mengeluarkan pengecualian.

Metode default koleksi- removeIf (Predicate filter) untuk menghapus semua elemen dari koleksi ini yang memenuhi predikat yang diberikan.

Collection spliterator() method- mengembalikan instance Spliterator yang dapat digunakan untuk melintasi elemen secara berurutan atau paralel.

Memetakan metode replaceAll(), compute(), merge().

Peningkatan Concurrency API

Beberapa penyempurnaan API bersamaan yang penting adalah:

ConcurrentHashMap compute(), forEach(), forEachEntry(), forEachKey(), forEachValue(), merge(), reduce() dan search().

CompletableFuture yang mungkin diselesaikan secara eksplisit (menyetel nilai dan statusnya).

Pelaksana metode newWorkStealingPool() untuk membuat kumpulan thread yang mencuri pekerjaan menggunakan semua prosesor yang tersedia sebagai level paralelisme targetnya.

Perbaikan Java IO

Beberapa perbaikan IO yang saya ketahui adalah:

Files.list (Path dir) yang mengembalikan Stream yang lambat diisi, yang elemennya adalah entri dalam direktori.

Files.lines (Path path) yang membaca semua baris dari file sebagai Stream.

Files.find() yang mengembalikan Stream yang diisi dengan Path secara malas dengan mencari file di pohon file yang di-root pada file awal yang diberikan.

BufferedReader.lines() yang mengembalikan Stream, yang elemennya adalah baris yang dibaca dari BufferedReader ini.

Perbaikan Core API Miscellaneous

Beberapa peningkatan misc API yang mungkin berguna adalah:

Metode statis ThreadLocal denganInitial (Supplier supplier) untuk membuat instance dengan mudah.

Antarmuka pembanding telah diperpanjang dengan banyak metode default dan statis untuk pengurutan alami, urutan terbalik, dll.

min(), max() dan sum() di wrapper classes Integer, Long dan Double.

metode logicalAnd(), logicalOr() dan logicalXor() di kelas Boolean.

ZipFile.stream() untuk mendapatkan Stream yang teratur melalui entri file ZIP. Entri muncul di Stream dalam urutan kemunculannya di direktori pusat file ZIP.

Beberapa metode utilitas di kelas Matematika.

Sekian untuk peningkatan besar di Java 8.

Core Java : Pertanyaan Dan Jawaban Wawancara *Brain Wash*

Apakah kamu seorang profesional yang lebih segar atau sangat berpengalaman, core java memainkan peran penting dalam setiap wawancara Java/JEE. Core Java adalah area favorit di sebagian besar wawancara dan memainkan peran penting dalam menentukan hasil wawancaramu. Saya telah menulis banyak tentang pertanyaan wawancara java untuk topik tertentu seperti String, Collections dan Multithreading.

- Pertanyaan Wawancara String Java
- Pertanyaan Wawancara Thread Java
- Pertanyaan Wawancara Java Collection
- Pertanyaan Wawancara Jawa Exception

Di sini saya memberikan beberapa pertanyaan wawancara core java yang penting dengan jawaban yang harus kamu ketahui.

- *Apa saja fitur penting dari rilis Java 8?*
- *Apa yang dimaksud dengan independensi platform Java?*
- *Apa itu JVM dan apakah itu platform independen?*
- *Apa perbedaan antara JDK dan JVM?*
- *Apa perbedaan antara JVM dan JRE?*
- *Kelas mana yang merupakan superclass dari semua kelas?*
- *Mengapa Java tidak mendukung multiple inheritance?*
- *Mengapa Java bukan bahasa asli Object Oriented?*
- *Apa perbedaan antara variabel path dan classpath?*
- *Apa pentingnya metode utama di Java?*
- *Apa itu overloading dan overriding di java?*
- *Bisakah kita overloading metode utama?*
- *Bisakah kita memiliki beberapa public classes dalam java source file?*
- *Apa itu Paket Java dan paket mana yang diimpor secara default?*
- *Apa itu access modifier?*
- *Apa itu final keyword?*
- *Apa itu static keyword?*
- *Apa itu finally dan finalize di java?*
- *Bisakah kita mendeklarasikan kelas sebagai statis?*
- *Apa itu impor statis?*
- *Apa itu try-with-resources di java?*
- *Apa itu blok multi-catch di java?*
- *Apa itu blok statis?*
- *Apa itu interface/antarmuka?*
- *Apa itu kelas abstrak?*
- *Apa perbedaan antara Kelas Abstrak dan Antarmuka?*
- *Dapatkah sebuah antarmuka menerapkan atau memperluas antarmuka lain?*
- *Apa itu Marker interface?*
- *Apa itu Wrapper class?*
- *Apa itu Enum di java?*
- *Apa itu Anotasi Java?*
- *API Refleksi? Mengapa sangat penting untuk dimiliki?*
- *Apa komposisi di java?*
- *Apa manfaat compisition dibandingkan Inheritance?*
- *Bagaimana cara mengurutkan collection pada Objek kustom di Java?*
- *Apa itu inner class di java?*
- *Apa itu inner class anonim?*
- *Apa itu Classloader di Java?*
- *Apa jenis classloader yang berbeda?*
- *Apa itu operator terner di java?*

- *Apa yang dilakukan keyword super?*
- *Apa itu pernyataan break dan continue?*
- *Apa keyword ini?*
- *Apa itu konstruktor default?*
- *Bisakah kita mencoba tanpa catch block?*
- *Apa itu Garbage Collection?*
- *Apa itu Serialisasi dan Deserialisasi?*
- *Bagaimana cara menjalankan file JAR melalui command prompt?*
- *Apa gunanya system class?*
- *Apa itu instanceof keyword?*
- *Bisakah kita menggunakan String dengan switch case?*
- *Apa hasil dari program-program berikut?*

Penjelasan

- *Apa saja fitur penting dari rilis Java 8?*

Java 8 telah dirilis pada Maret 2014, jadi ini adalah salah satu topik hangat dalam pertanyaan wawancara java. Jika Anda menjawab pertanyaan ini dengan jelas, itu akan menunjukkan bahwa Anda ingin selalu mengikuti perkembangan teknologi terkini. Java 8 telah menjadi salah satu rilis terbesar setelah penjelasan dan generik Java 5. Beberapa fitur penting Java 8 adalah:

Interface berubah dengan metode default dan statis.
Interface fungsional dan Lambda expression.
Java Stream API untuk kelas koleksi.
Java Date Time API.

- *Apa yang dimaksud dengan independensi platform Java?*

Independensi platform berarti Anda dapat menjalankan Program Java yang sama di Sistem Operasi apa pun. Misalnya, Anda dapat menulis program java di Windows dan menjalankannya di Mac OS.

- *Apa itu JVM dan apakah itu platform independen?*

Java Virtual Machine (JVM) adalah jantung dari bahasa pemrograman java. JVM bertanggung jawab untuk mengubah kode byte menjadi kode yang dapat dibaca mesin. JVM bukanlah platform independen, itulah mengapa Anda memiliki JVM yang berbeda untuk sistem operasi yang berbeda. Kita dapat menyesuaikan JVM dengan Opsi Java, seperti mengalokasikan memori minimum dan maksimum ke JVM. Ini disebut virtual karena menyediakan antarmuka yang tidak bergantung pada OS yang mendasarinya.

- *Apa perbedaan antara JDK dan JVM?*

Java Development Kit (JDK) adalah untuk tujuan pengembangan dan JVM adalah bagian darinya untuk menjalankan program java.

JDK menyediakan semua alat, file yang dapat dieksekusi, dan biner yang diperlukan untuk mengkompilasi, men-debug, dan menjalankan Program Java. Bagian eksekusi ditangani oleh JVM untuk memberikan kebebasan mesin.

- *Apa perbedaan antara JVM dan JRE?*

Java Runtime Environment (JRE) adalah implementasi JVM. JRE terdiri dari JVM dan java binari dan kelas lain untuk mengeksekusi program apapun dengan sukses. JRE tidak berisi alat pengembangan seperti compiler java, debugger dll. Jika Anda ingin menjalankan program java, Anda harus menginstal JRE.

- *Kelas mana yang merupakan superclass dari semua kelas?*

java.lang.Object adalah kelas root untuk semua kelas java dan kita tidak perlu memperluasnya.

- *Mengapa Java tidak mendukung multiple inheritance?*

Java tidak mendukung multiple inheritance dalam kelas karena "Diamond Problem". Untuk mengetahui lebih banyak tentang Diamond Problem dengan contoh, baca multiple inheritance di Java. Namun beberapa peinheritance didukung di antarmuka. Sebuah antarmuka dapat memperluas banyak antarmuka karena mereka hanya mendeklarasikan metode dan implementasinya akan hadir di kelas yang mengimplementasikan. Jadi tidak ada Diamond Problem dengan antarmuka.

- *Mengapa Java bukan bahasa asli Object Oriented?*

Java tidak dikatakan murni berorientasi objek karena mendukung tipe primitif seperti int, byte, short, long dll. Saya percaya itu membawa kesederhanaan pada bahasa saat menulis kode kita. Jelas java bisa memiliki objek pembungkus untuk tipe primitif tetapi hanya untuk representasi, mereka tidak akan memberikan manfaat apa pun.

Seperti yang kita ketahui, untuk semua tipe primitif kita memiliki wrapper classes seperti Integer, Long dll yang menyediakan beberapa metode tambahan.

- *Apa perbedaan antara variabel path dan classpath?*

PATH adalah variabel lingkungan yang digunakan oleh sistem operasi untuk menemukan file yang dapat dieksekusi. Itulah mengapa ketika kita menginstal Java atau ingin file executable ditemukan oleh OS, kita perlu menambahkan lokasi direktori di variabel PATH. Jika Anda bekerja pada OS Windows, Classpath dikhususkan untuk java dan digunakan oleh java executable untuk mencari file kelas. Kami dapat menyediakan lokasi classpath saat menjalankan aplikasi java dan dapat berupa direktori, file ZIP, file JAR, dll.

- *Apa pentingnya metode utama di Java?*

mai() metode adalah titik masuk dari setiap aplikasi java mandiri. Sintaks dari metode utama ispublic static void main (String args []). metode utama bersifat publik dan statis sehingga java dapat mengaksesnya tanpa menginisialisasi kelas. Parameter input adalah larik String yang melaluinya kita dapat meneruskan argumen runtime ke program java.

- *Apa itu overloading dan overriding di java?*

Ketika kita memiliki lebih dari satu metode dengan nama yang sama dalam satu kelas tetapi argumennya berbeda, maka itu disebut metode overloading. Konsep overriding muncul dalam gambar dengan peinheritance ketika kita memiliki dua metode dengan tanda tangan yang sama, satu di kelas induk dan satu lagi di kelas anak.

Kita dapat menggunakan anotasi @Override dalam metode penggantian kelas anak untuk memastikan jika metode kelas induk diubah, begitu juga dengan kelas anak.

- *Bisakah kita overloading metode utama?*

Ya, kita bisa memiliki beberapa metode dengan nama “main” dalam satu kelas. Namun jika kita menjalankan kelas, java runtime environment akan mencari metode utama dengan sintaks sebagai public static void main (String args []).

- *Bisakah kita memiliki beberapa public classes dalam java source file?*

Kami tidak dapat memiliki lebih dari satu kelas publik dalam satu file sumber java. Satu file sumber dapat memiliki beberapa kelas yang tidak bersifat publik.

- *Apa itu Paket Java dan paket mana yang diimpor secara default?*

Paket Java adalah mekanisme untuk mengatur kelas-kelas java dengan mengelompokkannya. Logika pengelompokan dapat didasarkan pada fungsionalitas atau berbasis modul. Sebuah kelas java nama yang sepenuhnya diklasifikasikan berisi nama paket dan kelas. Misalnya, java.lang.Object adalah nama class Object yang sepenuhnya diklasifikasikan yang merupakan bagian dari paket java.lang. Paket java.lang diimpor secara default dan kami tidak perlu mengimpor kelas apa pun dari paket ini secara eksplisit.

- *Apa itu access modifier?*

Java menyediakan kontrol akses melalui keyword pengubah akses publik, pribadi dan dilindungi. Jika tidak ada yang digunakan, ini disebut pengubah akses default. Kelas java hanya dapat memiliki pengubah akses publik atau default.

- *Apa itu final keyword?*

Final keyword digunakan dengan Class untuk memastikan tidak ada kelas lain yang dapat memperpanjangnya, misalnya kelas String adalah final dan kami tidak dapat memperpanjangnya. Kita dapat menggunakan keyword terakhir dengan metode untuk memastikan kelas anak tidak dapat menimpanya.

Final keyword dapat digunakan dengan variabel untuk memastikan bahwa keyword tersebut hanya dapat diberikan sekali. Namun status variabel dapat diubah, misalnya kita dapat menetapkan variabel terakhir ke objek hanya sekali tetapi variabel objek dapat berubah nanti. Variabel antarmuka Java secara default final dan statis.

- *Apa itu static keyword?*

keyword statis dapat digunakan dengan variabel tingkat kelas untuk membuatnya global, yaitu semua objek akan berbagi variabel yang sama. keyword statis juga dapat digunakan dengan metode. Metode statis hanya dapat mengakses variabel statis kelas dan hanya memanggil metode statis kelas.

- *Apa itu finally dan finalize di java?*

Finally block digunakan dengan try-catch untuk meletakkan kode yang ingin Anda jalankan selalu, bahkan jika ada pengecualian yang dilemparkan oleh blok try-catch.

Finally block sebagian besar digunakan untuk melepaskan sumber daya yang dibuat di blok percobaan. finalize() adalah metode khusus di kelas Objek yang bisa kita ganti di kelas kita.

Metode ini dipanggil oleh pengumpul sampah saat objek mendapatkan sampah yang dikumpulkan. Metode ini biasanya diganti untuk melepaskan sumber daya sistem saat objek dikumpulkan sampahnya.

- *Bisakah kita mendeklarasikan kelas sebagai statis?*

Kami tidak dapat mendeklarasikan kelas tingkat atas sebagai statis, namun inner class dapat dideklarasikan sebagai statis. Jika inner class dideklarasikan sebagai statis, itu disebut kelas bertingkat statis. Kelas bertingkat statis sama dengan kelas tingkat atas lainnya dan disarangkan hanya untuk kenyamanan pengemasan.

- *Apa itu impor statis?*

Jika kita harus menggunakan variabel atau metode statis dari kelas lain, biasanya kita mengimpor kelas dan kemudian menggunakan metode/variabel dengan nama kelas.

```
import java.lang.Math;
//inside class
double test = Math.PI * 5;
```

Kita dapat melakukan hal yang sama dengan mengimpor metode atau variabel statis saja dan kemudian menggunakannya di kelas seolah-olah itu miliknya.

```
import static java.lang.Math.PI;

//no need to refer class now
double test = PI * 5;
```

Penggunaan impor statis dapat menyebabkan kebingungan, jadi lebih baik menghindarinya. Penggunaan impor statis yang berlebihan dapat membuat program Anda tidak dapat dibaca dan tidak dapat dikelola.

- *Apa itu try-with-resources di java?*

Salah satu fitur Java 7 adalah pernyataan coba-dengan-sumber daya untuk pengelolaan sumber daya otomatis. Sebelum Java 7, tidak ada pengelolaan sumber daya otomatis dan kita harus menutup sumber daya secara eksplisit. Biasanya, ini dilakukan di finally block dari pernyataan coba-tangkap.

Pendekatan ini biasanya menyebabkan kebocoran memori saat kita lupa menutup sumber daya. Dari Java 7, kita dapat membuat sumber daya di dalam blok coba dan menggunakannya. Java akan menutupnya segera setelah blok try-catch selesai.

- *Apa itu blok multi-catch di java?*

Java 7 salah satu perbaikannya adalah multi-catch block dimana kita dapat menangkap banyak pengecualian dalam satu catch blockan. Ini membuat kode lebih pendek dan lebih bersih ketika setiap catch blockan memiliki kode yang serupa. Jika blok penangkap menangani beberapa pengecualian, Anda dapat memisahkannya

menggunakan pipa (|) dan dalam hal ini parameter pengecualian (mis) bersifat final, jadi Anda tidak dapat mengubahnya.

- *Apa itu blok statis?*

Blok statis Java adalah grup pernyataan yang dijalankan saat kelas dimuat ke dalam memori oleh Java ClassLoader. Ini digunakan untuk menginisialisasi variabel statis kelas. Sebagian besar digunakan untuk membuat sumber daya statis saat kelas dimuat.

- *Apa itu interface/antarmuka?*

Antarmuka adalah bagian inti dari bahasa pemrograman java dan banyak digunakan tidak hanya di JDK tetapi juga pola desain java, sebagian besar kerangka kerja dan alat. Antarmuka menyediakan cara untuk mencapai abstraksi di java dan digunakan untuk menentukan kontrak yang akan diterapkan subclass. Antarmuka bagus sebagai titik awal untuk menentukan Jenis dan membuat hierarki tingkat atas dalam kode kita. Karena kelas java dapat mengimplementasikan banyak antarmuka, lebih baik menggunakan antarmuka sebagai kelas super di sebagian besar kasus.

- *Apa itu kelas abstrak?*

Kelas abstrak digunakan di java untuk membuat kelas dengan beberapa implementasi metode default untuk subkelas. Kelas abstrak dapat memiliki metode abstrak tanpa isi dan dapat juga memiliki metode dengan implementasi. keyword abstrak digunakan untuk membuat kelas abstrak.

Kelas abstrak tidak dapat dibuat instance-nya dan sebagian besar digunakan untuk menyediakan basis bagi sub-kelas untuk memperluas dan mengimplementasikan metode abstrak dan mengganti atau menggunakan metode yang diimplementasikan dalam kelas abstrak.

- *Apa perbedaan antara Kelas Abstrak dan Antarmuka?*

keyword abstrak digunakan untuk membuat kelas abstrak sedangkan antarmuka adalah keyword untuk antarmuka. Kelas abstrak dapat memiliki implementasi metode sedangkan antarmuka tidak. Sebuah kelas hanya dapat memperluas satu kelas abstrak tetapi dapat mengimplementasikan banyak antarmuka. Kita bisa menjalankan kelas abstrak jika memiliki metode main() sedangkan kita tidak bisa menjalankan antarmuka. Beberapa perbedaan lebih detail ada di Perbedaan antara Kelas Abstrak dan Antarmuka.

- *Dapatkah sebuah antarmuka menerapkan atau memperluas antarmuka lain?*

Antarmuka tidak mengimplementasikan antarmuka lain, mereka memperluasnya. Karena antarmuka tidak dapat memiliki implementasi metode, tidak ada Diamond Problem. Itulah mengapa kami memiliki banyak peinheritance dalam antarmuka, yaitu antarmuka dapat memperluas banyak antarmuka.

- *Apa itu Marker interface?*

Antarmuka marker adalah antarmuka kosong tanpa metode apa pun, tetapi digunakan untuk memaksa beberapa fungsionalitas dalam mengimplementasikan kelas dengan Java. Beberapa antarmuka penanda terkenal adalah Dapat Berseri dan Dapat Dikloning.

- *Apa itu Wrapper class?*

Wrapper Class Java adalah representasi Objek dari delapan tipe primitif di java. Semua wrapper classes di java tidak dapat diubah dan final. Autoboxing dan unboxing Java 5 memungkinkan konversi yang mudah antara tipe primitif dan wrapper classesnya yang sesuai.

- *Apa itu Enum di java?*

Enum diperkenalkan di Java 1.5 sebagai tipe baru yang bidangnya terdiri dari kumpulan konstanta tetap. Misalnya, di Java kita dapat membuat Arah sebagai enum dengan bidang tetap seperti TIMUR, BARAT, UTARA, SELATAN. enum adalah keyword untuk membuat jenis enum dan mirip dengan class. Konstanta enum secara implisit bersifat statis dan final.

- *Apa itu Anotasi Java?*

Anotasi Java memberikan informasi tentang kode dan tidak memiliki efek langsung pada kode yang dianotasinya. Penjelasan diperkenalkan di Java 5. Anotasi adalah metadata tentang program yang tertanam dalam program itu sendiri. Ini dapat diurai dengan alat parsing anotasi atau oleh kompiler. Kami juga dapat menentukan ketersediaan anotasi baik untuk waktu kompilasi saja atau hingga waktu proses juga. Anotasi bawaan Java adalah @Override, @Deprecated, dan @SuppressWarnings.

- *Apa itu Java Reflection API? Mengapa sangat penting untuk dimiliki?*

Java Reflection API menyediakan kemampuan untuk memeriksa dan mengubah perilaku runtime aplikasi java. Kita dapat memeriksa kelas java, antarmuka, enum dan mendapatkan metode dan detail bidangnya. API Refleksi adalah topik tingkat lanjut dan kita harus menghindarinya dalam pemrograman normal. Penggunaan API Refleksi dapat merusak pola desain seperti pola Singleton dengan memanggil konstruktor privat, yaitu melanggar aturan pengubah akses.

Meskipun kami tidak menggunakan API Refleksi dalam pemrograman normal, itu sangat penting untuk dimiliki. Kami tidak dapat memiliki kerangka kerja apa pun seperti Spring, Hibernate, atau server seperti Tomcat, JBoss tanpa API Refleksi. Mereka memanggil metode yang sesuai dan membuat instance kelas melalui API refleksi dan banyak menggunakannya untuk pemrosesan lain.

- *Apa komposisi di java?*

Komposisi adalah teknik desain untuk mengimplementasikan hubungan has-a di kelas. Kita dapat menggunakan komposisi Objek untuk penggunaan kembali kode. Komposisi Java dicapai dengan menggunakan variabel instan yang mengacu pada objek lain. Manfaat menggunakan komposisi adalah kita dapat mengontrol visibilitas objek lain ke kelas klien dan hanya menggunakan kembali yang kita butuhkan.

- *Apa manfaat compisition dibandingkan Inheritance?*

Salah satu praktik terbaik pemrograman java adalah "lebih mengutamakan komposisi daripada inheritance". Beberapa kemungkinan alasannya adalah:

Setiap perubahan dalam superclass mungkin memengaruhi subclass meskipun kita mungkin tidak menggunakan metode superclass. Misalnya, jika kita memiliki metode test() di subclass dan tiba-tiba seseorang memperkenalkan metode test() di

superclass, kita akan mendapatkan kesalahan kompilasi di subclass. Komposisi tidak akan pernah menghadapi masalah ini karena kami hanya menggunakan metode yang kami butuhkan.

Peinheritance memperlihatkan semua metode dan variabel kelas super kepada klien dan jika kita tidak memiliki kendali dalam mendesain kelas super, hal itu dapat menyebabkan lubang keamanan. Komposisi memungkinkan kami memberikan akses terbatas ke metode dan karenanya lebih aman.

Kita bisa mendapatkan pengikatan waktu proses dalam komposisi di mana peinheritance mengikat kelas-kelas pada waktu kompilasi. Jadi komposisi memberikan fleksibilitas dalam pemanggilan metode.

- *Bagaimana cara mengurutkan collection pada Objek kustom di Java?*

Kita perlu mengimplementasikan antarmuka Comparable untuk mendukung pengurutan objek kustom dalam koleksi. Antarmuka yang dapat dibandingkan memiliki metode bandingkanTo (T obj) yang digunakan oleh metode pengurutan dan dengan menyediakan implementasi metode ini, kami dapat menyediakan cara default untuk mengurutkan koleksi objek kustom.

Namun, jika Anda ingin mengurutkan berdasarkan kriteria yang berbeda, seperti mengurutkan koleksi Karyawan berdasarkan gaji atau usia, kami dapat membuat instance Pembanding dan meneruskannya sebagai metodologi pengurutan.

- *Apa itu inner class di java?*

Kita dapat mendefinisikan kelas di dalam kelas dan mereka disebut nested class. Setiap nested class non-statis dikenal sebagai inner class. Kelas-inner class dikaitkan dengan objek kelas dan mereka dapat mengakses semua variabel dan metode kelas luar. Karena inner class dikaitkan dengan instance, kita tidak dapat memiliki variabel statis apa pun di dalamnya. Kita dapat memiliki inner class lokal atau inner class anonim di dalam kelas.

- *Apa itu inner class anonim?*

Inner class lokal tanpa nama dikenal sebagai inner class anonim. Kelas anonim didefinisikan dan dibuat dalam satu pernyataan. Inner class anonim selalu memperluas kelas atau mengimplementasikan antarmuka. Karena kelas anonim tidak memiliki nama, tidak mungkin untuk mendefinisikan konstruktor untuk kelas anonim. Inner class anonim hanya dapat diakses pada titik di mana kelas itu didefinisikan.

- *Apa itu Classloader di Java?*

Java Classloader adalah program yang memuat program kode byte ke dalam memori ketika kita ingin mengakses kelas apa pun. Kita dapat membuat classloader kita sendiri dengan memperluas kelas ClassLoader dan mengganti metode loadClass (Nama string).

- *Apa jenis classloader yang berbeda?*

Ada tiga jenis Class Loader bawaan di Java:

Bootstrap Class Loader - Memuat kelas internal JDK, biasanya memuat rt.jar dan kelas inti lainnya.

Extensions Class Loader - Memuat kelas dari direktori ekstensi JDK, biasanya direktori $JAVA_HOME/lib/ext.

System Class Loader - Memuat kelas dari classpath saat ini yang dapat disetel saat menjalankan program menggunakan opsi baris perintah -cp atau -classpath.

- *Apa itu operator terner di java?*

Operator terner Java adalah satu-satunya operator bersyarat yang membutuhkan tiga operan. Ini adalah pengganti satu baris untuk pernyataan if-then-else dan banyak digunakan dalam pemrograman java. Kita dapat menggunakan kondisi operator terner if-else atau bahkan mengganti kondisi menggunakan operator terner bersarang.

- *Apa yang dilakukan keyword super?*

keyword super dapat digunakan untuk mengakses metode kelas super ketika Anda telah mengganti metode dalam kelas anak. Kita dapat menggunakan keyword super untuk memanggil konstruktor kelas super dalam konstruktor kelas anak tetapi dalam hal ini harus menjadi pernyataan pertama dalam metode konstruktor.

SuperClass.java

```
package com.journaldev.access;
public class SuperClass {
public SuperClass(){
}
public SuperClass(int i){}
public void test(){
System.out.println("super class test method");
}
}
```

Penggunaan keyword super dapat dilihat pada implementasi kelas anak di bawah ini.

ChildClass.java

```
package com.journaldev.access;
public class ChildClass extends SuperClass {
public ChildClass(String str){
//access super class constructor with super keyword
super();
//access child class method
test();
//use super to access super class method
super.test();    }
@Override
public void test(){
System.out.println("child class test method");
}
}
```

2 3 4	

- *Apa itu pernyataan break dan continue?*

Kita dapat menggunakan pernyataan break untuk menghentikan for, while, atau do-while loop. Kita dapat menggunakan pernyataan break dalam pernyataan switch untuk keluar dari kasus switch. Anda dapat melihat contoh pernyataan break pada java break.

Kita bisa menggunakan break with label untuk menghentikan loop bersarang. Pernyataan lanjutkan melompati iterasi saat ini dari perulangan for, while atau do-while. Kita dapat menggunakan pernyataan continue dengan label untuk melewati iterasi saat ini dari loop terluar.

- *Apa keyword ini?*

Keyword ini memberikan referensi ke objek saat ini dan sebagian besar digunakan untuk memastikan bahwa variabel objek digunakan, bukan variabel lokal yang memiliki nama yang sama.

	//constructor
	public Point(int x, int y) {
	this.x = x;
	this.y = y;
	}

Kita juga dapat menggunakan keyword ini untuk memanggil konstruktor lain dari sebuah konstruktor.

	public Rectangle() {
	this(0, 0, 0, 0); }
	public Rectangle(int width, int height) {
	this(0, 0, width, height);
	}
	public Rectangle(int x, int y, int width, int height) {
	this.x = x;
	this.y = y;
	this.width = width;
	this.height = height;
	}

0 1	

- *Bisakah kita mencoba tanpa catch block?*

Ya, kita dapat memiliki pernyataan coba-akhirnya dan karenanya menghindari catch block.

- *Apa itu Garbage Collection?*

Garbage Collection adalah proses melihat memori tumpukan, mengidentifikasi objek mana yang sedang digunakan dan mana yang tidak, dan menghapus objek yang tidak digunakan. Di Java, proses pelepasan alokasi memori ditangani secara otomatis oleh pengumpul sampah.

Kita dapat menjalankan pengumpul sampah dengan kode Runtime.getRuntime(). Gc() atau menggunakan metode utilitasSystem.gc().

- *Apa itu Serialisasi dan Deserialisasi?*

Kita dapat mengubah objek Java menjadi Stream yang disebut Serialisasi. Setelah sebuah objek diubah menjadi Stream, itu dapat disimpan ke file atau dikirim melalui jaringan atau digunakan dalam koneksi soket.

Objek harus mengimplementasikan antarmuka Serializable dan kita dapat menggunakan java.io.ObjectOutputStream untuk menulis objek ke file atau ke objek OutputStream. Baca lebih lanjut di Serialisasi Java. Proses mengubah data aliran yang dibuat melalui serialisasi ke Objek disebut deserialization.

- *Bagaimana cara menjalankan file JAR melalui command prompt?*

Kita dapat menjalankan file jar menggunakan perintah java tetapi membutuhkan entri Kelas-Utama dalam file manifes jar. Main-Class adalah titik masuk dari jar dan digunakan oleh perintah java untuk mengeksekusi kelas.

- *Apa gunanya system class?*

System class Java adalah salah satu kelas inti. Salah satu cara termudah untuk mencatat informasi untuk debugging adalah metode System.out.print().

System class bersifat final sehingga kita tidak dapat membuat subkelas dan mengganti perilakunya melalui peinheritance. System class tidak menyediakan konstruktor publik apa pun, jadi kami tidak dapat membuat instance kelas ini dan itulah sebabnya semua metodenya bersifat statis. Beberapa metode utilitas system class adalah untuk salinan array, mendapatkan waktu saat ini, membaca variabel lingkungan.

- *Apa itu instanceof keyword?*

Kita dapat menggunakan keyword instanceof untuk memeriksa apakah suatu objek milik kelas atau tidak. Kita harus menghindari penggunaannya sebanyak mungkin. Penggunaan sampel adalah:

```
public static void main(String args[]){
Object str = new String("abc");
if(str instanceof String){
System.out.println("String value:"+str);
}
 if(str instanceof Integer){
System.out.println("Integer value:"+str);
```

```
}
}
```

Karena str berjenis String pada waktu proses, pertama jika pernyataan bernilai benar dan yang kedua bernilai salah.

- *Bisakah kita menggunakan String dengan switch case?*

Salah satu fitur Java 7 adalah peningkatan case switch dari allow Strings. Jadi jika Anda menggunakan Java 7 atau versi yang lebih tinggi, Anda dapat menggunakan String dalam pernyataan switch-case.

- *Apa hasil dari program-program berikut?*

metode statis di kelas

```
package com.journaldev.util;
public class Test {
public static String toString(){
System.out.println("Test toString called");
return "";
}
public static void main(String args[]){
System.out.println(toString());
}
}
```

0

Jawaban: Kode tidak dapat dikompilasi karena kita tidak dapat memiliki metode kelas Object dengan keyword statis. Anda akan mendapatkan kesalahan waktu kompilasi sebagai "Metode statis ini tidak dapat menyembunyikan metode instance dari Objek". Alasannya adalah bahwa metode statis adalah milik kelas dan karena setiap basis kelas adalah Object, kita tidak dapat memiliki metode yang sama di dalam instance maupun di dalam kelas.

Pemanggilanmetode statis

```
package com.journaldev.util;
public class Test {
public static String foo(){
System.out.println("Test foo called");
return "";
}
public static void main(String args[]){
Test obj = null;
System.out.println(obj.foo());
}
}
```

0

1	

Jawaban: Ini adalah situasi yang aneh. Kita semua telah melihat NullPointerException saat memanggil metode pada objek yaitu NULL. Kompilator akan memberikan peringatan sebagai "Metode statis foo() dari tipe Test harus diakses secara statis" tetapi ketika mengeksekusinya akan bekerja dan mencetak "Test foo dipanggil".

Idealnya Java API seharusnya memberikan kesalahan saat metode statis dipanggil dari objek daripada memberikan peringatan, tapi saya pikir sudah terlambat sekarang untuk memaksakan ini. Dan yang paling aneh dari semuanya adalah bahwa meskipun obj adalah nol di sini, ketika menggunakan metode statis, ia berfungsi dengan baik. Saya rasa ini berfungsi dengan baik karena Java runtime mengetahui bahwa foo() adalah metode statis dan memanggilnya pada kelas yang dimuat ke dalam memori dan tidak menggunakan objek sama sekali, jadi tidak ada NullPointerException.

Ingat: Saya harus mengakui bahwa ini adalah pertanyaan yang sangat rumit dan jika Anda mewawancarai seseorang, ini akan membuat dia terkejut.

- *Apa itu konstruktor default?*

Tidak ada konstruktor argumen kelas yang dikenal sebagai konstruktor default. Jika kita tidak mendefinisikan konstruktor apa pun untuk kelas tersebut, kompilator java secara otomatis membuat konstruktor noargs default untuk kelas tersebut. Jika ada konstruktor lain yang ditentukan, maka compiler tidak akan membuat konstruktor default untuk kita.

BAB 27
KESEPAKATAN TERAKHIR
SEKARANG MULAI JAVA DAN UCAPKAN SELAMAT TINGGAL JAVA...

Pengantar :

Saya telah membahas setiap pertanyaan dari dunia Java. Sementara pertanyaan terkait bahasa Javar ditemukan di seluruh buku ini, bab ini membahas pertanyaan tentang bahasa dan sintaksis.

Pertanyaan semacam itu lebih tidak biasa di perusahaan besar, yang lebih percaya pada pengujian bakat kandidat daripada pengetahuan kandidat (dan yang memiliki waktu dan sumber daya untuk melatih kandidat dalam bahasa tertentu). Namun, di perusahaan lain, pertanyaan mengganggu ini bisa sangat umum.

Bagaimana Melakukan Pendekatan

Karena pertanyaan-pertanyaan ini terlalu terfokus pada pengetahuan, mungkin tampak konyol untuk membicarakan pendekatan terhadap masalah-masalah ini. Lagipula, bukankah ini hanya tentang mengetahui jawaban yang benar?

Iya dan tidak. Tentu saja, hal terbaik yang dapat Anda lakukan untuk menguasai pertanyaan-pertanyaan ini adalah mempelajari Java luar dalam. Tetapi, jika Anda bingung, Anda dapat mencoba mengatasinya dengan pendekatan berikut:

- Buat contoh skenario, dan tanyakan pada diri Anda bagaimana hal-hal seharusnya berjalan.
- Tanyakan pada diri Anda sendiri bagaimana bahasa lain akan menangani skenario ini.
- Pertimbangkan bagaimana Anda akan merancang situasi ini jika Anda adalah desainer bahasa.

Apa implikasi dari setiap pilihan?

Pewawancara Anda mungkin sama atau lebih terkesan jika Anda bisa mendapatkan jawabannya daripada jika Anda secara otomatis mengetahuinya. Namun, jangan mencoba menggertak. Katakan kepada pewawancara, "Saya tidak yakin saya dapat mengingat jawabannya, tetapi biarkan saya melihat apakah saya dapat memahaminya. Misalkan kita memiliki kode ini ..."

keyword terakhir

Saya sudah menjelaskannya beberapa kali. Keyword terakhir di Java memiliki arti yang berbeda tergantung pada apakah itu diterapkan ke variabel, kelas atau metode.

Variabel: Nilai tidak dapat diubah setelah diinisialisasi.

Metode: Metode ini tidak dapat diganti oleh subclass.

C/oss.-Kelas tidak bisa disubkelas.

Finally keyword

Keyword terakhir digunakan terkait dengan blok coba/tangkap dan menjamin bahwa bagian kode akan dieksekusi, bahkan jika pengecualian dilemparkan. Finally block akan dieksekusi setelah blok coba dan tangkap, tetapi sebelum kontrol ditransfer kembali ke asalnya. Perhatikan bagaimana ini dimainkan pada contoh di bawah ini.

```
public static String lem() {
System.out.println("lem")J
return "return from lem";
}
public static String foo() {
int x = 0;
int y = 5;
try {
System.out.println("start try");
int b = y/x;
System.out.println("end try");
return "returned from try";
} catch (Exception ex) {
System.out.println("catch");
return lem() + " | returned from catch";
} finally {
System.out.println("finally");
}
}
public static void bar() {
System.out.println("start bar");
String v = foo();
System.out.println(v);
System.out.println("end bar");
}
public static void main(String[] args) {
bar();
}
```

The output for this code is the following:

```
1 start bar
2 start try
3 catch
4 lem
5 finally
6 return from lem | returned from catch
7 end bar
```

Perhatikan baik-baik baris 3 hingga 5 di keluaran. Blok catch dijalankan sepenuhnya (termasuk pemanggilan fungsi dalam pernyataan return), lalu blok last, dan kemudian fungsi tersebut benar-benar kembali.

Menyelesaikan metode

Pengumpul sampah otomatis memanggil metode finalize() sebelum benar-benar menghancurkan objek. Oleh karena itu, kelas dapat menimpa metode f inalize() dari kelas Object untuk mendefinisikan perilaku kustom selama Garbage Collection.

```
1 protected void finalizeQ throws Throwable {
2/* Tutup file yang terbuka, lepaskan sumber daya, dll * /
3}
```

Overloading vs. Overriding

Overloading adalah istilah yang digunakan untuk mendeskripsikan ketika dua metode memiliki nama yang sama tetapi berbeda dalam jenis atau jumlah argumen.

```
1 public double computeArea (Lingkaran c) {...}
2 public double computeArea (Persegi) {...}
```

Namun, penimpaan terjadi saat metode berbagi nama dan fungsi tanda tangan yang sama dengan metode lain di kelas supernya.

```
public abstract class Shape {
public void printMe() {
System.out.println("I am a shape.");
}
public abstract double computeAreaQ;
}
public class Circle extends Shape {
private double rad = 5;
public void printMeQ {
System.out.println("I am a circle.");
}
public double computeAreaQ {
return rad * rad * 3.15;
}
}
public class Ambiguous extends Shape {
private double area = 10;
public double computeAreaQ {
return area;
}
}
public class IntroductionOverriding {
public static void main(String[] args) {
```

```
Shapef] shapes = new Shape[2];
Circle circle = new CircleQ;
Ambiguous ambiguous = new Ambiguous();
shapes[0] = circle;
shapes[l] = ambiguous;
for (Shape s : shapes) {
s.printMeQ;
System.out.println(s.computeArea());
}
}
}
```

Kode di atas akan mencetak:

```
1 I am a Circle
2 78.75
3 I am a shape.
4 10.0
```

Perhatikan bahwa Circle menimpa printMe(), sedangkan Ambiguous membiarkan metode ini apa adanya.

Kerangka Koleksi

Kerangka java collection sangat berguna, dan Anda akan melihatnya digunakan di seluruh buku ini. Berikut beberapa item yang paling berguna:

ArrayList:

ArrayList adalah larik yang berubah ukuran secara dinamis, yang tumbuh saat Anda menyisipkan elemen.

```
ArrayList<String> myArr = new ArrayList<String>();
myArr.add("one");
myArr.add("two");
System.out.println(myArr.get(0)); /* prints <one> */
```

Vektor:

Sebuah vektor sangat mirip dengan ArrayList, hanya saja ia disinkronkan. Sintaksnya juga hampir sama.

```
Vector<String> myVect = new Vector<String>();
myVect.add("one");
myVect.add("two");
System.out.println(myVect.get(0));
```

LinkedList:

LinkedList, tentu saja, adalah kelas LinkedList bawaan Java. Meskipun jarang muncul dalam wawancara, itu berguna untuk dipelajari karena menunjukkan beberapa sintaksis untuk iterator.

```
LinkedList<String> myLinkedList = new LinkedList<String>();
myLinkedList.add("two");
myLinkedList.addFirst("one");
Iterator<String> iter = myLinkedList.iteratorQ;
while (iter.hasNextQ) {
System.out.printIn(iter.next());
}
```

HashMap:

Koleksi HashMap banyak digunakan, baik dalam wawancara maupun di dunia nyata.

Kami telah menyediakan potongan sintaks di bawah ini.

```
HashMap<String, String> map = new HashMap<String, String>();
map.put("one", "uno");
map.put("two", "dos");
System.out.printIn(map.get("one"));
```

Sebelum wawancara Anda, pastikan Anda sangat nyaman dengan sintaks di atas. Anda akan membutuhkannya.

Pertanyaan Wawancara

Harap dicatat bahwa karena hampir semua solusi dalam buku ini diimplementasikan dengan Java, kami hanya memilih sejumlah kecil pertanyaan untuk bab ini. Selain itu, sebagian besar pertanyaan ini berhubungan dengan "hal-hal sepele" dari bahasa-bahasa tersebut, karena bagian lain dari buku ini diisi dengan pertanyaan tentang pemrograman Java.

Pertanyaan: 1 Dalam hal peinheritance, apa efek menjaga privasi konstruktor?

Jawaban: Mendeklarasikan private konstruktor akan memastikan bahwa tidak ada orang di luar kelas yang dapat langsung membuat instance kelas. Dalam kasus ini, satu-satunya cara untuk membuat instance kelas adalah dengan menyediakan metode publik statis, seperti yang dilakukan saat menggunakan Pola Metode Pabrik. Selain itu, karena konstruktor bersifat pribadi, kelas juga tidak dapat diwariskan.

Pertanyaan: 2 Di java, apakah blok last dieksekusi jika kita memasukkan pernyataan return di dalam blok try dari try-catch-akhirnya?

Jawaban: Ya, itu akan dieksekusi. Finally block dieksekusi ketika blok percobaan keluar. Bahkan ketika kami mencoba untuk keluar dalam blok percobaan (melalui pernyataan pengembalian, pernyataan lanjutan, pernyataan break atau pengecualian apa pun), finally block masih akan dieksekusi.

Perhatikan bahwa ada beberapa kasus di mana blok akhirnya tidak akan dijalankan, seperti berikut ini:

- Jika mesin virtual keluar selama eksekusi blok coba/tangkap.
- Jika utas yang menjalankan blok coba/tangkap terbunuh.

Pertanyaan: 3 Apa perbedaan antara final, finally, dan finalize?

Jawaban: Meskipun namanya terdengar mirip, final, finally, dan finalize memiliki tujuan yang sangat berbeda. Untuk berbicara dalam istilah yang sangat umum, final digunakan untuk mengontrol apakah variabel, metode, atau kelas "dapat diubah". Keyword terakhir digunakan dalam blok coba/tangkap untuk memastikan bahwa segmen kode selalu dieksekusi. Metode finalizeQ dipanggil oleh pengumpul sampah setelah menentukan bahwa tidak ada lagi referensi. Rincian lebih lanjut tentang keyword dan metode ini disediakan di bawah.

Final

Pernyataan terakhir memiliki arti yang berbeda tergantung pada konteksnya.
Jika diterapkan ke variabel (primitif): Nilai variabel tidak dapat berubah.
Jika diterapkan ke variabel (referensi): Variabel referensi tidak dapat menunjuk ke objek lain di heap.
Saat diterapkan ke suatu metode: Metode ini tidak dapat diganti.
Saat diterapkan ke kelas: Kelas tidak dapat dijadikan subkelas

Finally

Ada finally block opsional setelah try block atau setelah catch block. Pernyataan di finally block akan selalu dieksekusi (kecuali jika Java Virtual Machine keluar dari blok percobaan). Finally block digunakan untuk menulis kode pembersihan.

Finalized

Metode finalize() dipanggil oleh pengumpul sampah ketika itu menentukan bahwa tidak ada lagi referensi. Ini biasanya digunakan untuk membersihkan sumber daya, seperti menutup file.

Pertanyaan: 4 Jelaskan perbedaan antara template di C++ dan generik di Java.

Jawaban: Banyak pemrogram menganggap konsep template dan generik setara hanya karena keduanya memungkinkan Anda melakukan sesuatu di sepanjang baris List <St ringx Tapi, bagaimana setiap bahasa melakukannya, dan mengapa, sangat bervariasi.

Implementasi Java generik berakar pada ide "penghapusan tipe". Teknik ini menghilangkan tipe berparameter ketika kode sumber diterjemahkan ke kode byte Java Virtual Machine (JVM).

Misalnya, Anda memiliki kode Java di bawah ini:

```
Vector<string> vector = pew Vector<String>();
vector.add(new String("hello"));
String str = vector.get(0);
```

Selama kompilasi, kode ini ditulis ulang menjadi:

```
Vector vector = new Vector()j
vector.add(new String("hello"));
String str = (String) vector.get(0)j
```

Penggunaan Java generics tidak benar-benar mengubah banyak hal tentang kemampuan kami; itu hanya membuat segalanya menjadi sedikit lebih cantik. Karena alasan ini, obat generik Java terkadang disebut "gula sintaksis".

Ini sangat berbeda dari C++. Di C++, template pada dasarnya adalah kumpulan makro yang dimuliakan, dengan compiler membuat salinan baru dari kode template untuk setiap jenis. Buktinya adalah fakta bahwa instance MyClass <Foo> tidak akan berbagi variabel statis dengan MyClass <Bar>. Namun, dua instance MyClass <Foo> akan berbagi variabel statis.

Untuk mengilustrasikan hal ini, perhatikan kode di bawah ini:

```
/*** MyClass.h ***/
template<class T> class MyClass {
public:
static int valj
MyClass(int v) { val = v; }
};

/*** MyClass.cpp ***/
template<typename T>
int MyClass<T>::barj

template class MyClass<Foo>;
template class MyClass<Bar>;

/*** main.cpp ***/
 MyClass<Foo> * fool = new MyClass<Foo>(10);
MyClass<Foo> * foo2 = new MyClass<Foo>(15);
MyClass<Bar> * barl = new MyClass<Bar>(20);
MyClass<Bar> * bar2 = new MyClass<Bar>(35);

int fl = fool->val; // will equal 15
int f2 = foo2->val; // will equal 15
int bl = barl->val; // will equal 35
int b2 = bar2->val; // will equal 35
```

Di Java, variabel statis akan dibagikan di seluruh instance MyClass, apa pun jenis parameternya. Karena perbedaan arsitekturalnya, template Java generik dan C++ memiliki sejumlah perbedaan lain, termasuk:

Template C++ dapat menggunakan tipe primitif, seperti int. Java tidak bisa dan harus menggunakan Integer.

Di Java, Anda dapat membatasi parameter tipe template menjadi tipe tertentu. Misalnya, Anda mungkin menggunakan generik untuk mengimplementasikan CardDeck dan menentukan bahwa parameter type harus diperluas dari CardGame. Di C++, parameter type bisa dipakai, sedangkan Java tidak mendukung ini.

Di Java, parameter type (yaitu, Foo di MyClass <Foo>) tidak dapat digunakan untuk metode dan variabel statis, karena ini akan dibagi antara MyClass <Foo> dan MyClass <Bar>. Dalam C++, kelas-kelas ini berbeda, sehingga parameter tipe dapat digunakan untuk metode dan variabel statis.

Di Java, semua instance MyClass, apa pun parameter tipenya, adalah tipe yang sama. Parameter tipe dihapus saat runtime. Di C++, instance dengan parameter tipe yang berbeda adalah tipe yang berbeda.

Ingatlah bahwa meskipun generik Java dan template C++ terlihat sama dalam banyak hal, keduanya sangat berbeda.

Pertanyaan: 5 Jelaskan apa itu refleksi objek di Java dan mengapa itu berguna.

Jawaban: Refleksi Objek adalah fitur di Java yang menyediakan cara untuk mendapatkan informasi reflektif tentang kelas dan objek Java, dan melakukan operasi seperti:

- Mendapatkan informasi tentang metode dan kolom yang ada di dalam kelas saat runtime.
- Membuat instance baru dari kelas.
- Mendapatkan dan menyetel bidang objek secara langsung dengan mendapatkan referensi bidang, apa pun pengubah aksesnya.

  Kode di bawah ini menawarkan contoh refleksi objek.

```
/* Parameters */
Object[] doubleArgs = new Object[] { A.2, 3.9 };

/* Get class */
Class rectangleDefinition = Class.forName("MyProj.Rectangle");

/* Equivalent: Rectangle rectangle = new Rectangle(4.2, 3.9); */
Class[ ] doubleArgsClass = new Classf ] {double.class, double.class};
Constructor doubleArgsConstructor =
rectangleDefinition.getConstructor(doubleArgsClass);
Rectangle rectangle =
(Rectangle) doubleArgsConstructor.newInstance(doubleArgs);

/* Equivalent: Double area = rectangle.areaQ; */
Method m = rectangleDefinition.getDeclaredMethod("area");
Double area = (Double) m.invoke(rectangle);
Kode ini sama dengan:
Rectangle rectangle = new Rectangle(4.2, 3.9);
Double area = rectangle.areaQ;
```

Mengapa Refleksi Objek Berguna?

Tentu saja, ini tampaknya tidak terlalu berguna dalam contoh di atas, tetapi refleksi bisa sangat berguna dalam kasus-kasus tertentu. Refleksi objek berguna karena tiga alasan utama:

Ini membantu dalam mengamati atau memanipulasi perilaku runtime aplikasi.

Ini dapat membantu dalam men-debug atau menguji program, karena kami memiliki akses langsung ke metode, konstruktor, dan bidang.

Kita dapat memanggil metode dengan nama ketika kita tidak mengetahui metode tersebut sebelumnya. Misalnya, kita dapat membiarkan pengguna memasukkan nama kelas, parameter untuk konstruktor, dan nama metode. Kami kemudian dapat menggunakan informasi ini untuk membuat objek dan memanggil metode. Melakukan operasi ini tanpa refleksi akan membutuhkan serangkaian pernyataan jika yang kompleks, jika memungkinkan.

Pertanyaan: 6 Implementasikan kelas CircularArray yang mendukung struktur data seperti array yang dapat dirotasi secara efisien. Kelas harus menggunakan tipe generik, dan harus mendukung iterasi melalui standar untuk notasi (Obj o: CircularArray).

Jawaban: Masalah ini sebenarnya memiliki dua bagian. Pertama, kita perlu mengimplementasikan kelas CircularArray. Kedua, kita perlu mendukung iterasi. Kami akan membahas bagian ini secara terpisah.

Menerapkan kelas CircularArray

Salah satu cara untuk mengimplementasikan kelas CircularArray adalah dengan menggeser elemen setiap kali kita memanggil rotate (int shift Right). Melakukan ini, tentu saja, tidak terlalu efisien. Sebagai gantinya, kita bisa membuat kepala variabel anggota yang menunjuk ke apa yang seharusnya dilihat secara konseptual sebagai awal dari larik melingkar. Daripada berpindah-pindah elemen dalam array, kita hanya menaikkan head dengan shiftRight. Kode di bawah mengimplementasikan pendekatan ini.

```
public class CircularArray<T> {
private T[] items;
private int head = 0;

public CircularArray(int size) {
items = (T[]> new Object[size];
}

private int convert(int index) { 1
if (index < 0) {
index += items.length;
}
return (head + index) % items.length;
```

```
}

public void rotate(int shiftRight) {
head = convert(shiftRight); 18            }
}
public T get(int i) {
if (i < 0 | | i >= items.length) {
throw new java.lang.lndexOutOfBoundsException("...");
}
return items[convert(i)];
}
public void set(int i, T item) {
items[convert(i)] = item;
}
}
```

Ada beberapa hal di sini yang mudah membuat kesalahan, seperti:

Kami tidak dapat membuat array dengan tipe generik. Sebagai gantinya, kita harus mentransmisikan larik atau mendefinisikan item menjadi tipe Daftar <T>. Untuk kesederhanaan, kami telah melakukan yang pertama.

Operator% akan mengembalikan nilai negatif saat kita melakukan negValue% posVal. Misalnya, -8% 3 adalah -2. Ini berbeda dengan bagaimana ahli matematika mendefinisikan fungsi modulus.

Kita harus menambahkan item. panjang ke indeks negatif untuk mendapatkan hasil positif yang benar. Kita perlu memastikan untuk secara konsisten mengonversi indeks mentah ke indeks yang dirotasi. Untuk alasan ini, kami telah mengimplementasikan fungsi konversi yang digunakan oleh metode lain. Bahkan fungsi rotate menggunakan convert. Ini adalah contoh yang baik dari penggunaan ulang kode.

Sekarang setelah kita memiliki kode dasar untuk CircularArray, kita dapat fokus pada penerapan iterator.

Menerapkan Antarmuka Iterator

Bagian kedua dari pertanyaan ini meminta kita untuk mengimplementasikan kelas CircularArray sehingga kita dapat melakukan hal berikut:

```
CircularArray<String> array = ...
for (String s : array) { ... }
```

Menerapkan ini membutuhkan implementasi antarmuka Iterator. Untuk mengimplementasikan antarmuka Iterator, kita perlu melakukan hal berikut:

Ubah definisi CircularArray <T> untuk menambahkan implements Iterable <T>. Ini juga akan meminta kita untuk menambahkan metode iterator() ke CircularArray <T>.

Buat CircularArrayIterator <T> yang mengimplementasikan Iterator <T>. Ini juga akan mengharuskan kita untuk mengimplementasikan, di CircularArrayIterator, metode hasNextQ, next(), dan remove().

Setelah kita menyelesaikan item di atas, loop for akan "secara ajaib" bekerja.

Pada kode di bawah ini, kami telah menghapus aspek CircularArray yang identik dengan implementasi sebelumnya.

```
public class CircularArray<T> implements Iterable<T> {

public Iterator<T> iteratorQ {
return new CircularArrayIterator<T>(this);
}

private class CircularArrayIterator<TI> implements Iterator<TI>{
/* current reflects the offset from the rotated head, not
* from the actual start of the raw array. */
private int _current = -1;
private TI[] _items;

public CircularArrayIterator(CircularArray<TI> array){
items = array.items;
}

^Override
public boolean hasNextQ {
return _current < items.length - 1;
}
@0verride
public TI next( ) {
_current++;
TI item = (TI) _items[convert(_current)];
return item;
}
^Override
public void removeQ {
throw new UnsupportedOperationException("...");
}
}
}
```

Pada kode di atas, perhatikan bahwa iterasi pertama dari loop for akan memanggil hasNext() dan kemudian next(). Pastikan sangat yakin bahwa penerapan Anda akan mengembalikan nilai yang benar di sini.

Ketika Anda mendapatkan masalah seperti ini dalam sebuah wawancara, ada kemungkinan besar Anda tidak ingat persis apa yang disebut berbagai metode dan antarmuka. Dalam kasus ini, selesaikan masalahnya sebaik mungkin. Jika Anda dapat memikirkan metode macam apa yang mungkin dibutuhkan seseorang, itu saja akan menunjukkan tingkat kompetensi yang baik.

BAB 28
STANDAR PENGKODEAN JAVA IX
(STRANDAR CODING LANJUTAN UNTUK JAVA)

(Java.lang — Math Class, Strings, dan Wrapper)

Pengantar

Gunakan Standar Pengkodean Sun Java:

Bab ini dan bab selanjutnya akan membantu Anda memahami pedoman kode yang tepat. Pendamping ini sangat penting dan akan membantu Anda memecahkan Wawancara Pengembang Java yang diikuti oleh 10 perusahaan perangkat lunak teratas Dunia dan mempelajari standar pengkodean Java. Ada banyak masalah desain yang kompleks untuk dipertimbangkan, dan sejumlah teknologi Java tingkat lanjut harus dipahami dan diterapkan dengan benar.

Penilai praktik bekerja di bawah pedoman yang sangat ketat. Anda dapat membuat aplikasi paling brilian yang pernah ada untuk menghiasi JVM, tetapi jika Anda tidak memenuhi syarat dan nilai Anda, penilai tidak punya pilihan selain mengurangi poin penting (dan terkadang substansial) dari proyek Anda. Bab ini akan membantu Anda melewati batas dan menandai "i" Anda.

Mengikuti standar pengkodean tidaklah sulit; itu hanya membutuhkan ketekunan. Jika Anda berhati-hati, hal itu tidak perlu dipikirkan lagi, dan sayang kehilangan poin karena penjepit keriting di tempat yang salah.

Wawancara Pengembang menekankan hal-hal yang harus dilakukan untuk menghindari kegagalan otomatis. Wawancara menggunakan kata harus sering. Ketika kami menggunakan kata harus, kami menggunakannya dalam semangat wawancara, jika Anda harus melakukannya, jadi lanjutkan saja. Mari selami dunia Java Coding Standards yang menakjubkan.

Standar Jarak

Bagian bonus ini mencakup standar untuk indentasi, batas panjang garis, pemutusan garis, dan spasi. Mengindentasi Kami mengatakan ini akan menarik bukan? Setiap tingkat lekukan harus empat spasi, persis empat spasi, selalu empat spasi.

Tab harus diatur ke delapan spasi. Jika Anda berada di beberapa tingkat lekukan, Anda dapat menggunakan kombinasi tab dan (kumpulan empat) spasi untuk menyelesaikan lekukan yang benar.

Jadi jika Anda menggunakan metode dan Anda perlu memasukkan 12 spasi, Anda dapat menekan spasi 12 kali, atau tekan TAB sekali dan kemudian tekan spasi empat kali. (Pelatih perlambat.) Kami merekomendasikan untuk tidak menggunakan tombol TAB, dan tetap berpegang pada spasi — itu hanya sedikit lebih aman.

Kapan harus membuat indentasi jika Anda membuat indentasi seperti ini, Anda akan membuat penilai bangga:

Komentar awal, deklarasi paket, pernyataan import, deklarasi antarmuka, dan deklarasi kelas tidak boleh diindentasi.

Variabel statis, variabel instan, konstruktor, metode, dan komentar masing-masing * harus diindentasi satu tingkat.

di dalam konstruktor dan metode, variabel lokal, pernyataan, dan komentarnya harus diindentasi ke tingkat lain.

Pernyataan (dan komentarnya) dalam pernyataan blok harus diindentasi pada level lain untuk setiap level bersarang yang terlibat. (Jangan khawatir; kami akan memberi Anda contoh.)

```
public class Indent {

    static int staticVar = 7;

    public Indent() { }

    public static void main(String [] args) {

        int x = 0;

        for(int z=0; z<7; z++) {
            x = x + z;
            if (x < 4) {
                x++;
            }
        }
    }
}
```

Panjang Garis dan Pembungkus Garis

Aturan umumnya adalah satu baris tidak boleh lebih dari 80 karakter. Kami merekomendasikan 65 karakter hanya untuk memastikan bahwa berbagai macam editor akan menangani kode Anda dengan baik. Ketika satu baris kode lebih panjang dari yang akan muat pada satu baris, ada beberapa pedoman pembungkusan baris yang harus diikuti. Kami tidak dapat mengatakan dengan pasti bahwa ini adalah suatu keharusan, tetapi jika Anda mengikuti pedoman ini, Anda dapat yakin bahwa Anda berada di tempat yang aman:

- Istirahat setelah koma.
- Istirahat di depan operator.
- Sejajarkan baris baru satu tab (atau delapan spasi) di luar awal baris yang putus.
- Cobalah untuk tidak membobol ekspresi dalam tanda kurung. (Tunggu, contohnya akan datang.)

Cuplikan berikut menunjukkan penggabungan baris yang dapat diterima:

```
/* example of a line wrap */
System.out.println(((x * 42) + (z - 343) + (x % z ))
        + numberOfParsecs);

/* example of a line wrap for a method */
x = doStuffWithLotsOfArgs(coolStaticVar, instanceVar,
        numberOfParsecs, reallyLongShortName, x, z);
```

White space

Percayakah Anda bahwa kami harus membahas tingkat detail ini? Ternyata jika Anda tidak membagi ruang kosong Anda seperti yang dikatakan standar, Anda bisa kehilangan poin. Dengan pemikiran bahagia itu, mari kita bahas penggunaan yang tepat dari baris kosong dan pernyataan kosong.

Penggunaan Garis Kosong dengan Benar

Baris kosong digunakan untuk membantu pembaca kode Anda (yang mungkin adalah Anda, berbulan-bulan setelah Anda menulisnya) untuk dengan mudah menemukan blok logika dalam file sumber Anda. Jika Anda mengikuti rekomendasi ini di file sumber Anda, kekhawatiran baris kosong Anda akan berakhir.

Gunakan baris kosong,

- Antara metode dan konstruktor.
- Setelah variabel contoh terakhir Anda.
- Di dalam metode antara variabel lokal dan pernyataan pertama.
- Di dalam metode untuk memisahkan segmen kode logis.
- Sebelum satu baris atau blokir komentar.
- Gunakan dua baris kosong antara bagian utama file sumber: paket, pernyataan import, kelas, dan antarmuka.
- Penggunaan Ruang Kosong dengan Benar
- Spasi kosong digunakan untuk membuat pernyataan lebih mudah dibaca, dan tidak terlalu rapat.

Gunakan ruang kosong,

- Antara operator biner.
- Setelah koma dalam daftar argumen.
- Setelah ekspresi di pernyataan for.
- Antara keyword dan tanda kurung. Setelah gips.

Contoh kode berikut menunjukkan bentuk yang tepat untuk digunakan saat membuat indentasi, melompati baris, membungkus baris, dan menggunakan spasi. Kami belum membahas semua aturan yang terkait dengan penggunaan komentar yang tepat; oleh karena itu, sampel ini tidak menunjukkan komentar standar:

```
/*
 * This listing demonstrates only proper spacing standards
 *
 * The Javadoc comments will be discussed in a later chapter
 */

package com.wickedlysmart.utilities;

import java.util.*;

/**
 * CoolClass description
 *
 * @version .97 10 Oct 2002
 * @author  Joe Beets
 */
public class CoolClass {

    /** Javadoc static var comment */
    public static int coolStaticVar;

    /** Javadoc public i-var comment */

      public long instanceVar;

      /* private i-var comment */
      private short reallyLongShortName;

      /** Javadoc constructor comment */
      public CoolClass() {
        // do stuff
      }

      /** Javadoc comment about method */
      void coolMethod() {
          int x = 0;
          long numberOfParsecs = 0;

          /* comment about for loop */
          for(z = 0; z < 7; z++) {
              x = x + z;
```

```
            /* comment about if test */
            if (x < 4) {
                x++;
            }

            /* example of a line wrap */
            System.out.println(((x * 42) + (z - 343) + (x % z ))
                    + numberOfParsecs);

            /* example of a line wrap for a method */
            x = doStuffWithLotsOfArgs(coolStaticVar, instanceVar,
                    numberOfParsecs, reallyLongShortName, x, z);
        }
    }

    /** Javadoc comment about method */
    int doStuffWithLotsOfArgs(int a, long b, long c, short d, int e,
            int f) {
        return e * f;
    }
}
```

Deklarasi Itu Menyenangkan

Deklarasi adalah bagian besar dari Java. Mereka juga kompleks, sarat dengan aturan, dan jika digunakan sembarangan dapat menyebabkan bug dan kemampuan pemeliharaan yang buruk. Rangkaian pedoman berikut dimaksudkan untuk membuat kode Anda lebih mudah dibaca, lebih dapat di-debug, dan lebih mudah dipelihara.

Mengurutkan Deklarasi Anda

Elemen-elemen dalam file sumber Java Anda harus diatur dalam urutan standar. Dalam beberapa kasus, kompilator menuntutnya, dan untuk kasus lainnya, konsistensi akan membantu Anda mendapatkan teman dan memengaruhi orang. Ini dia:

- komentar kelas.
- deklarasi paket.
- pernyataan impor.
- deklarasi kelas.
- variabel statis.
- variabel instan.
- konstruktor.

Lokasi dan Inisialisasi

Panduan berikut harus dipertimbangkan saat membuat deklarasi Java: Di dalam metode:

Deklarasikan dan inisialisasi variabel lokal sebelum pernyataan lain (jika memungkinkan).

Deklarasikan dan inisialisasi variabel blok sebelum pernyataan blok lainnya (bila memungkinkan). Nyatakan hanya satu anggota per baris.

Hindari variabel bayangan. Ini terjadi ketika variabel instan memiliki nama yang sama dengan variabel lokal atau blok. Sementara kompiler akan mengizinkannya, shadowing dianggap sangat tidak ramah terhadap rekan kerja berikutnya (ingat: berpotensi psikopat) yang harus menjaga kodemu.

Kapitalisasi

Tiga tebakan. Sebaiknya Anda menggunakan kapitalisasi dengan benar saat mendeklarasikan dan menggunakan nama paket, kelas, antarmuka, metode, variabel, dan konstanta. Aturannya cukup sederhana:

- Nama paket Taruhan teraman adalah menggunakan huruf kecil jika memungkinkan: com.wickedlysmart.utilities
- Nama Kelas dan Antarmuka Biasanya mereka harus kata benda; kapitalisasi huruf pertama dan huruf pertama lainnya di kata kedua di dalam nama: Customer atau CustomTable
- Nama metode Biasanya harus berupa kata kerja; kata pertama harus huruf kecil, dan jika ada kata kedua, huruf pertama masing-masing harus menggunakan huruf besar:
- menginisialisasi(); atau getTelescopicOrientation();
- Nama variabel Mereka harus mengikuti aturan kapitalisasi yang sama sebagai metode; Anda harus memulainya dengan huruf (meskipun Anda dapat menggunakan _ atau $, jangan), dan hanya variabel sementara seperti variabel perulangan yang harus menggunakan nama karakter tunggal:
- currentIndex; atau nama; atau x;
- Nama konstanta Untuk diberi label konstanta, variabel harus dinyatakan statis dan final. Nama mereka harus semua huruf besar dan garis bawah harus digunakan untuk memisahkan kata: MAX HEIGHT; atau USED;

Kejelasan Dan Pemeliharaan Kode Java

Pengenalan

Kejelasan dan Pemeliharaan

Selalu Tulis Kode yang Jelas dan Mudah Dipelihara Sekarang setelah Anda membuat kode Anda dapat dibaca, apakah kode Anda yang mudah dibaca benar-benar masuk akal? Apakah bisa dengan mudah dirawat?

Ini adalah masalah besar untuk wawancara, bernilai sebagian besar dari skor penilaian Anda. Kami akan melihat semuanya mulai dari desain kelas hingga penanganan kesalahan. Ingatlah bahwa Anda adalah seorang Pemain Tim. Beberapa area utama kejelasan kode dibahas secara lebih mendetail di bab Dokumentasi, jadi kami tidak akan membahasnya di sini.

Area tersebut mencakup pentingnya komentar yang bermakna dan pengenal dokumentasi mandiri. Masalah yang diangkat dalam bab ini adalah

- Pertimbangan gaya pemrograman umum.
- Mengikuti prinsip desain OO.
- Menemukan kembali roda. Penanganan kesalahan.
- Pertimbangan Pemrograman Umum
- Konvensi pengkodean yang dibahas di bab sebelumnya adalah titik awal yang bagus. Tetapi wawancara juga mencari konsistensi dan kesesuaian dengan gaya pemrograman Anda.

Bagian berikut mencantumkan beberapa poin penting yang harus Anda ingat saat menulis kode yang diformat sempurna. Beberapa di antaranya akan dijelaskan di bagian selanjutnya; beberapa dari poin ini terkait dengan desain OO, misalnya, dan kami membahasnya lebih detail di bagian itu.

Sekali lagi, ini bukan waktunya untuk memperdebatkan manfaat sebenarnya dari prinsip-prinsip ini. Sekali lagi, bayangkan Anda telah menjadi tim proyek dan perlu membuktikan diri Anda sebagai, apa? Iya! Pemain Tim Man!.

Hal pertama yang dicari tim adalah apakah Anda dapat mengikuti konvensi dan standar sehingga setiap orang dapat bekerja sama tanpa ingin saling melempar keluar jendela lantai tujuh dan ke air mancur semen di bawah. (Kecuali jika Anda adalah perusahaan dot-com dan kantor Anda sekarang mengawasi pompa bensin yang ditinggalkan.)

Poin-poin ini tidak dalam urutan tertentu, jadi jangan menyimpulkan bahwa poin pertama lebih penting daripada yang terakhir. Namun, Anda dapat menyimpulkan bahwa penilai wawancara Anda mungkin akan menanyakan apakah Anda telah melakukan hal-hal ini dengan tepat.

Pertahankan Ruang Lingkup Variabel Sekecil Mungkin

Jangan menggunakan variabel instance saat variabel lokal akan berfungsi! Hal ini tidak hanya memengaruhi penggunaan memori, tetapi juga mengurangi risiko objek "tergelincir" ke suatu tempat sehingga tidak boleh digunakan, baik secara tidak sengaja atau sengaja.

Tunggu untuk mendeklarasikan variabel sampai sebelum digunakan. Dan Anda harus selalu menginisialisasi variabel lokal pada saat ia dideklarasikan (yang tepat sebelum digunakan), dengan pengecualian blok try/catch. Dalam hal ini, jika variabel dideklarasikan dan ditugaskan di blok coba/tangkap, kompilator tidak akan membiarkan Anda menggunakannya di luar blok itu, jadi jika Anda memerlukan variabel setelah blok coba atau tangkap, maka Anda harus menyatakan

Ini pertama di luar coba/tangkap. Cara lain untuk mengurangi ruang lingkup adalah dengan menggunakan perulangan for daripada while.

Hindari Mendesain Kelas yang Tidak Memiliki Metode

Objek dimaksudkan untuk memiliki status dan perilaku; mereka bukan hanya struct yang dimuliakan. Jika Anda membutuhkan struktur data, gunakan Koleksi. Ada pengecualian untuk ini, Namun, itu mungkin berlaku untuk tugas wawancara Anda. Terkadang Anda memang membutuhkan objek yang tujuan utamanya adalah membawa data dari satu lokasi ke lokasi lain — biasanya sebagai hasil dari permintaan database.

Sebuah baris dalam tabel, misalnya, harus direpresentasikan sebagai objek dalam program Java Anda, dan mungkin tidak selalu memerlukan metode jika tugas utamanya adalah, katakanlah, ditampilkan dalam tabel GUI. Ini dikenal sebagai pola ValueObject. Yang membawa kita ke edisi berikutnya.

Gunakan Pola Desain

Saat Anda menggunakan pola yang sudah dikenal, Anda memiliki semacam cara singkat untuk mendiskusikan desain Anda dengan pemrogram lain (meskipun diskusi itu terjadi antara kode/komentar Anda dan orang lain.

Jika Anda melakukannya dengan benar, Anda tidak akan secara pribadi berada di sana untuk membicarakannya, seperti dalam kasus wawancara Pengembang). Jika Anda membutuhkan Singleton, buatlah Singleton — jangan hanya mendokumentasikan bahwa hanya ada satu dari hal-hal ini.

Di sisi lain, jangan memaksakan desain Anda menjadi pola hanya untuk menggunakan pola. Kesederhanaan harus menjadi perhatian pertama Anda, tetapi jika itu adalah kekacauan antara pendekatan Anda dan pola desain terkenal yang sama rumitnya, pilih polanya

Kurangi Visibilitas Sesuatu Sebisa Mungkin

Secara umum, semakin banyak hal publik yang Anda tunjukkan kepada dunia, semakin sedikit kebebasan Anda untuk membuat perubahan nanti tanpa melanggar kode orang lain. Semakin sedikit Anda mengekspos, semakin banyak fleksibilitas yang Anda miliki untuk perubahan implementasi nanti.

Dan Anda tahu selalu ada perubahan. Jadi, membuat variabel, metode, dan kelas sekecil mungkin sambil membatasi apa yang Anda tunjukkan ke "antarmuka publik" Anda,

Anda akan bugar di jalan. Jelas ada masalah halus lainnya tentang inheritance (seperti dalam, apa yang subclass dapatkan aksesnya?), Jadi ada lebih banyak yang perlu dipertimbangkan di sini, tetapi secara umum, pikirkan untuk mengurangi eksposur Anda (anggap itu sebagai mengurangi tanggung jawab Anda di jalan). Hal ini terkait erat dengan pengurangan ruang lingkup variabel.

Gunakan Overloading Daripada Logika

Jika Anda memiliki metode yang perlu berperilaku berbeda bergantung pada jenis hal yang sebenarnya diberikan, pertimbangkan untuk membebani secara berlebihan.

Setiap kali Anda melihat apakah atau beralih blok yang menguji tipe argumen, Anda mungkin harus mulai berpikir untuk membebani metode ini. Dan saat Anda melakukannya…

Hindari Daftar Argumen Panjang

Jika Anda memiliki banyak argumen yang masuk ke suatu metode, mungkin Anda perlu merangkum hal-hal yang Anda butuhkan dalam metode itu ke dalam kelas dengan tipenya sendiri.

Jangan Panggil yang Berpotensi Dapat Diganti

Metode dari Pembuat Anda sudah tahu bahwa Anda tidak dapat mengakses hal-hal nonstatis sebelum superkonstruktor Anda berjalan, namun perlu diingat bahwa bahkan setelah superkonstruktor objek selesai, objek tersebut masih dalam keadaan tidak lengkap hingga konstruktornya selesai. Polimorfisme masih berfungsi di konstruktor.

Jadi jika B memanjang A, dan A memanggil metode dalam konstruktornya yang telah diganti oleh B, coba tebak apa yang terjadi ketika seseorang membuat turunan dari B. Anda mendapatkannya. Konstruktor B memanggil superkonstruktornya (konstruktor A). Tetapi di dalam konstruktor A, ia memanggil salah satu metodenya sendiri, tetapi B telah menimpa metode itu. Metode B berjalan!

Dengan kata lain, sebuah objek dapat memiliki salah satu metodenya dipanggil bahkan sebelum konstruktornya selesai! Jadi, meskipun B bukan objek yang sepenuhnya terbentuk, B masih dapat menjalankan kode dan bahkan mengakses variabel instance-nya sendiri.

Ini menjadi masalah karena variabel instance-nya belum diinisialisasi ke selain nilai default, meskipun variabel tersebut diberi nilai eksplisit saat dideklarasikan. Astaga! Jadi jangan lakukan itu. Jika ini adalah metode instance final atau pribadi, Anda aman karena Anda tahu itu tidak akan pernah diganti.

Kode ke Antarmuka

Polimorfisme, polimorfisme, polimorfisme. Gunakan argumen polimorfik, tipe kembalian, dan variabel bila memungkinkan (dengan kata lain, nyatakan variabel, tipe kembalian, atau argumen sebagai tipe antarmuka daripada tipe kelas tertentu).

Menggunakan antarmuka sebagai tipe memungkinkan Anda mengekspos hanya definisi dari apa yang dapat dilakukan kode Anda, dan membiarkan penerapannya fleksibel dan dapat diperluas. Dan bisa dipelihara. Dan semua hal baik OO lainnya-yang-berakhir-dengan-ble. Tetapi jika Anda tidak bisa…

Gunakan Kelas Abstrak Saat Anda Membutuhkan Fungsi untuk Diwarisi

Jika Anda benar-benar harus memiliki kode implementasi dan/atau variabel instan, gunakan kelas abstrak dan gunakan kelas itu sebagai variabel polimorfik yang dideklarasikan, argumen, dan tipe kembalian.

Buat Objek yang Anda Selesaikan

Dengan Memenuhi Syarat untuk Garbage Collection Anda sudah tahu bagaimana melakukan ini. Baik secara eksplisit menetapkan variabel referensi ke null saat Anda tidak lagi menggunakan objek, atau menetapkan ulang objek yang berbeda ke variabel referensi tersebut (sehingga mengabaikan objek yang awalnya direferensikan olehnya). Pada waktu bersamaan…

Jangan Membuat Lebih Banyak Objek Dari Yang Anda Butuhkan

Hanya karena ada pengumpul sampah bukan berarti Anda tidak akan memiliki "masalah memori". Jika Anda menyimpan terlalu banyak objek di heap, tidak memenuhi syarat untuk Garbage Collection (tetapi tidak, setelah membaca poin sebelumnya), Anda masih bisa kehabisan memori.

Namun, kemungkinan besar hanya masalah bahwa kinerja Anda mungkin sedikit menurun oleh overhead dari pembuatan semua objek tersebut dan kemudian meminta pengumpul sampah mengambilnya kembali.

Jangan melakukan apa pun untuk mengubah desain Anda hanya untuk mencukur beberapa objek, tetapi perhatikan kode penerapan Anda. Dalam beberapa kasus, Anda mungkin dapat menggunakan kembali objek yang ada dengan menyetel ulang statusnya.

Hindari Logika kompleks dan depply Nested

Lebih sedikit lebih baik dalam hal percabangan. Faktanya, penilai Anda mungkin menerapkan ukuran Cyclomatic Complexity ke kode Anda, yang menganggap kode menjadi kompleks bukan berdasarkan baris kode, tetapi lebih pada berapa banyak titik cabang yang ada. (Sebenarnya jauh lebih kompleks dari itu.

Ironisnya, pengujian untuk kompleksitas kode itu sendiri merupakan rumus yang agak rumit.) Intinya adalah, setiap kali Anda melihat jika atau sesuatu yang bersarang selain aliran logika yang sangat sederhana dalam suatu metode, Anda harus secara serius mempertimbangkan untuk mendesain ulang metode tersebut atau memisahkan fungsionalitas menjadi terpisah. metode.

Gunakan Getters dan Setter That Follow

Konvensi Penamaan JavaBean Itu berarti Anda harus menggunakan set <yourPropertyName> untuk metode yang bisa memodifikasi properti (biasanya properti dipetakan langsung ke variabel instance, tetapi tidak harus) dan mendapatkan <yourPropertyName> untuk metode yang bisa membaca properti.

Misalnya, nama variabel String akan memiliki metode pengambil/penyetel berikut:

setName(String name)

String getName()

Jika propertinya adalah boolean, maka Anda punya pilihan (ya, Anda sebenarnya punya pilihan) apakah akan memanggil metode baca get <property> atau <property>. Misalnya, variabel instance boolean motorOn dapat memiliki metode pengambil/penyetel berikut:

setMotorOn(boolean state)

boolean getMotorOn()

boolean isMotorOn()

Keindahan mengikuti konvensi penamaan JavaBeans adalah, hei, Anda harus menamainya sesuatu dan jika Anda tetap berpegang pada konvensi tersebut, maka sebagian besar alat yang terkait dengan Java (dan beberapa teknologi) dapat membaca kode Anda dan secara otomatis mendeteksi bahwa Anda memiliki yang dapat diedit properti, misalnya. Itu keren; kamu harus melakukannya.

Jangan Menjadi Pemrogram Prosedural di Dunia OO

Dua hadiah mati bahwa Anda belum benar-benar melakukan transisi ke objek "makhluk" yang lengkap adalah ketika Anda menggunakan yang berikut:

Kelas Sangat Besar yang memiliki metode untuk semuanya.

Banyak metode statis. Faktanya, semua metode harus nonstatic kecuali Anda memiliki alasan yang benar-benar bagus untuk membuatnya statis. Ini OO. Kami tidak memiliki variabel dan fungsi global.

Tidak ada "mulai di sini dan kemudian terus mengeksekusi secara linier kecuali saat Anda bercabang, tentu saja ...". Ini adalah OO, dan itu berarti objek sepenuhnya turun.

Jadikan Variabel dan Metode Penjelasan Sendiri Sebisa Mungkin

Jangan gunakan nama variabel seperti x dan y. Apa sih artinya ini: int x = 27; 27 apa? Kecuali jika Anda benar-benar merasa dapat mengunci keamanan pekerjaan dengan memastikan tidak ada yang dapat memahami kode Anda (dan dengan asumsi pelaku pembunuhan yang mencoba tidak akan menemukan Anda), maka Anda harus membuat pengenal Anda semaksimal mungkin.

Tidak harus berupa paragraf. Faktanya, jika dibutuhkan satu paragraf untuk menjelaskan apa yang diwakili oleh sebuah variabel, mungkin Anda perlu memikirkan desain Anda lagi. Atau setidaknya, gunakan komentar. Tapi jangan membuatnya tegas! Ambil pelajaran dari API inti. Mereka bisa saja memanggil ArInBException, tetapi sebaliknya mereka memanggilnya

ArrayIndexOutOfBoundsException. Apakah ada pertanyaan tentang apa yang diwakili oleh pengecualian itu? Tentu saja, kesalahan besar Sun adalah NullPointerException yang terkenal. Tetapi meskipun menggunakan penunjuk kata

terlarang, semua orang tahu apa artinya ketika mereka mendapatkannya. Tetapi mungkin ada beberapa kebingungan jika itu disebut NPTException atau bahkan NullException.

Gunakan core API!

Jangan menciptakan kembali roda, dan jangan — atau kamu akan otomatis gagal untuk penggunaan tertentu pustaka apa pun selain kode yang kamu kembangkan dan API Java inti. Tahan godaan apa pun untuk berpikir bahwa kamu dapat membangun sesuatu dengan lebih cepat, lebih bersih, lebih efisien, dll.

Bahkan jika itu benar, tidak ada gunanya melepaskan manfaat menggunakan kelas standar yang sudah dikenal orang lain, dan yang telah sangat, sangat diuji di lapangan.

Buat Kelas Pengecualianmu Sendiri Jika kamu Tidak Dapat Menemukan Yang Sesuai Dengan Kebutuhan mu

Jika tidak ada kelas Exception yang dicentang yang sempurna untuk dirimu di java.lang, buat kelasmu sendiri. Dan buat itu cukup spesifik agar bermakna bagi penangkapnya. Dengan kata lain, jangan membuat BadThingHappenedException dan membuangnya untuk setiap kemungkinan kesalahan bisnis yang terjadi dalam programmu.

Jangan Kembalikan Kode Kesalahan!

Ini Jawa. Ini OO. Jika kamu benar-benar perlu menunjukkan kondisi luar biasa, gunakan Exception! Jika kamu benar-benar ingin mengganggu penilai, gunakan kode kesalahan sebagai nilai kembalian dari beberapa metodemu. Bahkan satu metode mungkin berhasil.

Buat Pengecualian Anda dengan Argumen Pembuat String

Melakukannya memberi kamu kesempatan untuk mengatakan lebih banyak tentang apa yang terjadi sehingga menyebabkan pengecualian. Saat kamu membuat instance Exception, panggil konstruktor yang mengambil String (atau konstruktor yang mengambil pengecualian tingkat rendah lainnya jika kamu melakukan rangkaian pengecualian). Saat kamu membuat kelas Exception Anda sendiri, pastikan untuk meletakkan konstruktor yang menggunakan String.

Ikuti Prinsip Desain OO Dasar

Di bagian sebelumnya, beberapa poin utama telah disinggung di area yang akan kita gali lebih dalam di sini. Kamu tidak harus menjadi Desainer OO Terbaik Dunia, tetapi kamu harus mengikuti prinsip dasar yang di atasnya manfaat OO bergantung.

Jelas kita tidak bisa menjadikan ini sebagai "Cara Menjadi Desainer OO yang Baik dalam 10 Halaman Mudah". Kamu perlu lebih banyak belajar dan berlatih, yang kami anggap sudah kamu lakukan.

Ini seharusnya menjadi berita lama sekarang, tetapi kamu dapat bertaruh bahwa penilaimu akan mempelajari masalah ini, jadi penyegar tidak akan merugikan. Kami mencapai sorotan tentang area di mana kamu mungkin mendapatkan pengurangan poin dari tugasmu.

Sembunyikan Detail Implementasi

Ini berlaku di banyak tempat, tetapi pengkodean dengan antarmuka dan menggunakan enkapsulasi adalah cara terbaik untuk melakukannya. Jika kamu menganggap kodemu sebagai komponen kecil yang berdiri sendiri dan dapat dicolokkan, kamu tidak ingin siapa pun yang menggunakan salah satu komponenmu memikirkan cara kerjanya. Itu semua bermuara pada input dan output.

Antarmuka publik menjelaskan apa yang dibutuhkan metode dari kamu, dan apa yang akan dikembalikan kepada dirimu. Tidak ada penjelasan tentang bagaimana hal itu dicapai. Kamu bisa mengubah implementasimu (bahkan kelas yang melakukan implementasi) tanpa memengaruhi kode panggilan.

Detail implementasi juga dapat disebarkan melalui pengecualian, jadi berhati-hatilah agar kamu tidak menggunakan antarmuka, tetapi kemudian letakkan pengecualian khusus penerapan di klausa lemparan! Jika klien melakukan "penelusuran", mereka tidak harus menangkap SQLException,

Sebagai contoh. Jika kode implementasimu kebetulan melakukan pekerjaan database yang dapat menghasilkan SQLExceptions (seperti kode JDBC), klien tidak perlu mengetahuinya. Tugas Anda adalah menangkap pengecualian khusus penerapan tersebut dan mengembalikan sesuatu yang lebih bermakna pengecualian khusus bisnis ke kode klien.

Gunakan Perincian Kelas yang Sesuai

Kelas harus benar, kamu tahu, perincian. Tidak boleh terlalu besar atau terlalu kecil. Jarang terjadi masalah di kelas yang terlalu kecil; namun, sebagian besar programmer yang tidak terlalu OO membuat kelas yang terlalu besar.

Suatu kelas seharusnya mewakili suatu hal yang memiliki keadaan dan perilaku. Teruslah bertanya pada diri sendiri, saat kamu menulis setiap metode, apakah perilaku itu mungkin tidak lebih cocok untuk beberapa hal lain.

Misalnya, kamu memiliki kelas Dapur yang melakukan segala macam hal tentang Dapur. Seperti peralatan Oven dan lemari es, dll. Jadi sekarang kamu memiliki peralatan Dapur (Dapur menjadi ruangan) dan peralatan Kulkas dan peralatan Oven semuanya dalam kelas yang sama.

Itu tiga hal yang berbeda. Kelas (dan dengan demikian objek yang dipakai darinya) harus benar-benar spesialis. Mereka harus melakukan jenis perilaku yang harus dilakukan oleh jenis itu, dan tidak lebih. Jadi, daripada meminta kelas Dapur menyertakan semua kode untuk perilaku Kulkas dan Oven, minta kelas Dapur menggunakan Kulkas dan Oven dalam hubungan HAS-A.

Ini membuat ketiga kelas tetap sederhana, dan dapat digunakan kembali. Dan itu memecahkan masalah penamaanmu, sehingga kamu tidak perlu menamai dapur serba guna, KitchenFridgeOven.

Kemungkinan penyebab Kelas Besar lainnya adalah kamu terlalu banyak menentukan kelas batin. Terlalu banyak arti beberapa inner class seharusnya merupakan kelas tingkat atas (untuk digunakan kembali) atau hanya metode dari kelas penutup. Pastikan kelas batin atau bertingkat Anda benar-benar perlu disertakan.

Batasi Subclass

Jika kamu perlu membuat subclass baru untuk menambahkan fungsionalitas penting, mungkin fungsionalitas tersebut harus benar-benar ada di kelas induk (sehingga menghilangkan kebutuhan untuk subclass — kamu hanya perlu memperbaiki superclass).

Saat kamu merasa perlu untuk memperluas kelas, selalu lihat apakah kelas induk harus berubah, atau apakah kamu memerlukan komposisi (yang berarti menggunakan hubungan HAS-A daripada IS-A).

Lihat di API Java inti untuk mendapatkan petunjuk tentang subkelas versus komposisi: hierarki peinheritance API inti sangat luas tetapi sangat dangkal. Dengan beberapa pengecualian (seperti komponen GUI), sebagian besar hierarki kelas tidak lebih dari dua hingga tiga level.

Gunakan Granularitas Metode yang Tepat

Sama seperti kelas harus menjadi spesialis, begitu pula metode. Kamu hampir pasti akan mendapatkan poin berlabuh untuk tugasmu jika metode kamu panjang (meskipun dalam beberapa kasus, terutama dalam kode GUI Swing kamu, metode yang panjang tidak selalu merupakan cerminan dari desain yang buruk).

Namun, dalam banyak kasus, semakin lama metode tersebut semakin kompleks, karena seringkali metode yang panjang merupakan cerminan dari metode yang melakukan terlalu banyak pekerjaan.

Kamu semua adalah pemrogram, jadi kami tidak perlu menekankan pentingnya fungsi modular yang lebih kecil sehingga lebih mudah untuk melakukan debug, memodifikasi, menggunakan kembali, dll. Selalu lihat apakah masuk akal untuk memecah metode yang lebih panjang menjadi yang lebih kecil. Tetapi sementara dalam krisis tenggat waktu kamu mungkin lolos dengan metode panjang di dunia nyata (merasa bersalah tentu saja), itu tidak akan berhasil untuk tugas Pengembangmu.

Gunakan Enkapsulasi

Tugasmu akan diteliti untuk prinsip OO yang paling mendasar ini. Harapkan penilai untuk melihat cara Anda mengontrol akses ke status objekmu.

Dengan kata lain, cara kamu melindungi variabel instance dengan penyetel dan pengambil. Tidak perlu membahasnya di sini, lakukan saja. Izinkan akses ke data

mu(kecuali konstanta, tentu saja) hanya melalui metode yang lebih dapat diakses. Berhati-hatilah dengan pengubah aksesmu.

Memiliki sekumpulan metode pengakses yang bagus tidak masalah jika kamu membiarkan variabelmu terbuka lebar untuk akses langsung. Sekali lagi, jadikan hal-hal pribadi dan ruang lingkup sebisa mu.

Pisahkan Kode yang Mungkin Berubah dari Kode yang Tidak Perlu

Saat kamu mendesain kelasmu, pastikan untuk memisahkan fungsionalitas yang mungkin berubah menjadi kelas terpisah. Dengan demikian, kamu membatasi tempat-tempat yang harus kamu lacak dan membuat modifikasi seiring dengan berkembangnya program.

Gunakan Core API

Selalu selalu periksa API inti, dan ketahuilah bahwa terkadang kamu mungkin menemukan kelas yang kamu cari dalam paket selain yang kamu harapkan.

Jadi pastikan untuk benar-benar mencari melalui API, bahkan menggali paket dan kelas yang mungkin kamu anggap agak menyimpang. Terkadang solusi bisa berada di tempat yang tidak kamu duga, jadi tetap terbuka untuk pendekatan yang belum tentu seperti yang biasanya kamu lakukan.

Membolak-balik buku API referensi dapat membantu. Sebuah metode mungkin menarik perhatianmu dan bahkan jika ternyata bukan solusimu, itu mungkin memicu ide tentang solusi yang berbeda.

Dalam beberapa kasus, kamu mungkin tidak menemukan apa yang kamu cari, tetapi kamu mungkin menemukan kelas yang dapat diperluas, sehingga mewarisi sekumpulan fungsionalitas yang sekarang tidak perlu kamu tulis dan uji (tunduk pada peringatan tentang subclassing). kami sebutkan sebelumnya).

Menggunakan API inti (selain penting untuk ujian) memungkinkan kamu memanfaatkan banyak keahlian dan pengujian, plus kamu menggunakan kode yang sudah dikenal oleh ratusan ribu pengembang Java lainnya.

Gunakan Pola Desain Standar

Kami tidak dapat memberi tahu kamu mana yang benar-benar kamu perlukan untuk tugasmu; itu tergantung pada tugas dan pendekatan khusus dirimu. Namun ada banyak pola desain standar yang memungkinkan kamu memanfaatkan pengalaman kolektif semua orang yang pernah berjuang dengan masalah kamu sebelum dirimu (meskipun biasanya pada tingkat yang cukup abstrak — biasanya di situlah sebagian besar pola melakukan tugasnya).

Jadi, meskipun API inti memungkinkan kamu memanfaatkan kode implementasi orang lain, pola desain memungkinkan kamu memanfaatkan pendekatan orang lain terhadap suatu masalah. Namun, jika dirimu menaruh senjata di kepala kami, kami mungkin harus mengatakan bahwa Singleton harus berada di atas daftar hal-hal yang perlu Anda pertimbangkan saat mengembangkan tugasmu.

Tetapi kamu juga dapat melihat MVC (untuk GUI klien Anda), Façade, Decorator, Observer, Command, Adapter, Proxy, dan Callback, sebagai permulaan. Ambil buku tentang pola desain (referensi klasik dikenal sebagai buku "Gang of Four" (GOF),

Pola Desain: Elemen Perangkat Lunak Berorientasi Objek yang Dapat Digunakan Kembali, oleh Erich Gamma, Richard Helm, Ralph Johnson, dan John Vlissides) dan luangkan waktu untuk mundur dan melihat di mana programmu mungkin mencoba melakukan sesuatu yang diselesaikan dengan baik oleh pola desain .

Pola tersebut tidak memberi tahumu cara menyusun algoritme dan menerapkan kode baris demi baris, tetapi pola tersebut dapat memandu kamu ke dalam desain yang baik dan dapat dipelihara.

Mungkin yang paling penting, karena pola desain menjadi semakin terkenal, developer memiliki kosa kata yang sama untuk mendiskusikan trade-off dan keputusan desain.

Kami yakin bahwa penggunaan pola desain akhir-akhir ini menjadi lebih penting dalam penilaian ujian daripada sebelumnya, sebagian besar karena pertumbuhan popularitasnya.

Tangani Kesalahan dengan Tepat

Kamu akan dievaluasi untuk penanganan kesalahan yang tepat dan jelas di seluruh proyekmu. Kamu mungkin melakukannya dengan sangat baik di GUI mu dan kemudian jatuh di server mu, tetapi itu penting di mana-mana dalam programmu.

Jangan Kembalikan Kode Kesalahan

Ini Jawa. Menggunakan kode kesalahan sebagai nilai kembali, daripada menggunakan pengecualian, adalah Ide yang Sangat Buruk. Kami yakin penilai ujian Anda mengetahui hal itu.

Jangan Mengirim Pesan Baris Perintah yang Berlebihan

Jangan terlalu bertele-tele dengan pesan baris perintahmu, dan pastikan untuk tidak membiarkan pesan debug masuk! Pesan baris perintah kamu harus menyertakan hanya apa yang diperlukan untuk memverifikasi startup programmu dan jumlah yang sangat minimal dari pesan status yang mungkin penting jika program gagal.

Tetapi secara umum, jika terjadi kesalahan yang kamu tahu bisa menjadi salah, kamu harus menanganinya dengan pengecualian. Apa pun yang kamu lakukan, jangan gunakan pesan baris perintah untuk mengirimkan pesan peringatan kepada pengguna! Gunakan kotak dialog yang tepat jika sesuai.

Gunakan Kotak Dialog Jika Sesuai

Di sisi lain, jangan gunakan kotak dialog untuk setiap kemungkinan pesan yang mungkin perlu diketahui pengguna. Jika kamu perlu menampilkan informasi yang tidak bersifat mendesak kepada pengguna (hal-hal mendesak seperti masalah penguncian data atau jika kamu perlu menawarkan opsi "Yakin ingin Keluar?"). Dalam banyak

kasus, kotak dialog adalah apa yang akan kamu gunakan untuk memperingatkan pengguna ketika sesuatu dalam programmu telah menangkap pengecualian, dan kamu membutuhkan masukan pengguna untuk menanganinya dengan tepat.

Lempar Pengecualian yang Dicentang dengan Benar

Ada waktu dan tempat yang tepat untuk melempar pengecualian yang dicentang, dan keengganan untuk membuangnya bisa sama buruknya dengan melemparkannya secara sembarangan.

Gunakan pengecualian waktu proses untuk kesalahan pemrograman.

Gunakan pengecualian yang dicentang untuk hal-hal yang mungkin dipulihkan kode mu(mungkin dengan bantuan dari pengguna).

Pengecualian yang dicentang hanya untuk kondisi yang benar-benar luar biasa.

Jangan gunakan pengecualian untuk kontrol aliran! Ya, tidak jika kamu berharap berhasil baik dalam ujian dan kehidupan nyata.

Ingat, pengecualian yang dicentang pasti tidak datang secara gratis pada waktu proses; mereka punya overhead. Gunakan saat, tetapi hanya saat, kamu membutuhkannya.

Buat dan Lempar Pengecualian Anda Jika Sesuai

Manfaatkan pengecualian standar jika memungkinkan, tetapi jangan ragu untuk membuatnya sendiri jika sesuai. Jika terdapat kemungkinan yang wajar bahwa kondisi luar biasa dapat dipulihkan, gunakan pengecualian yang dicentang dan coba menanganinya.

Biasanya, pengecualian yang kamu buat dapat dianggap sebagai Pengecualian Bisnis — dengan kata lain, hal-hal seperti “RecordLockedException” atau “InsufficientSearchCriteriaException”.

Semakin spesifik pengecualianmu, semakin mudah kode kamu dapat menanganinya, dan kamu mendapatkan manfaat dari menyediakan catch blockan tertentu, sehingga menjaga perincian catch blockan kamu berguna. Kebalikan dari strategi itu adalah hanya memiliki segalanya dalam satu blok percobaan besar yang menangkap Exception (atau lebih buruk lagi, Throwable!).

Tangkap Pengecualian Penerapan Tingkat Rendah

Dan Lempar Pengecualian Bisnis Tingkat Tinggi Katakanlah kamu menangkap SQLException (kemungkinan tidak pada ujian Pengembang). Apakah kamu mengembalikan ini ke klien? Tentu saja tidak. Untuk klien, ini termasuk dalam kategori "terlalu banyak informasi". Klien seharusnya tidak tahu — atau peduli — bahwa server database kebetulan menggunakan SQL. Sebaliknya, berikan pengecualian bisnis kustom yang lebih bermakna kepada klien yang dapat dia tangani.

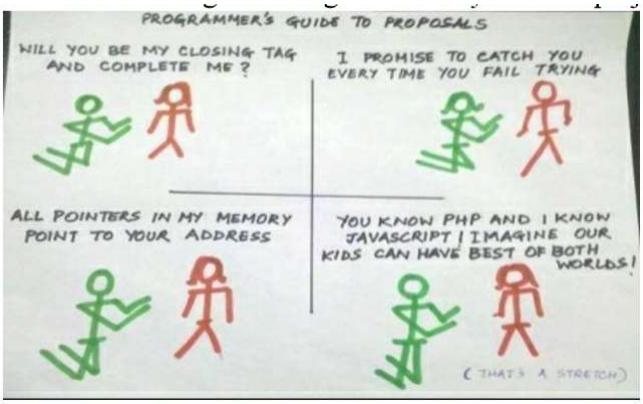
Pengecualian bisnis yang lebih bermakna itu didefinisikan dalam antarmuka publikmu, sehingga klien mengharapkannya sebagai kemungkinan. Namun hanya meneruskan pengecualian tingkat rendah ke klien mencerminkan desain yang buruk, karena ini memasangkan klien dengan detail implementasi server yang tidak pernah menjadi ide bagus dalam desain OO.

Never, Ever, Ever Eat dan Exception

Yang kami maksud dengan makan adalah praktik mengerikan berikut ini:

```
try {
doRiskyThing(); }
catch(Exception e) { }
```

Lihat apa yang hilang? Dengan menangkap pengecualian dan kemudian tidak menanganinya dengan cara apa pun, itu benar-benar tidak diperhatikan, seolah-olah tidak pernah terjadi. Kamu setidaknya harus mencetak jejak tumpukan. Menempatkan sesuatu seperti ini dalam proyek ujian mu mungkin merupakan pukulan mematikan.



Masalah Database Java : Memahami Masalah Utama Database Java

Pengantar

Ini adalah area yang sangat penting menurut saya, Ini adalah area di mana solusimu untuk masalah akan berdampak terbesar pada skormu. Kamu akan diminta untuk membuat database.

Dari awal. Dan karena akan ada klien bersamaan (atau setidaknya kemungkinan klien bersamaan), kamu harus yakin benar bahwa kamu mengelola penguncian data dengan benar.

Bagaimana dirimu menerapkan mekanisme pencarian, pembaruan, dan penguncian sepenuhnya terserah dirimu. Sekali lagi, pasti tidak ada Satu Jawaban yang Tepat untuk solusi kamudalam masalah ini. Tetapi bagaimanapun kamu memilih untuk melakukannya, pastikan bahwa logikanya masuk akal. Misalnya, meskipun kamu tidak pernah mengalami kebuntuan selama pengujian, bahkan jika ada kemungkinan sekecil apa pun (tidak peduli seberapa kecil kemungkinannya) hal itu bisa terjadi, kamu dapat dengan mudah gagal dalam ujian wawancara meskipun hampir semua hal lain dalam lamaranmu sempurna.

Dua masalah terbesar adalah mengunci dan mencari, tetapi mengunci adalah tempat Uang Besar sebenarnya. Kita akan mulai dengan ikhtisar singkat tentang konsep-konsep utama, diikuti dengan daftar pertanyaan inspiratif lainnya yang menggugah pikiran.

Membangun database java

Jika kamu ingat dari Bab 10, dirimulah yang harus membangun database; klien terlalu murah atau neurotik untuk berinvestasi dalam database komersial, bahkan yang gratis. Jadi, apa itu database?

Itu tergantung pada tugasmu, tetapi untuk tujuan ujian, perangkat lunak yang memungkinkan-kamu-mengakses-kumpulan-catatan akan dilakukan. Kamu memiliki beberapa data, dalam beberapa format file di suatu tempat, dengan skema yang diketahui, dan tugasmu adalah menulis aplikasi yang memungkinkan data tersebut dicari dan dimodifikasi. Kamu mungkin juga perlu menambahkan dan menghapus catatan. Jadi konsepnya sederhana: klien membuat permintaan, berdasarkan beberapa kriteria pencarian, dan databasemu mengembalikan hasilnya.

Terkadang klien mungkin ingin, katakanlah, memesan Horse Cruise, dalam hal ini satu atau lebih catatan harus diperbarui. Dan kamu mungkin perlu memasukkan kapal pesiar baru atau menghapus kapal pesiar yang dibatalkan. Terlepas dari skenario sebenarnya, Masalah yang Sangat Besar adalah

Bagaimana cara melindungi data dari akses bersamaan?

Dengan kata lain, bagaimana cara mengunci record?

CATATAN:

Desain penguncian mu dan keputusan implementasi (dan eksekusi) adalah bagian terpenting dari tugas Pengembangmu. Habiskan sebagian besar waktu kamu untuk memastikan dirimu memiliki solusi yang tepat. Pastikan dirimu telah memenuhi semua persyaratan dalam dokumen tugasmu yang berkaitan dengan penguncian dan pembukaan kunci.

Jika bagian dari spesifikasi tugas kamu tidak jelas atau ambigu, kamu perlu membuat interpretasi (tebakan terbaik kamu tentang apa yang harus dilakukan) dan kemudian mendokumentasikan asumsi dan strategimu.

Dan ingat, klien bisa jadi lokal atau jarak jauh (dengan kata lain, di mesin yang sama dengan database atau di mesin di tempat lain di jaringan), jadi kamu harus memikirkan masalah yang terkait dengan kedua skenario tersebut.

Penguncian itu penting, tetapi untungnya ujian Pengembang tidak meminta kamu untuk menerapkan sistem transaksi terdistribusi lengkap menggunakan protokol komit dua fase.

Sebenarnya, ini jauh lebih sederhana daripada transaksi, tetapi kamu harus memahami masalah mendasar seputar akses bersamaan ke data. Di mana suami dan istri berbagi akun yang sama?

Jika kamu tidak sepenuhnya paham tentang cara menangani sinkronisasi, baca kembali bab itu. Untuk menerapkan strategi penguncianmu dengan benar, kamu akan membutuhkan pemahaman yang kuat tentang sinkronisasi, wait(), notify(), dan notifyAll(). Jadi, siap untuk beberapa pertanyaan? Sekali lagi, ini bukanlah urutan tertentu.

Pertanyaan untuk Ditanyakan pada Diri mu sendiri

Kami telah membaginya menjadi dua kategori, menelusuri dan mengunci. Namun ada banyak hal tentang penelusuran yang juga termasuk dalam kategori masalah GUI (Bab 13). Secara khusus, Anda harus yakin bahwa pengguna akhirmu tahu cara membuat kueri penelusuran.

Mencari

Seberapa mudah klien melakukan penelusuran? Dengan asumsi GUI itu sendiri ramah pengguna (dan kami akan banyak bicara tentang itu di Bab 13), bagaimana dengan kriterianya?

Bagaimana item kriteria pencarian direpresentasikan? Sebuah thread? A CriteriaObject?

Bagaimana klien tahu persis apa yang dapat mereka lakukan untuk mendasarkan pencarian?

Apakah pencarian kamu mendukung kecocokan boolean?

Apakah itu perlu? Basis data belum tentu diindeks, jadi pernahkah kamu memikirkan cara lain untuk membuat penelusuran seefisien mungkin?

CATATAN:

Jangan mengorbankan kejelasan dan kesederhanaan untuk keuntungan kinerja yang kecil. Jika perolehan kinerja besar, maka desain ulang sehingga kamu dapat memiliki algoritme yang cukup efisien yang juga jelas dan dapat dipelihara. Sudahkah kamu mendokumentasikan algoritme penelusuranmu?

Jika Anda menulis banyak dokumentasi untuk menjelaskan algoritme penelusuran Anda, mungkin ada yang salah dengan desainnya.

Apakah dokumentasi algoritme penelusuran Anda mudah dibaca dan dipahami?

Ketika klien mengirimkan permintaan pencarian, apakah bagian tertentu dari kriteria pencarian secara eksplisit cocok dengan bidang tertentu? Atau apakah Anda mencari semua bidang untuk setiap pencarian?

Jika Anda menggunakan 1.4, apakah Anda sudah menyelidiki apakah ekspresi reguler akan membantu?

Apa yang terjadi jika tidak ada yang cocok dengan kriteria pencarian klien?

Apakah harus sama persis?

Mungkinkah ada skenario di mana terlalu banyak catatan cocok dengan kriteria pencarian?

Pernahkah Anda mempertimbangkan masalah bandwidth saat merancang dan menerapkan format permintaan kriteria pencarian dan hasil server? Apakah Anda mengirimkan barang melalui kawat yang lebih besar dari yang mereka butuhkan?

Apakah kemampuan pencarian Anda fleksibel untuk pengguna akhir?

Apakah kemampuan pencarian Anda fleksibel untuk perubahan program di masa mendatang?

Berapa banyak kode, jika ada, yang harus diubah jika skema database berubah? Sudahkah Anda mengisolasi tempat di mana perubahan dapat terjadi untuk menghindari masalah pemeliharaan?

Apakah Anda benar-benar yakin bahwa Anda telah memenuhi persyaratan penelusuran yang ditentukan dalam spesifikasi tugas Anda? Kembali dan baca ulang. Slooooooooowly.

Mengunci

Apakah Anda benar-benar yakin bahwa skema penguncian Anda berfungsi di semua skenario yang memungkinkan?

Apakah tugas ujian Anda menentukan jenis penguncian tertentu sehubungan dengan membaca dan menulis?

Apa yang terjadi jika klien mencoba mendapatkan record dan record sudah dikunci? Apa yang dialami klien?

CATATAN : Ini penting. Pikirkan baik-baik tentang apa yang Anda inginkan terjadi.

Bagaimana Anda akan melacak catatan mana yang dikunci?

Bagaimana Anda akan melacak siapa yang mengunci setiap catatan? Apa kamu perlu tahu itu?

Bagaimana Anda mengidentifikasi klien secara unik sedemikian rupa sehingga Anda dapat mengetahui klien mana yang mengunci rekaman mana? Apakah ini tanggung jawab server atau klien?

Sudahkah Anda mempertimbangkan apakah ID untaian sesuai untuk mengidentifikasi klien secara unik?

Pernahkah Anda mempertimbangkan apakah nomor Math.random() sesuai untuk mengidentifikasi klien secara unik?

Jika klien membuat permintaan pada rekaman yang terkunci, bagaimana Anda akan memverifikasi bahwa klien yang sama yang memegang kunci?

Apa yang terjadi jika klien mencoba menggunakan rekaman terkunci ketika klien tersebut bukan klien yang memegang kunci?

Apakah mungkin memiliki catatan yang dikunci untuk waktu yang terlalu lama? Berapa lama waktu yang terlalu lama?

Adakah yang dapat atau harus Anda lakukan tentang durasi penguncian?

Apa yang terjadi jika klien mati tanpa membuka kunci?

Apakah server memerlukan cara untuk mengetahui klien turun? (Berbeda dengan hanya mengambil waktu manis mereka atau jika koneksi mereka sangat lambat.)

Adakah kemungkinan kebuntuan? Di mana dua atau lebih klien menunggu kunci satu sama lain?

CATATAN :

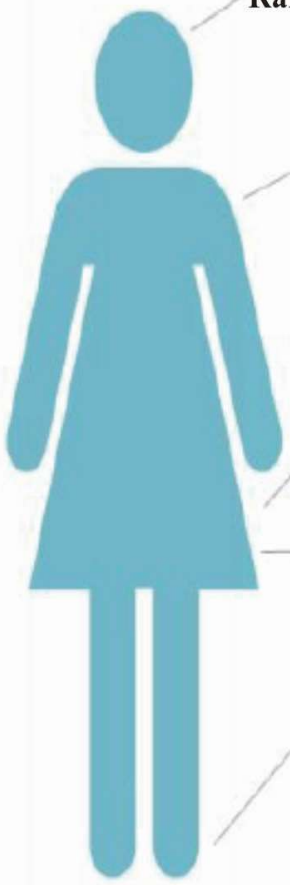
Periksa ini lebih dari yang Anda periksa untuk hal lain.

Apakah Anda benar menggunakan wait(), notify(), dan notifyAll()?

Apakah Anda sudah jelas tentang implikasi notify() vs notifyAll()?

Gambar 31.1 Pakaian untuk wawancara

BAGAN PAKAIAN WAWANCARA: WANITA



Rambut: Rambut pendek bisa dibiarkan rapi, rambut panjang diikat dan dirapikan dari wajah, juga bisa digunakan untuk membentuk wajah.

Blus: Blus harus sesuai tubuh anda dan memungkinkan anda untuk bernapas.

Rok: Rok tidak boleh jatuh di atas lutut, pilih warna netral.

Celana: Celana harus cukup longgar agar tidak mengganggu ketika berjalan dan juga pilihlah warna yang netral.

Sepatu: Sepatu harus sesuai dengan warna celana atau rok anda.

Pengaplikasian make-up seminimal mungkin.

Gunakan minimal perhiasan - tidak ada tindik pada wajah atau tubuh.

Gunakan minimal parfum.

Kenakan blus berkancing berwarna terang.

Setelan celana atau rok berwarna gelap.

Bawalah tas atau portofolio.

Jika mengenakan setelan rok, sebaiknya panjang rok tidak di atas lutut.

Kuku jari harus dirawat dengan baik.

Stocking tidak boleh robek dan harus bening atau cokelat.

Sepatu harus menutupi ujung kaki

Menurut data tahun lalu dan tahun ini yang telah kami kumpulkan dari berbagai sumber, lebih dari 5,67.000 siswa dan profesional TI telah membaca buku ini dan berhasil memecahkan pekerjaan mereka di industri TI dan industri lainnya juga. Jangan Lupa untuk menulis review atau komentar pelanggan tentang buku ini. Untuk Struktur Data dan Algoritme & Pertanyaan Wawancara C-C++, Baca Buku Mendatang Harry- “Memecahkan Wawancara C & C++” dan Memecahkan “Wawancara Algoritme” Beri tahu teman Anda tentang Buku Java yang luar biasa ini.

DAFTAR PUSTAKA

B. H. Sirenden and E. L. Dachi. *Buat Sendiri Aplikasi Petamu Menggunakan CodeIgniter dan Google Maps API*, Yogyakarta: ANDI offset, 2012.

Badriyah, M. (2015). *Manajemen sumber daya manusia*. Bandung: Pustaka Setia .

Bangun, W. (2012). *Manajemen sumber daya manusia* . Bandung: Erlangga.

Dewi, Sutrisna. 2007. *Komunikasi Bisnis*. Yogyakarta: Penerbit Andi.

Hodgkinson, & J. K. Ford. *International review of industrial and organizational psychology 26*, 240-292. Los Angeles: Willey-Blackwell.

Kadir Abdul, 2002. *Dasar Pemograman Web Dinamis Menggunakan PHP,* edisi 1, Andi. Yogyakarta.

Kim, B. H. (2011). *Deception and applicant faking: putting the pieces together*. In G. P.

Law, S. J., Bourdage, J., & O'Neill, T. A. (2016). *To fake or not to fake: Antecendents to interview faking, warning instruction, and its impact on applicant reactions*. Original research, 7, 1-12.

Narbuko, C., & Achmadi, H. A. (2003). *Metodologi Penelitian*. Jakarta: PT Bumi Aksara.

Orina, N. B. (2018). *Impact of Individual Differences on Faking Behavior [Electronic Theses Dissertations]*. Amerika Serikat (AS): South Dakota State University.

Periantalo, J. (2015). Penyusunan Skala Psikologi: Asyik, Mudah & Bermanfaat. *Yogyakarta: Pustaka Belajar.*

Pervin, L. A., Cervone, D., & Oliver, P. J. (2012). *Psikologi kepribadian: Teori dan Penelitian (edisi ke-9)*. Kencana Prenada Media Group. Jakarta.

Prasmara, A., & Wijaya, N. H. (2017). *Relations between big five personalities, motivation to fake, and applicant faking behavior.* JMK, 19(2), 85-90.

Primawestri, K, R., & A, R. (2016). *Kepribadian honesty – huminity dan perilaku impression management pada karyawan dinas koperasi dan umkm prov. jawa tengah.* Jurnal Empati, 5(4), 780-785.

S.Rosa A. dan M. Shalahudin 2013. Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek. *Bandung : Informatika*.

Stevens, C. K., & Kristof, A. L. (1995). *Making the right impression: a field study of applicant impression management during job interviews*. J. Appl. Psychol. 80, 587–606. doi: 10.1037/0021-9010.80.5.587

Tubbs, Stewart L and Moss, Sylvia., 2000. *Human Communication*, Bandung: Rosdakarya.

Wahyuni, S., Hakam, M. S., & Iqbal, M. (2015). *Analisis mekanisme seleksi berbasis pendekatan teori human capital.* Jurnal Administrasi Bisnis, 21(1), 1-7.

Winantu, Asih dan Saputro, Wahyu T. 2010. *Pemrograman Web dengan HTML, XHTML, CSS, Javascript.* Yogyakarta: Explore.