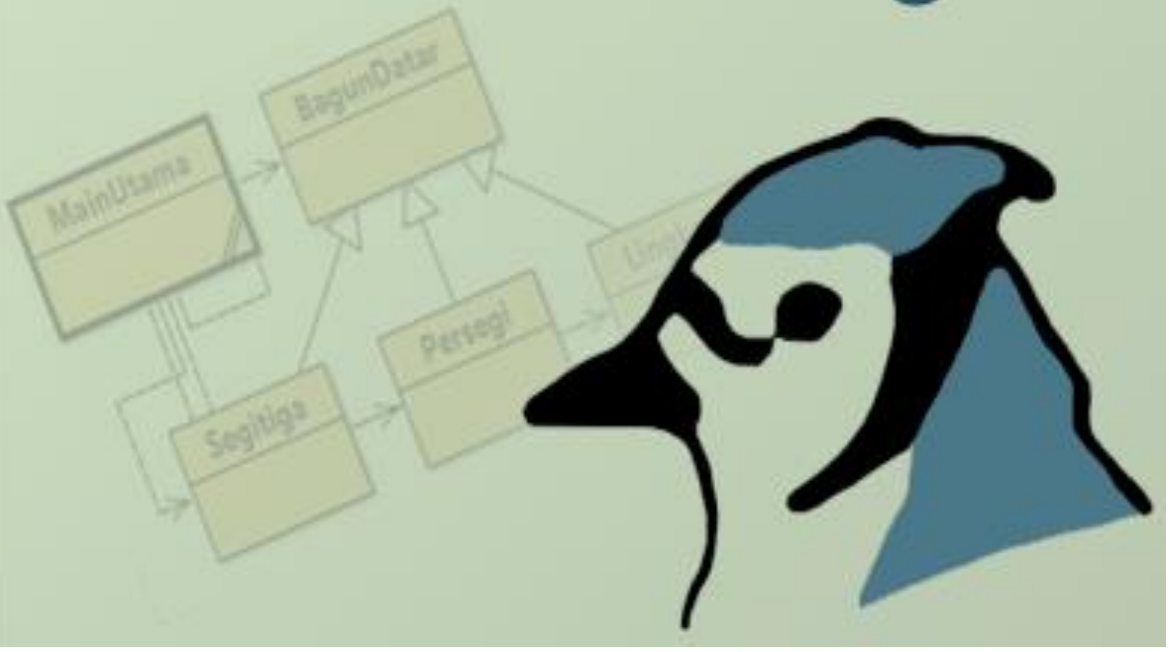
Oleh : Budi Hartono. M.Kom

PEMROGRAMAN BERORIENTASI OBJEK dengan JAVA BlueJ



Oleh : Budi Hartono. M.Kom

PEMROGRAMAN BERORIENTASI OBJEK dengan JAVA BlueJ

PENERBIT :
YAYASAN PRIMAAGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@yahoo.ac.id

PEMROGRAMAN BERORIENTASI OBJEK dengan JAVA BlueJ

Penulis :
Budi Hartono, M.Kom

ISBN : 978-623-8120-21-5 (PDF)

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

KATA PENGANTAR

Puji syukur saya panjatkan kehadirat Alloh SWT yang telah melimpahkan rahmat dan hidayahnya sehingga buku dengan judul pemrograman berorientasi objek dengan BlueJ dapat diselesaikan. Buku ini disusun sebagai panduan pembelajaran mata kuliah pemrograman berorientasi objek untuk dosen dan mahasiswa. Materi yang terkandung dalam buku dibuat dengan sesederhana mungkin sehingga mahasiswa mudah untuk mempelajari dan mempraktekkannya. Setelah mempelajari buku ini mahasiswa diharapkan dapat mengimplementasikan dalam pembuatan program baik untuk diri sendiri maupun organisasi atau bisnis. Penulis menyadari bahwa buku ini masih jauh dari sempurna dan mungkin juga banyak terdapat kesalahan. Untuk itu penulis sangat mengharapkan adanya saran dari semua pihak untuk dapat lebih menyempurnakan buku ini. Akhir kata penulis ucapkan banyak terima kasih kepada semua pihak yang telah membantu hingga tersusunnya buku ini, semoga buku ini dapat bermanfaat bagi semua pihak.

Semarang, Februari 2023

Dosen pengampu
Budi Hartono

Daftar Isi

BAB-1

KONSEP PEMROGRAMAN BERORIENTASI OBJEK (PBO)

Apa Itu Pemrograman

Memprogram adalah merancang sekaligus membuat program pada komputer. Membagun program dapat berupa pemngunan sebuah website, aplikasi dalam bentuk desktop , aplikasi Android, aplikasi jaringan, dll. Pembangunan pemrograman diawali dengan tahapan berupa menyusun source code, melakukan testing, mengedit, menyusun kembali , dan melakukan testing ulang. Langkah ini dilakukan berulang ulang sampai pada titik program yang dikehendaki.

Fungsi pemrograman

Tugas pemrograman adalah memberikan instruksi ke komputer sehingga orang dapat memberi tahu mesin dengan tepat apa yang harus dilakukan. Bahasa pemrograman adalah sarana komunikasi manusia-komputer yang memungkinkan komputer memproses data secara sistematis yang dibuat oleh penerjemah. Bahasa pemrograman juga memfasilitasi penggunaan mesin untuk mengurangi tenaga manusia.

Definisi PBO

Pemrograman Berbasis Objek (PBO) / Object Oriented Programing (OOP) adalah paradigma berbasis objek di mana semua anggota dan fungsi dikemas dalam wadah yang bernama objek atau kelas. Objek dapat menerima message, mengolah, melakukan pengiriman pesan, men-save dan mengolah data. Sementara objek melakukan komunikasi dengan cara memberikan informasi antara yang satu dengan yang lain. Pengertian pemrograman berbasis objek adalah cara membuat program dengan menggunakan objek. Objek itu sendiri memiliki data yang menggambarkan atribut objek. Selain itu juga memiliki fungsi atau prosedur yang lebih dikenal dengan metode.

Mudahnya , Pemrograman Berbasis Objek dapat disimpulkan sebagai suatu konsep pembangunan program dengan menguraikan masalah pada program menggunakan objek. Objek dapat digunakan sebagai fungsi tersendiri yang dibangun secara mandiri. Saat membangun aplikasi, objek-objek ini bertukar informasi atanra objek satu dengan obejek lain untuk mencapai hasil yang diinginkan.

Konsep dasar PBO

Pemrograman berorientasi objek (PBO) adalah suatua gaya pemrograman di mana pembuat program tidak hanya mendefinisikan tipe data dari organisasi data, tetapi juga jenis operasi (fungsi) yang dapat digunakan pada organisasi data. Cara sepeti ini, organisasi data menjadi objek yang berisikan data dan fungsi. Demikian juga, pengembang dapat membuat hubungan antara satu objek dengan objek lainnya. Contohnya, objek dapat mewarisi properti dari objek yang lain. Satu dari sekian banyak keunggulan utama teknis pemrograman berorientasi objek dibandingkan metode pemrograman sebelumnya adalah pembuat program dapat membuat modul yang tidak perlu diedit saat tipe objek baru ditambahkan. Seorang pembuat program hanya bisa membuat objek baru yang mewariskan properti dari objek yang pernah dibuat sebelumnya. Hal ini membuatnya lebih mudah untuk memperbaiki program objek.

Secara umum, objek dunia nyata memiliki dua sifat, perilaku dan keadaan. Misalnya sepeda dapat memiliki tempat berupa persneling, ban, dan pedal. Pada saat yang sama, sepeda juga memiliki perilaku seperti pengereman, perpindahan gigi, percepatan dan masih banyak lagi. Hal ini sesuai dengan seseorang yang menggunakan OOP, karena OOP sendiri juga memiliki dua sifat yaitu metode

dan variabel. Metode bertindak sebagai behavior dan peubah bertindak sebagai keadaan.

Kondisi di PBO

Java Bluej merupakan bahasa pemrograman berbasis objek dan hal ini di dukung dengan adanya fitur yang sering disebut dengan istilah berikut: Objek, kelas, properti, abstraksi, enkapsulasi, objek, instance, metode, pewarisan, polimorfisme, dan penguraian pesan. Ini mendukung begitu banyak fitur sehingga perlu didiskusikan secara individual.

- ✓ Object:

  Objek adalah komponen yang berasal dari kelas. Sebuah objek mempunyai beberapa properti kunci, yaitu: 1) State, 2) behavior, 3) attribut. Keadaan suatu objek menggambarkan data atau nilai yang dipunyai, contoh objek. Objek yang keadaan nilainya adalah warna, berat, bentuk, sedangkan perilaku objek merepresentasikan perilaku, fungsionalitas, dan tindakan objek tersebut, umpama objek sepeda yang berperilaku seperti berjalan, behenti dan juga dapat berbelok. Sedangkan identitas merupakan sesuatu yang melekat pada suatu benda dan didapat atau dapat diterapkan lewat status benda tersebut, seperti attribut yang dimiliki sepeda, seperti "biru", "roda 2", dan "Wymcicle".

  Di Java Blue, pembuatan objek dilakukan oleh kelas dengan struktur sebagai berikut:

  NamaClass namaobject = new NamaClass();

  Contoh

  ```
  Manusia tubuh = new Manusia();
  ```

Pada dasarnya sebuah objek memiliki dua sifat utama, contoh:

- Pada objek mempunyai atribut seperti state yang disebut state
- Pada objek mempunyai perilaku, yang kemudian disebut metode (behavior).

Berikut contoh state dan behavior dari sebuah object sepeda.

Tabel 1.1 State dan Behavior pada Objek Sepeda

State	Behavior
Pedal	Kecepatannya menaik
Roda	Kecepatannya menurun
Gigi	Perpindahan gigi sepeda

Dalam pemrograman berbasis objek, objek disini adalah objek perangkat lunak yang menyimpan statusnya dalam peubah dan informasi serta perilaku dalam metode atau fungsi. Menggunakan perintah baru, buat objek dengan nama kelas yang akan dipakai sebagai kelas.

Class

Merupakan model dari suatu objek. Kelas berisi kode yang mendefinisikan bagaimana objek berperilaku dan melakukan interaksi satu dengan yang lain atau dengan diri mereka sendiri. Kategori pemrograman bisa di terjemahkan sebagai suatu model. Jika kita ingin membuat sesuatu (kue), kita membutuhkan loyang kue. Dari cetakan kue yang ada saat ini, kita bisa membuat berbagai jenis kue tersebut. Cetakan yang berbeda tersebut dapat dilihat sebagai objek, yaitu hasil dari kelas. Secara visual, class dan bagiannya dibentuk menjadi struktur class sesuai dengan contoh struktur dibawah:

Class Name
Attribute
Methode ()

Struktur diagram kelas di atas menjelaskan hal-hal sebagai berikut:

- nama kelas adalah nama kelas
- Atribut adalah variabel yang diatur menurut jenisnya
- Metode () adalah fungsi atau perilaku yang dapat muncul di blok kelas.

Dalam struktur diagram kelas, visibilitas atribut dan metode ditentukan oleh kebutuhan dan tujuan desain. Visibilitas merupakan tingkatan dalam hak akses yang diberikan pada atribut dan metode. Visibilitas ini menandakan kedua bagian untuk dipakai secara lebar atau tidak dipakai. Pada umumnya, visibilitas atribut dan metode dalam suatu kelas dapat berupa:

Visibility / Modifier	Simbol	Hak Akses
Public	+	Semua class dapat mengakses variabel dan method yang visibility modifiernya public
Private	_	Variabel dan method dengan visibility private tidak dapat diakses oleh class lain
Protected	#	Visibility jenis ini attribut dapat diakses oleh method method pada class-class tertentu
Friendly	Tidak didifinisikan	Dapat diakses di subclass yang sama yang beradap ada package yang sama pula

Pada Kenyataannya, Java bluej mendukung esensial kelas, yang memvisualisasikan pemrograman berorientasi objek. Kelas Java juga ditulis struktur berikut:

```
<visibility> class NamaClass {
    // badan class
}
```

Berdasarkan struktur kelas di atas, penulisan kelas mengacu pada aturan berikut:

- Pengeditan dapat bersifat publik atau standar
- Kelas adalah kata kunci yang harus disertakan
- Nama kelas juga dikenal sebagai pengidentifikasi, dan penamaanya diawali dengan huruf besar; harus dimulai dengan angka; spasi kosong tidak boleh; dan ClassName tidak perlu panjang panjuntuk penamaan cukup dibuat singkat.
- // Badan kelas adalah badan blok yang dipakai untuk menulis souerce code program.
- Source code program bisa berupa atribut (variabel), metode, konstruktor atau deklarasi (event) deklarasi lainnya.
- Kelas ditandai dengan kurung kurawal { //badan kelas }

Contoh Penerapan class

```
class Manusia {
int usia;
float nilai;
float tinggi;
}
```

Perbedaan antara Kelas dan Obyek:

- Mudahnya , objek itu merupakan sesuatu yang dapat dipisahkan antara yang satu dengan yang lain. Segala sesuatu yang ada di alam semesta ini adalah objek. Contoh: Orang, sepeda motor, hewan, tumbuhan, meja, kursi atau bahkan benda non fisik seperti peristiwa atau konsep. Dari sini kita dapat menyimpulkan bahwa objek tidak harus fisik, tetapi dalam PBO objek adalah bentuk nalar. Attribut bisa menyimpan informasi (state) dan memiliki perilaku (behaviour) yang dapat diterapkan atau dimanipulasi pada state objeknya.
- Objek di PBO terus memiliki status dan properti yang sama dengan objek di dunia sebenarnya, karena objek di OOP pada dasarnya adalah memvisualisasikan dunia sebenarnya. Objek dalam OOP merepresentasikan status melalui variabel, sedangkan propertinya direpresentasikan sebagai metode. Metode adalah fungsi (subrutin) yang terkait dengan objek.
- Dalam konteks PBO, objek adalah turunan dari kelas (yang segera dibuat) pada waktu eksekusi (seperti pernyataan variabel dalam pemrograman tersetruktur).
- Jadi semua objek adalah turunan dari suatu kelas dan Objek juga disebut instances
- Jadi satu kelas digunakan untuk membuat banyak objek
- Proses pembuatan objek dari kelas disebut Instansiasi.

Karakteristik Objek:

- Pada objek mempunyai atribut seperti state
- Pada objek mempunyai perilaku (behaviour)
- Contoh:
  - ~ objek sepeda
  - ~ Berisi atribut (status):
  - ~ Pedal, roda, persneling, warna, jumlah roda.
  - ~ berperilaku (berperilaku):
  - ~ Tingkatkan kecepatan, kurangi kecepatan, persneling sepeda berhenti

Kelas:

- Kelas adalah konsep yang lebih luas dalam hierarki objek dan didapatlkan dari hasil generate objek yang memiliki beberapa karakteristik yang sama. Di PBO, sebuah kelas adalah hasil pemodelan fakta tentang objek yang berguna untuk aplikasi terprogram. Pemodelan fakta-fakta ini disebut abstraksi.
- Beberapa objek serupa sering ditemukan dalam sistem
- Beberapa objek yang mirip dapat diidentifikasi sebagai suatu kategori (category).
- Kelas menentukan model dan behavior objek
- Kelas adalah model/prototipe yang menjelaskan jenis objek
- Ada cara untuk mengenkapsulasi kumpulan data dan metode yang bekerja dengan kumpulan data
- Kelas adalah “keluaran” (cetak biru) dari suatu objek.
- Pada class tersebut dapat dibuat objek baru dan pada tiap tiap kelas dapat mempunya state yang berbeda.
- Beberapa objek serupa sering ditemukan dalam sistem
- Beberapa objek yang mirip dapat diidentifikasi sebagai suatu kategori (category).
- Kelas menentukan pola serta perilaku objek
- Kelas merupakan model/prototipe yang mendefinisikan jenis objek

- Ada cara untuk mengenkapsulasi kumpulan data dan metode yang bekerja dengan kumpulan data.

Properti

- Properti atau atribut adalah data yang ada di dalam kelas itu sendiri, sebenarnya properti sama dengan variabel tetapi dalam OOP disebut properti. Aturan pengetikannya sama dengan variabel, yang dapat didefinisikan secara langsung, yang juga dapat dimulai dengan kata kunci var dan yang juga dapat diberi nilai secara langsung. Analog dengan sepeda, ciri utama sepeda adalah seperti type, merek, warna, jenis ban, ukuran ban, dll.
- Di PBO, properti ini secara realnya adalah sebuah peubah yang hidup di kelas. Semua peraturan dan tipe data yang biasanya dimasukkan dalam variabel juga dapat dimasukkan dalam properti. Aturan penamaan properti sama dengan aturan penamaan variabel.
- Properti atau yang disebut atribut adalah informasi yang terkandung dalam suatu kelas, informasi tersebut biasanya berupa properti.

Properti vs. Field

- Untuk penggunaan umum, properti dan field mengacu pada variabel kelas. Perbedaannya, bagaimanapun, adalah bahwa properti adalah variabel kelas dengan izin publik, sedangkan bidang adalah variabel kelas dengan izin terbatas (pribadi atau dilindungi).

Messages and Methodes

- Dalam menulis program berorientasi objek, pertama-tama harus mendefinisikan kelas, dan saat program sedang berjalan, menggunakan kelas dan objek dari kelas ini untuk menyelesaikan tugas. Untuk menginstruksikan kelas atau objek melakukan tugas, akan mengirim pesan ke sana. Misalnya, mengirimkan deposit pesan ke objek Akun untuk menyetor Rp. 1.000.000.
- Untuk kelas atau objek untuk memproses pesan, itu harus diprogram sesuai. Tidak bisa begitu saja mengirim pesan ke kelas atau objek apa pun. Ketika mengirim pesan hanya ke kelas dan objek yang memahami pesan yang dikirimkan. Untuk kelas atau objek untuk memproses pesan yang diterimanya, itu harus memiliki metode yang cocok, yang merupakan urutan instruksi yang diikuti kelas atau objek untuk melakukan tugas. Metode yang didefinisikan untuk kelas disebut metode kelas, dan metode yang ditentukan untuk objek adalah metode instan.
- Misalkan metode yang disebut jalan didefinisikan untuk objek Robot dan menginstruksikan robot untuk berjalan pada jarak yang ditentukan. Dengan metode yang ditentukan ini, dapat mengirimkan pesan jalan kaki ke robot, beserta jarak yang harus ditempuh. Nilai yang diberikan ke suatu objek disebut argumen dari sebuah pesan. Yang perlu Perhatikan adalah bahwa nama pesan yang dikirim ke objek atau kelas harus sama dengan nama metodenya.
- Pada analogi diatas pesan mengilustrasikan komunikasi satu arah; yaitu, sebuah objek melakukan operasi yang diminta (itu berjalan pada jarak yang ditentukan) tetapi tidak menanggapi pengirim pesan. Dalam banyak situasi, memerlukan balasan di mana objek merespons dengan mengembalikan nilai ke pengirim pesan. Misalnya ingin mengetahui jarak dari robot ke rintangan terdekatnya.
- Desainer robot dapat menyertakan metode getObstacleDistance yang mengembalikan nilai yang diinginkan. Metode mengembalikan nilai ke pengirim pesan. Metode mengembalikan nilai numerik, sebuah metode dapat melaporkan kembali status operasi

yang diminta. Misalnya, metode jalan bisa didefinisikan untuk mengembalikan status berhasil/gagal untuk menunjukkan apakah jarak yang ditentukan berhasil atau tidak (misalnya, gagal saat robot menabrak rintangan).

- Lihat metode kelas. Pada metode ini mengembalikan kecepatan maksimum yang mungkin dari semua objek Robot. Metode seperti kecepatan maksimum yang berhubungan dengan informasi kolektif tentang contoh kelas biasanya didefinisikan sebagai metode kelas. Jadi mendefinisikan metode instance untuk tugas yang berhubungan dengan instance individual dan metode kelas untuk tugas yang berhubungan dengan semua instance

✓ Method:

Bagian ini adalah fungsi di dalam kelas. Anda dapat masuk dan memakai tiga bentuk variabel penggunaan. Dalam pemrograman, metode digunakan untuk menyimpan variabel status dan menerapkan sifatnya menggunakan suatu metode. Metode dapat dipakai di dalam kelas, sementara setiap kelas dapat memiliki beberapa metode. Sebuah metode dapat ditulis dengan struktur berikut:

```
<modifier> nilaibaliktipemethod namaMethod (daftar parameter) {
// badan method
}
```

Dimana,

- Modifier : sama seperti pengubah kelas dan atribut
- nilaibaliktipemethod : memberikan metode nilai balik
  namaMethod : nama metode yang digunakan
  daftar parameter : Sebuah atribut (variabel) dan tipe datanya, sebuah atribut bisa lebih dari satu variabel
- badan method : adalah bagian di mana ekspresi metode ditulis

✓ Overloading:

Ada pembatas di kelas. Sebagai contoh, sebuah mobil memiliki metode Info dan sebuah truk memiliki metode serupa. Ini disebut kelebihan beban. Saat mobil nanti menggunakan atau memanggil metode info, maka dibuatlah metode komunikasi yang ada dalam kelas mobil. Tapi saat truk memanggil metode info, ada dua pilihan. metode info di kelas truk atau bahkan di kelas mobil

✓ Encapsulation :

Enkapsulasi, atau pembungkus, adalah mekanisme untuk menghubungkan metode dan properti yang nantinya akan oleh dan melindungi dari manipulasi eksternal dan penyalahgunaan oleh pengguna kelas. Dalam enkapsulasi, pemrograman objek memfasilitasi pelacakan bug. Konsep enkapsulasi meminimalkan kesalahan pengkodean, yang juga berkontribusi pada modularitas, memungkinkan satu kelas untuk berintegrasi dengan kelas lainnya. Agar enkapsulasi berfungsi seperti yang dirancang, jangan pernah menggunakan atribut langsung dari kelas lain, atribut harus digunakan dengan metode kelas Anda sendiri. Di Jawa, enkapsulasi dilakukan sebagai berikut:

(1) mendeklarasikan atribut kelas privat, dan (2) mendeklarasikan metode penyetel dan pengambil formulir menjadi publik sehingga data (variabel) dapat dimodifikasi dan ditampilkan. Misalnya, kelas memiliki nama dan nilai atribut (variabel), sehingga kelas tersebut harus memiliki metode setName() dan getName() dan setValue() dan getValue(). Ini adalah karakteristik dari implementasi modular. Untuk mengimplementasikan modul di Java, gunakan struktur dengan atribut seperti ini:

```
class ClassName{
  private tipe namaattribute1;
  private tipe namaattribute2;
}
```

- ✓ Polymorphism :
  Pada dasarnya polimorfisme atau transformasi menggunakan metode yang mirip akan tetapi menghasilkan sifat yang berbeda. Dalam pemrograman, polimorfisme mengambil banyak bentuk. Konsep tersebut memperluas konsep pewarisan di mana kita dapat membuat dan menghasilkan kelas baru yang nantinya dapat diperluas ke berbagai bentuk. Misalnya, ada superclass bernama Message, dan tiga subclass diturunkan dari class Message, yaitu class MailMessage, VoiceMessage, dan FaxMessage. Saat pemrogram memanggil metode SendMessage pada ketiga kelas ini, ketiga objek tersebut benar-benar mengirim pesan, tetapi dengan cara yang berbeda

- ✓ Inheritance / Pewarisan :
  Warisan adalah kemampuan suatu kelas untuk membuat kelas baru yang berasal dari kelas sebelumnya. Kelas yang baru dibuat dapat menambahkan properti atau metode baru, atau menimpa properti atau metode asalnya. Misalnya, kelas mobil mewah mewarisi keadaan dan perilaku dari kelas mobil tersebut. Hal yang sama berlaku untuk kelas mobil reguler. Kelas mobil mewah dan mobil biasa disebut sebagai subclass dari kelas mobil, yang disebut sebagai superclass (kelas induk). Semua subclass mewarisi status dan perilaku dari superclass mereka. Jadi semua subclass dari superclass yang sama memiliki status dan perilaku yang sama. Namun, setiap subclass dapat menambahkan status atau perilakunya sendiri.

- ✓ Coupling :
  Coupling merupakan tingkatan hubungan kode program satu dengan kode program lainnya. Dengan pemrograman berbasis objek, tingkat pengikatan dapat diminimalkan, yang membuat pemrograman menjadi lebih mudah.

- ✓ Subclass :
  Subclass adalah kelas yang mewarisi properti superclass. Ketika suatu kelas diturunkan dari kelas lain, objek turunannya disebut subkelas. Dalam pemrograman, subclass menyiapkan perilaku khusus yang membedakannya dari kelas induk, memungkinkan pemrogram untuk menggunakan kembali kode sumber dari superclass yang ada

- ✓ Superclass / Class Induk:
  Superclass merupakan kelas yang mewarisi atribut dan metode dari subclass. Itu juga dapat diartikan sebagai kelas yang masuk ke kelas lain (sebagai lawan dari subkelas).

- ✓ Abstraction :

  Abstrak merupakan cara melihat objek dalam bentuk mudahnya. Misalnya, perhatikan sebuah sepeda motor. Kita tidak perlu tahu komposisi bagian mesin dan penopang kelistrikannya yang agak rumit dan sulit, namun kita dapat melihat sepeda motor secara keseluruhan yang merupakan suatu benda dengan ciri dan cara tersendiri. Dengan cara berpikir yang sederhana ini, mengendarai sepeda motor tidak menuntut kita untuk mengetahui betapa rumitnya elemen dan menyusun elemen sepeda tersebut. Karena untuk mengendarai sepeda motor, seseorang harus tahu cara mengendalikan sepeda motor. Jadi, konsep abstraksi ini memungkinkan kita untuk melihat sistem yang besar dan rumit yang terdiri dari subsistem yang kompleks dan banyak sebagai paket sistem yang sederhana.

  Memahami objek-objek di sekitar kita adalah dasar untuk memahami pemrograman berorientasi objek. Yang penting adalah bagaimana mengubah objek menjadi program yang Anda ketahui.

- ✓ Instance :

  Membuat kelas menjadi objek disebut instance. Lebih lanjut dapat dijelaskan bahwa suatu class tidak dapat digunakan jika hanya berupa sketsa objek karena tidak dikonstruksi, sehingga class tersebut harus dibuat menjadi objek yang akan digunakan, demikian penerapannya pada pemrograman

- ✓ Special Method / Magic Method :

  Metode khusus merupakan metode yang difasilitasi oleh Java yang memiliki tujuan tersendiri. Sebagai contoh:

  metode konstruktor untuk dijalankan saat objek pertama kali dibuat dan dimaksudkan untuk menginisialisasi objek kelas, metode destruktor untuk menghapus sumber daya dari objek kelas saat objek tidak lagi diperlukan.

- ✓ Attributes :

  Attibut merupakan data yang mencirikan satu objek dengan objek lainnya. Atribut sering juga disebut sebagai peubah. Dalam pemrograman, atribut dibagi menjadi dua jenis, yaitu. H. variabel instan dan variabel kelas. Variabel instan merupakan suatu atribut dari setiap objek dari kelas yang sama. Setiap objek mempunyai dan mempertahankan value atributnya sendiri. Jadi setiap objek dari kelas bokeh yang sama memiliki nilai yang sama atau berbeda. Variabel kelas merupakan suatu atribut untuk keseluruhan objek yang digunaka dari kelas yang sama. Semua objek memiliki nilai atribut yang sama. Jadi semua objek dari kelas yang sama hanya memiliki satu nilai yang memiliki nilai yang sama.

  Syntac Penggunaan attribute :

  <modifier> tipetada namaattibute;

  Atau dengan inisialisasi nilai attribute seperti struktur umum berikut :

  <modifier> tipetada namaattibute=nilaiattribute;

  Contoh

- *public int usia;*
- *private string id=”admin123”;*
- *float tinggi = 167.89;*

✓ Behavior :

Ada hal-hal yang dapat dilakukan objek kelas. Dalam pemrograman, behavior dapat dipakai untuk menggunakan nilai atribut objek, menerima pesan dari objek lain, dan mengirim pesan ke objek lain untuk melakukan fungsinya.

Keuntungan Pemrograman Berorientasi Objek

Apa kegunaan belajar dan menggunakan pemrograman berbasis objek?

- Metode yang sangat didukung oleh pustaka objek. Dengan cara ini, program dapat diselesaikan dalam waktu singkat untuk melanjutkan ke item atau proyek berikutnya.
- Sstem yang dimodifikasi tanpa membutuhkan banyak modul. Jadi hanya objek-objek saja yang nantinya dimasukkan ke dalam sistem. Selain itu, sistem perangkat lunak juga dapat dikembangkan untuk mengakses area yang lebih kompleks.
- Perangkat lunak yang mudah diselaraskan, bahkan dalam skala besar..
- Pengembangan yang lebih cepat yang mengurangi biaya pengembangan saat membuat program..
- Pengembangan yang lebih cepat yang dapat mengalokasikan banyak waktu dan juga sumber daya yang digunakan untuk memverifikasi perangkat lunak.

Kelemahan Pemrograman Berorientasi Objek

Setelah tahu kemudahan dan kelebihannya, perlu juga diketahuai apay yang perlu diwaspadai dalam memakai pemrograman berbasis. Berikut adalah beberapa kerugian yang mungkin Anda temui saat menggunakan OOP:

- Sebagian pembaut program masih perlu adaptasi banyak waktu untuk nyaman dengan OOP.
- Waktu eksekusi program berjalan pelan.
- Biasanya ada metode yang lebih besar untuk mengukur program yang dibangun menggunakan OOP.
- Tidak semua masalah pemrograman diselesaikan dengan OOP.

BAB II
MENGENAL JAVA BLUEJ

Sejarah Java

Java adalah bahasa pemrograman yang di perkenalkan oleh Sun Microsystems. Pada tahun 1991, Sun Microsystems memulai Project Green untuk mengembangkan bahasa yang dapat digunakan dalam chip tertanam untuk perangkat elektronik konsumen yang cerdas. Awalnya, Java ditujukan untuk memprogram perangkat kecil seperti chip. Untuk beberapa alasan perkembangannya dalam bahaya. Karena fungsi Java kompatibel dengan perkembangan Internet, Java telah menjadi bahasa universal yang mendukung pemrograman desktop. Awalnya bahasa pemrograman ini diberi nama lain, Nama lain tersebut memberikan inspirasi pagi pengembangnya. Nama lain tersebut tidak digunakan dalam versi Java karena perangkat lunak lain tersebut telah mendapatkan hak cipta dagang , maka dicarilah penggantinya untuk nama tersebut adalah "Java".

Versi Awal

Versi asli Java pada tahun 1996 sudah menjadi versi rilis, sehingga disebut Java Versi 1.0. Versi Java ini berisi banyak paket standar asli yang dikembangkan lebih lanjut di versi selanjutnya:

- ✓ java.lang: Notasi kelas untuk primitif.
- ✓ java.io: Tentukan kategori masukan dan keluaran, termasuk akses file.
- ✓ Jawa. until: Penamaan kelas tambahan umpama kelas struktur data dan kelas kalender.
- ✓ java.net: Nama untuk kelas TCP/IP yang memungkinkan komunikasi dengan komputer lain melalui jaringan TCP/IP.
- ✓ Jawa. awt: Kelas dasar untuk aplikasi antarmuka pengguna (GUI).
- ✓ Applet Java: Kelas UI aplikasi dasar yang dapat diimplementasikan di browser.

Manfaat

Multi-platform. Keuntungan utama Java adalah dapat digunakan pada berbagai sistem operasi

- ✓ Operasi komputer dengan basis sekali ketik dapa di gunakan di berbagai platform.
- ✓ OOP (Object Oriented Programming), artinya semua aspek Java adalah objek. Java adalah bahasa pemrograman yang murni berorientasi objek.
- ✓ Full Library, Java dikenal dengan banyak atau bisa dikatan full library yang memungkinkan developer untuk membangun aplikasinya dengan menjadi dipermudah.

Kekurangannya

- ✓ Kerjakan sesaat, perbaiki di mana-mana - Masih ada hal-hal yang tidak kompatibel antar platform.
- ✓ Pembongkaran mudah. Dekompresi merupakan proses kebalikan dari kode jadi ke kode sumber.
- ✓ Pemakaianya memori tinggi. Program berbasis Java menggunakan lebih banyak memori daripada generasi bahasa tingkat tinggi sebelumnya.

Java BlueJ

BlueJ adalah lingkungan pengembangan terintegrasi yang menyediakan para programmer kerangka kerja yang mencakup editor, kompiler, dan lingkungan runtime. BlueJ sangat cocok untuk programmer Java pemula. BlueJ sendiri sudah digunakan secara luas. BlueJ di kembangkan oleh University of Kent di London. BlueJ sangat sederhana, kecil dan ringan, dengan interface program yang mudah dipahami jika dibandingkan dengan tool Java yang profesional seperti NetBeans atau eclips. BlueJ di design untuk proses pembelajaran. BlueJ menunjukkan diagram class yang dibuat, dan menunjukkan relasi yang terjadi antar class. Diagram ini untuk memudahkan siswa memahami konsep class yang dibuat.

BlueJ interaktif. Ketika program di jalankan, jika ada argumen yang harus di masukkan, blueJ akan memberikan inputan pada user untuk memasukkan argumen. BlueJ juga memungkinkan untuk membuat object tanpa dari program, dan memanggil method yang ada didalam project tanpa dari program. Membuat object dan memanggil method object bisa dilakukan secara interaktif.
BlueJ memungkinkan untuk mengetahui isi dari sebuah object, sehingga siswa lebih memahami konsep object yang telah dibuat. BlueJ bisa berjalan di windows, mac os, linux dan platform lain yang bisa menjalankan Java.
BlueJ tersedia unduhan gratis kunjungi http://www.bluej.org dan ikuti petunjuk pengunduhan dan pemasangannya.

Gaya Program Bluej

Penamaan

- ✓ Gunakan nama yang bermakna.
  Gunakan nama deskriptif untuk semua pengidentifikasi (nama kelas, variabel dan metode). Hindari ambiguitas. Hindari singkatan. Metode mutator sederhana harus dinamai set Something (...). Metode pengakses sederhana harus dinamai get Something (...). Metode Accessor dengan nilai pengembalian boolean sering disebut adalah Sesuatu (...), misalnya, isEmpty ().
- ✓ Nama kelas dimulai dengan huruf besar.
- ✓ Nama kelas beruapa kata benda.
- ✓ Metode dan nama variabel dimulai dengan huruf kecil.
  Ketiga kelas, nama metode dan variabel - menggunakan huruf kapital di tengah untuk meningkatkan keterbacaan pengidentifikasi majemuk, misalnya numberOfItems.
- ✓ Konstanta ditulis dalam UPPERCASE (Huruf Kapital)
  Konstanta sesekali menggunakan garis bawah untuk menunjukkan pengidentifikasi majemuk: MAXIMUM_SIZE

Tata Letak

- ✓ Lekukan satu tingkat adalah empat ruang.
- ✓ Semua pernyataan dalam satu blok diberi indentasi satu tingkat.
- ✓ Kurung kurawal untuk kelas dan metode sendirian di satu baris.
  Kurung kurawal untuk blok kelas dan metode berada pada baris terpisah dan berada pada level indentasi yang sama, misalnya:

```
public int getAge ()
{
    pernyataan
}
```

- ✓ Untuk semua blok lainnya, kurung kurawal terbuka di ujung garis.
  Semua blok lain terbuka dengan kurung kurawal di ujung baris yang berisi kata kunci yang mendefinisikan blok. Kurung penutup ada di baris terpisah, disejajarkan di bawah kata kunci yang mendefinisikan blok. Sebagai contoh:

```
while (kondisi) {
    pernyataan
}

jika (kondisi) {
    pernyataan
}
lain {
    pernyataan
}
```

- ✓ Selalu menggunakan kurung kurawal dalam struktur kontrol.
  kurung kurawal digunakan dalam pernyataan if dan loop bahkan jika tubuh hanya satu pernyataan.
- ✓ Gunakan spasi sebelum penopang pembuka blok struktur kontrol.
- ✓ Gunakan ruang di sekitar operator.
- ✓ Gunakan garis kosong antara metode (dan konstruktor).
  Gunakan baris kosong untuk memisahkan blok kode logis. Ini berarti setidaknya antara metode, tetapi juga antara bagian logis dalam suatu metode.

Dokumentasi

- ✓ Setiap kelas memiliki komentar kelas di bagian atas.
  Komentar kelas berisi setidaknya : deskripsi umum kelas, nama penulis, nomor versi
  Nomor versi dapat berupa angka sederhana, tanggal, atau format lainnya. Yang penting adalah bahwa pembaca harus dapat mengenali jika dua versi tidak sama, dan dapat menentukan yang mana yang lebih baru.
- ✓ Setiap metode memiliki komentar metode.
- ✓ Komentar dapat dibaca Javadoc.
  Komentar kelas dan metode harus dikenali oleh Javadoc. Dengan kata lain: mereka harus mulai dengan simbol komentar / **.
- ✓ Kode komentar (hanya) jika perlu.
  Komentar dalam kode harus dimasukkan di mana kode tidak jelas atau sulit dipahami (sementara preferensi harus diberikan untuk membuat kode jelas atau mudah dipahami jika memungkinkan), dan di mana itu membantu memahami suatu metode. Jangan komentar pernyataan yang jelas – anggap pembaca Anda memahami Java!

Batasan Penggunaan Bahasa

Urutan deklarasi: bidang, konstruktor, metode

- ✓ Elemen-elemen definisi kelas muncul (jika ada) dalam urutan berikut: pernyataan paket; pernyataan impor; komentar kelas; header kelas; definisi bidang; konstruktor; metode
- ✓ Fields mungkin tidak bersifat publik (kecuali untuk field terakhir).

- ✓ Selalu gunakan pengubah akses.

  Tentukan semua bidang dan metode sebagai privat, publik, atau terlindungi. Jangan pernah menggunakan akses default (paket pribadi).
- ✓ Impor kelas secara terpisah.

  Mengimpor pernyataan yang secara eksplisit menyebutkan setiap kelas lebih disukai daripada mengimpor seluruh paket. Misalnya
  - import java.util.ArrayList;
  - import java.util.HashSet; lebih baik dari
  - import java.util. *;
- ✓ Selalu sertakan konstruktor (bahkan jika badan kosong).
- ✓ Selalu menyertakan panggilan konstruktor superclass.

Dalam konstruktor subclass, jangan mengandalkan penyisipan otomatis panggilan superclass. Sertakan panggilan super (...) secara eksplisit, bahkan jika itu akan berhasil tanpanya.

- ✓ Menginisialisasi semua bidang dalam konstruktor.

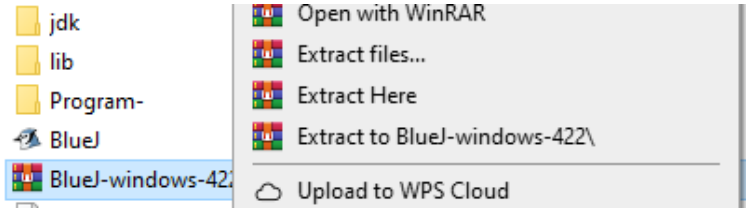
Instalasi BlueJ

BlueJ adalah lingkungan pengembangan terintegrasi yang menyediakan program dengan kerangka kerja yang mencakup editor, kompilator, dan lingkungan runtime., BlueJ sangat cocok untuk para programmer Java pemula. Untuk instalasi software BlueJ bisa didownload atau silakan kunjungi alamatnya di : https://www.bluej.org/ dan ikuti petunjuk pengunduhan dan pemasangannya sesuai dengan speksifikasi komputer Anda.

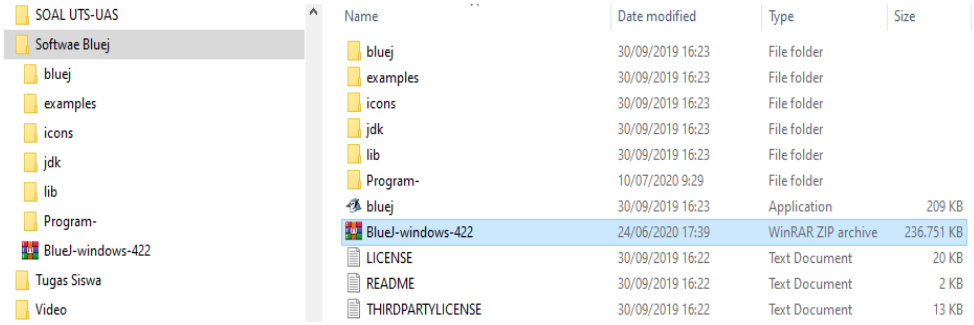
Dari halaman tersebut, kita bisa memilih sistem operasi komputer yang kita gunakan. Untuk operasi sistem windows, ada juga paket blueJ yang bisa dijalankan dari usb drive atau dari flash disk.

Jika dibutuhkan, anda harus melakukan instalasi komponen java yang disebut sebagai JDK. JDK bisa didownload di
https://www.oracle.com/technetwork/java/javase/downloads/index.html
Instalasi Bluej dengan sistem operasi windows , silahkan pilih atau klik Windows, tunggu beberapa saat biar proses download selesai, Jika sudah terdownload lakukan langkah 2 berikut ini.

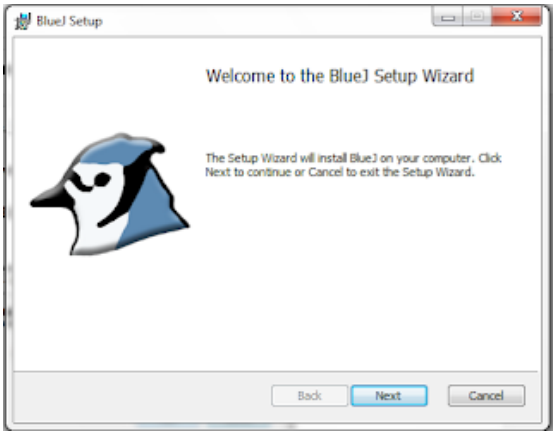
1. Extrack file dan Buka file installasi dan klick kanan Extract here



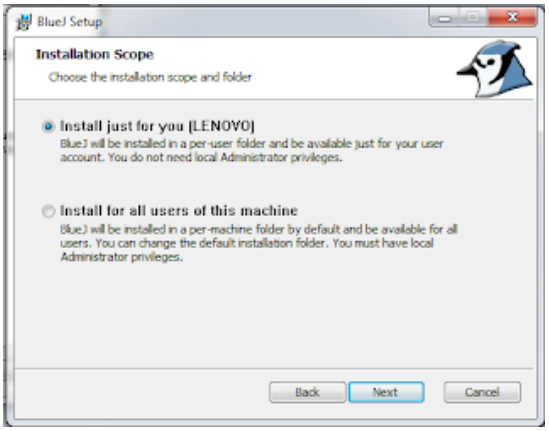
2. Hasil Exctrac File RAR



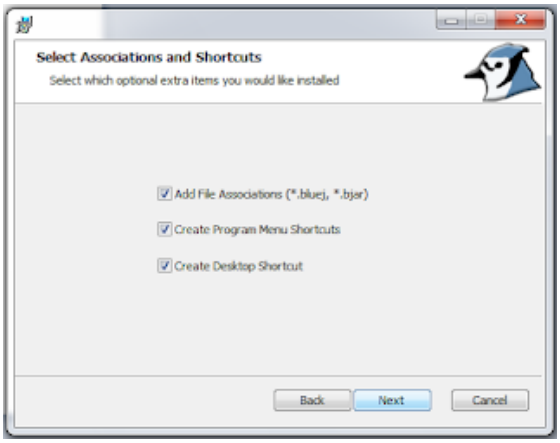
3. Klik Setup BlueJ



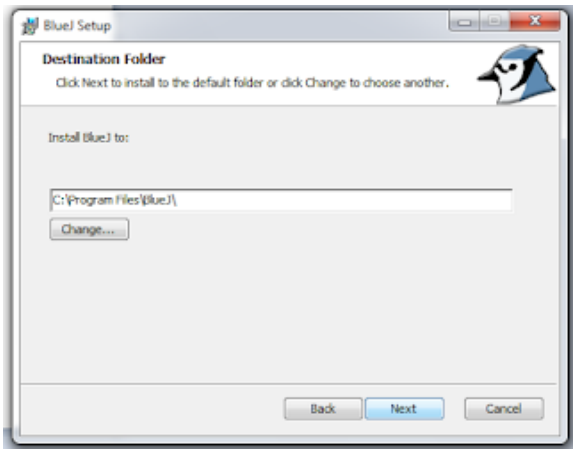
4. Pilih instal just for you



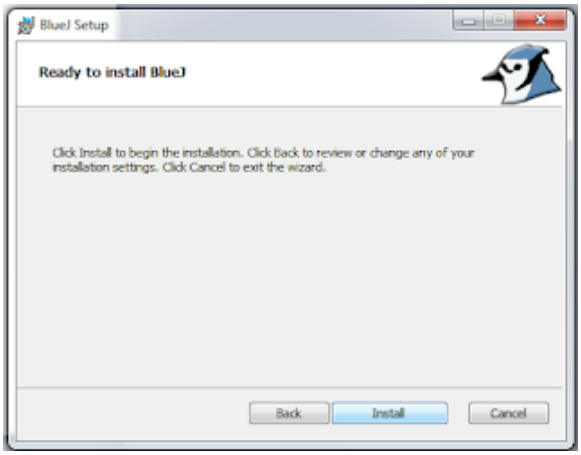
5. Centang semuanya



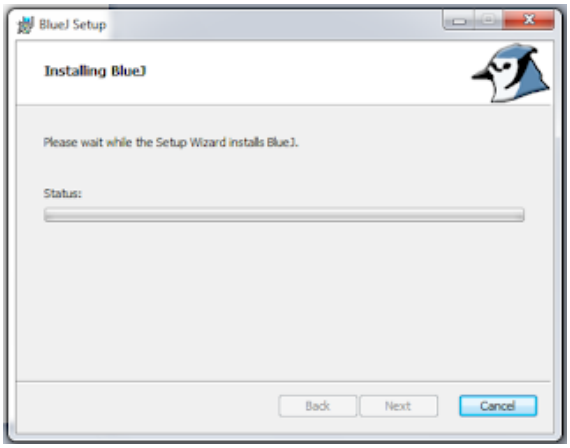
6. Pilih lokasi file untuk penyimpanan



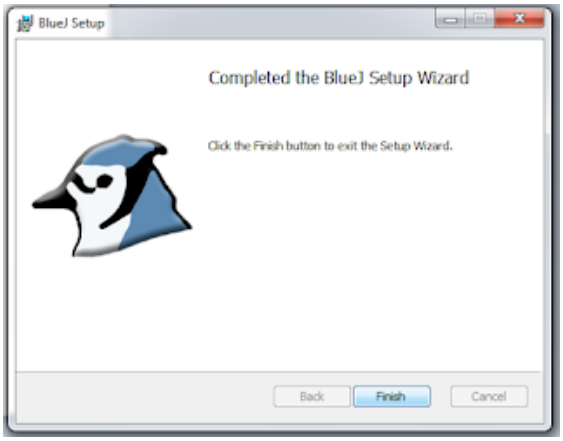
7. Pilih install



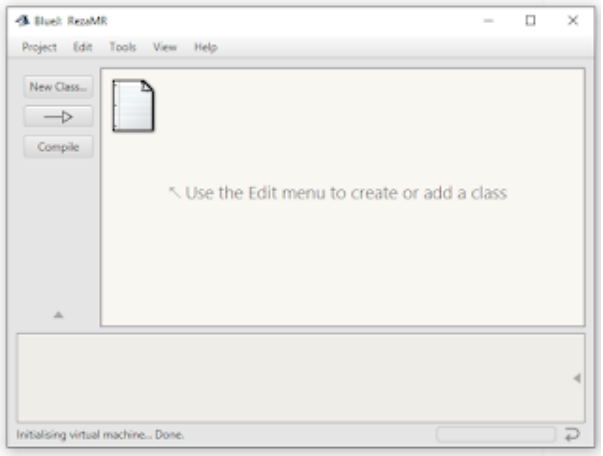
8. Tunggu installasi



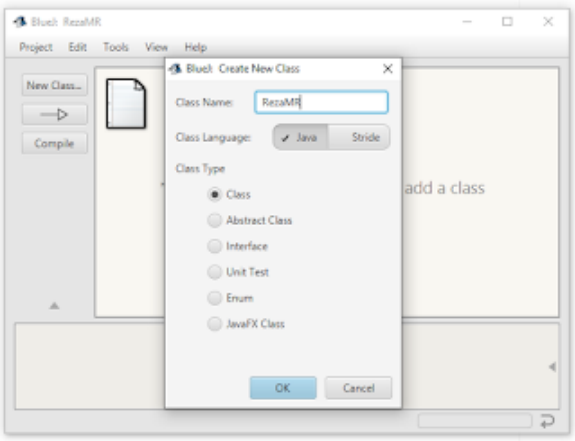
9. Klick finish



10. Ini tampilan awal BlueJ setelah selesai instal



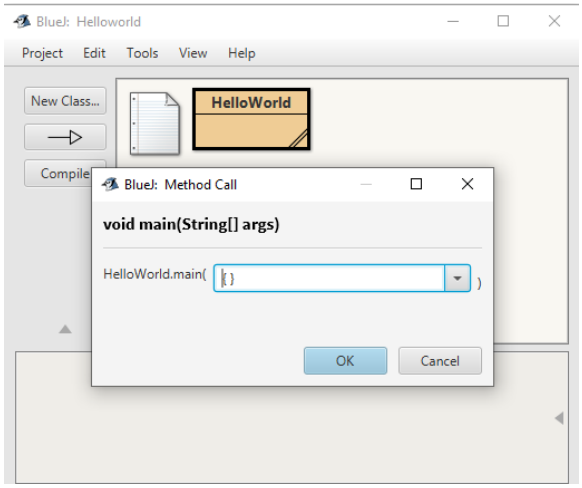
11. Baik Sekarang saatnya memulai coding simple untuk menampilkan nama pada BlueJ
12. Pertama buka pilih menu New Class dan akan muncul seperti foto dibawah setelah itu isi sesuai keinginan kamu



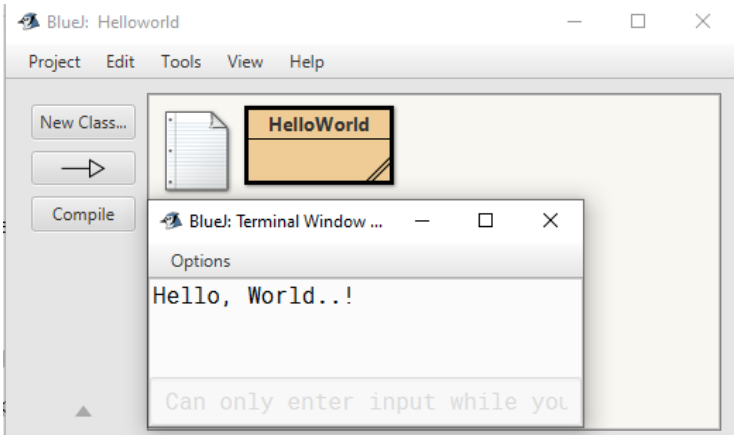
13. Buka file yang tadi dibuat dan hapus semua codingan awalnya dan masukkan codingan seperti ini

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World..!");
    }
}
```

14. Setelah itu pilih Compile, Jika coding berhasil dibawahnya akan ada tulisan Cass compiled - no systax errors
15. Setelah itu close coding dan klick kanan lalu pilih void main dan akan terlihat visualisasi seperti ini



16. Setelah itu ok dan akan muncul hasilnya

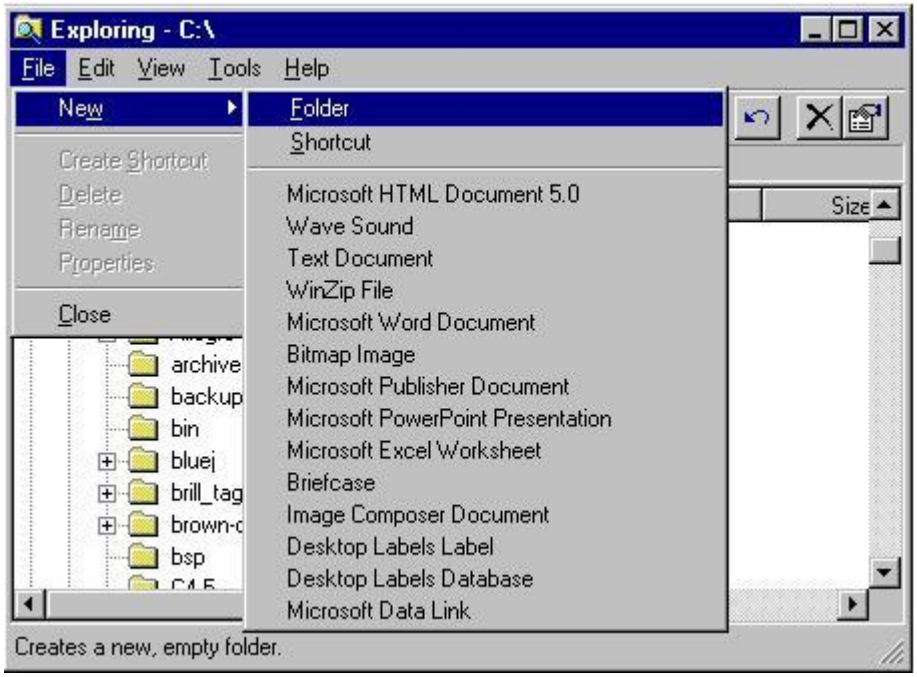


Langkah2 Membuat Program

Misalkan Anda ingin menyimpan **proyek** Java Anda di bawah direktori E: \ PBO-BLUEJ . (yang jelas letakan pada drive data, untuk nama menyesuaikan)

- ✓ **Buat folder / direktori menggunakan Microsoft Explorer.**

  Jika direktori ini (katakanlah E: \ PBO-BLUEJ) belum ada, gunakan **Windowd Explorer** untuk membuatnya. Pertama mulai Explorer dan pergi ke D / E: \ (direktori root drive D/ E). Kemudian di bawah **File** , pilih **New** , dan **Folder** .



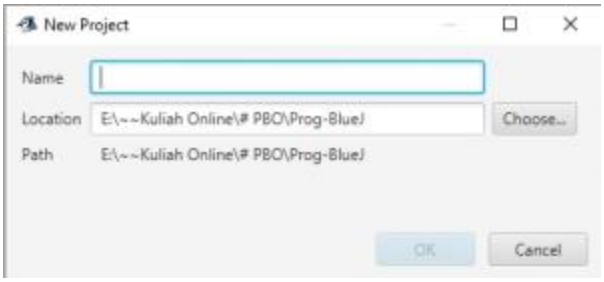
Ubah nama folder dari "Folder Baru" menjadi "PBO-BLUEJ".

- ✓ **Panggil BlueJ.**

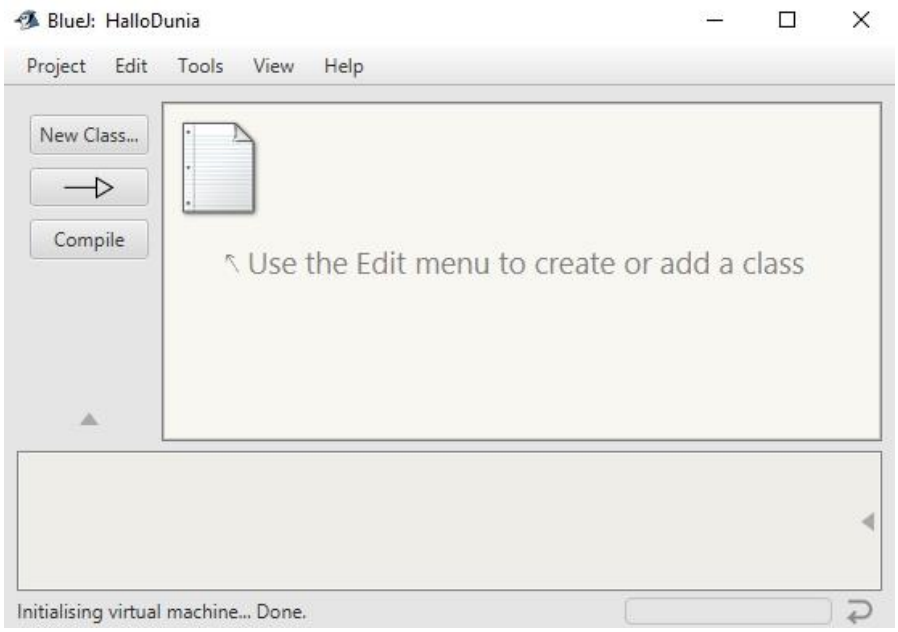
  Mulai BlueJ dengan mencari dan mengklik dua kali ikonnya. Jika Anda mengikuti petunjuk instalasi, itu akan memiliki ikon di desktop. 

- ✓ **Buat proyek baru.**

  Pada menu atas BlueJ, pilih Proyek dan Baru , untuk mendapatkan Dialog Proyek Baru. Dalam kotak Look In , cari dan pilih folder penyimpanan data yaitu : PBO-BLUEJ yang Anda buat pada Langkah sebelumnya. Ini mungkin mengharuskan Anda untuk naik / turun pohon direktori beberapa kali. Masukkan nama proyek (katakan "HelloWorld") di Nama file dan klik Buat .

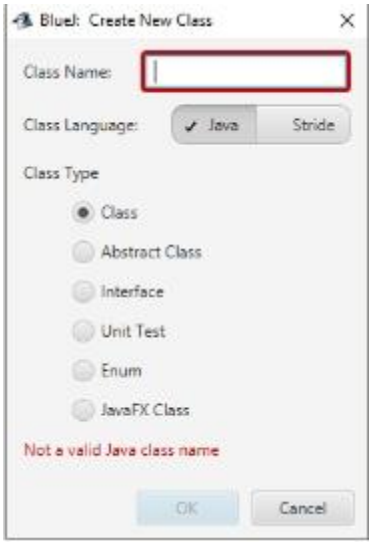


Project Display Area di bagian kanan tengah akan menjadi putih, dan ikon yang terlihat seperti selembar kertas akan muncul.

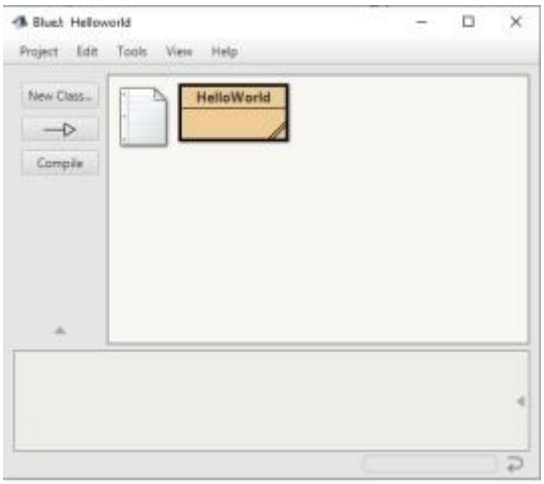


✓ **Buat kelas baru.**

Klik pada tombol **New Class** ... untuk mendapatkan Dialog Buat Kelas Baru. Masukkan "HelloWorld" di kotak teks Nama Kelas , dan pastikan bahwa tombol radio Kelas dipilih. Kemudian klik tombol OK.



Ikon kuning dengan label HelloWorld dan garis diagonal akan muncul.

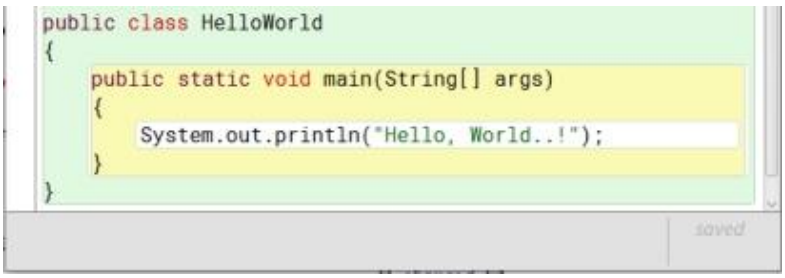


Yang perlu perhatian dalam pemberian nama kelas atau nama lain adalah sebagai berikut :

- ✓ Nama kelas
  Bahwa penggunaan huruf pertama dari semua nama kelas harus memakai huruf besar. Untuk membentuk nama kelas sebaiknya yang singkat, huruf pertama dari setiap kata internal harus dikapitalisasi. Misalnya, nama kelas default adalah: kelas kelasku
- ✓ Sensistive Case
  Java blues sensitif terhadap penggunaan huruf kecil - besar, yang berarti PHI dan Phi memiliki arti yang berbeda dalam Java blues.
- ✓ Nama Metode
  Nama sistem pakailah dengan huruf kecil. Saat beberapa kata dipakai untuk membentuk metode, huruf pertama dari setiap kata internal harus dikapitalisasi. Contoh perjanjian ini adalah sebagai berikut: publik membatalkan myClassMethod()
- ✓ Nama file
  Nama file sistem wajib tidak berbeda dengan nama kelas. Ketika kita menyimpan nama file, kita wajib menyimpannya sebagai nama kelas. Ingat bahwa Java sangat peka terhadap perbedaan huruf besar-kecil dan tambahkan ".java" di akhir nama. Jika nama dokumen dan nama kelas tidak cocok, sistem Anda tidak akan tersambung. Pertimbangkan kelas instan yang disebut Sampel. Dalam hal ini Anda harus memberi nama file sample.java.
- ✓ public static void main(String args[])
  Pelaksanaan proses sistem Java diawali dengan fungsi main(), yang merupakan bagian wajib dari setiap program Java.

✓ **Buka editor untuk kelas.**
Klik dua kali pada ikon kuning HelloWorld untuk membuka editor untuk kelas itu.

✓ **Ubah kode untuk kelas.**
Saat Anda membuka Kelas HelloWorld, Anda akan melihat beberapa kode contoh untuk kelas tersebut, termasuk variabel instance x, konstruktor HelloWorld, dan metode bernama sampleMethod. Kode ini mungkin menarik saat pertama kali Anda melihatnya, tetapi sama sekali tidak berharga. Pilih semua kode dan hapus . Kemudian tambahkan kode Anda. Misalnya, program HelloWorld adalah:



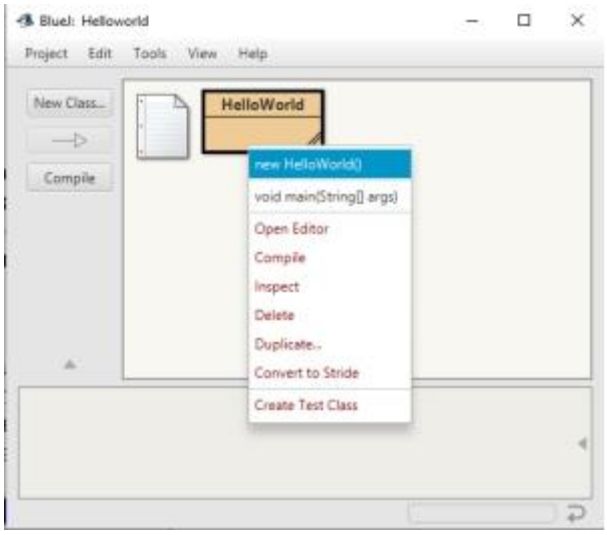
✓ **Kompilasi kelas.**
Klik pada tombol Compile di bagian atas editor. Anda akan mendapatkan pesan "Class compiled - no syntax erors."

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World..!");
    }
}
```

Class compiled - no syntax errors

✓ **Jalankan aplikasinya.**

Kembali ke jendela BlueJ utama. Jika Anda tidak melihatnya di layar desktop, lihat di bagian bawah layar monitor - ada dua jendela BlueJ. Kemudian, klik kanan pada ikon HelloWorld kuning, dan pilih void main (args) untuk menjalankan metode utama.

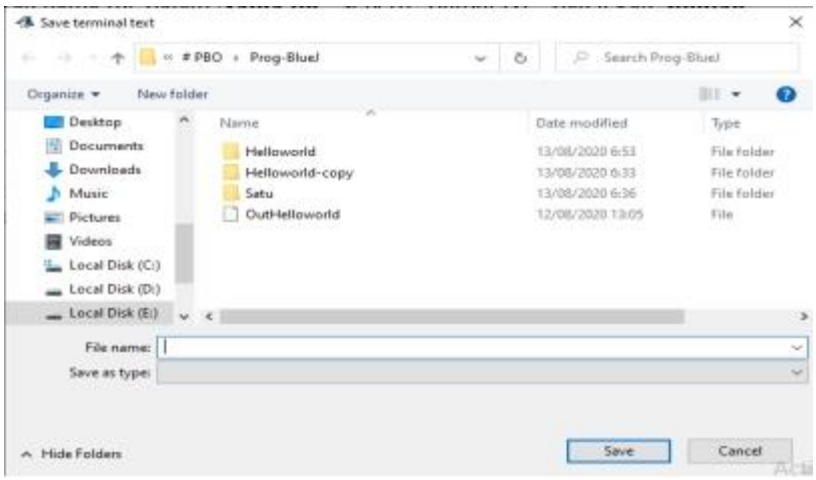


Anda akan mendapatkan Dialog Panggilan Metode. Karena Anda tidak memiliki argumen baris perintah apa pun untuk metode ini, cukup klik OK .

Catatan: Jika program (atau metode) Anda mengambil argumen dan Anda ingin memasukkannya, ketikkan nilai yang dipisahkan dengan koma.

Anda akan mendapatkan Jendela Terminal dengan pesan "Halo, dunia!"

Jika Anda ingin menyimpan output ke file, pilih Options dan Simpan ke file .. . Tentukan nama file dalam Nama file , seperti "output.txt", dan tekan Simpan .



✓ **Keluar dari BlueJ.**

Tutup Jendela Terminal, Editor, dan Jendela BlueJ Utama. Jendela konsol yang terbuka saat Anda memanggil BlueJ akan menutup secara otomatis.

Catatan :

- ✓ Untuk bersihkan layar , supaya hasil yang ditampilkan tidak tampak pada layar bisa ditambhakan dengan perintah “\u000c”
- ✓ Lengakapnya adalah System.out.println (“\u000c”) dibawah public static void main

BAB III
TYPE DATA DALAM BLUEJ

Pada Bab ini membahsa program dasar yang menggunakan konstanta dan variabel, melakukan perhitungan, mendapatkan input dari pengguna, dan menghasilkan output. Materi yang dibahas adalah literal, variabel, tipe data sederhana, kelas String, input, dan output.
Literal, variabel, dan tipe data sederhana adalah konsep yang akan sering ditemukan di sebagian besar bahasa pemrograman yang Anda pelajari.

Literal

Literal adalah sesuatau yang sudah umum bagi program untuk menyertakan konstanta; di Jawa ini disebut sebagai literal. Contohnya meliputi: 123, 123.45, 'a', "BlueJ". Sebagian besar waktu programmer digunakan untuk mengkode literal numerik dan literal boolean dengan cara yang sama seperti biasanya kita menuliskannya (misalnya 123, 123,45, true, false). Tanda kutip tunggal untuk menentukan satu karakter (misalnya 'a'), atau tanda kutip ganda untuk menentukan string teks (misalnya "Buku")

Identifier

Pengenal adalah sekelompok dari huruf yang bisa dipakai untuk menandai peubah, method, class, interface, dan package. Pada pemrograman Java identitas bisa katakan sah jika diawali dengan :

- ✓ Karakter / huruf
- ✓ Simbol Mata Uang
- ✓ Penghubung garis bawah(_)

Pengenal tidak boleh ada unsur @, spasi atau dimulai dengan angka serta tidak di ijinkan menggunakan kata kunci yang tidak boleh dipakai dalam pemrograman java. Selain karakter, Unicode juga dapat digunakan sebagai identifier.

Konstanta

Peubah tetap sama dengan variabel, tetapi Peubah tetap mewakili data tetap yang tidak dapat diubah. Defaultnya harus dinyatakan dan diinisialisasi dalam bentuk perintah yang sama. Final adalah keyword dalam Java untuk mendeklarasikan sebuah konstanta.
Bentuk umum konstanta

```
final tipedata NAMA_KONSTANTA = nilai;
```

Contoh Konstanta

```
final double PHI = 3.14159;
final String KOTA = “Semarang Atlas”;
```

Variabel

Variabel adalah konsep dasar dalam pemrograman. Secara umum, variabel disebut lokasi di memori komputer, dan nilai yang disimpan di lokasi tersebut diperiksa selama eksekusi program. Variabel adalah nama yang diberikan ke bagian dari memori komputer - bagian dari memori yang menyimpan nilai yang dapat diakses dan diubah oleh program selama eksekusi.

Variabel seperti wadah dengan kapasitas terbatas untuk menampung objek yang dikandungnya. Setiap benda tentunya memiliki perilaku sendiri sendiri, contoh benda cair, batu, pohon, hewan, dan seterusnya mereka memiliki sifat yang tidak sama dan tempat yang menampung keberadaannya juga berbeda kapasitas atau daya tampungnya. Ini adalah gambar wadah yang mewakili definisi variabel. Dalam pemrograman komputer, variabel adalah penyimpan informasi yang tidak permanen dan terus berubah. Ini menggunakan aturan yang hampir sama di seluruh pemrograman. Pada pemrograman Java, peubah skrip mengacu pada standar skrip berikut:

- ✓ Variabel ditulis wajib dengan huruf besar atau huruf kecil jika terdiri dari dua kata atau lebih dari , gunakan penyambung dengan garis bawah (_), dan tidak menggunakan spasi, tidak mengizinkan awalan berupa angka dan lambang . lambang %, &, @, ! , ^, #, *, (, dan,) dan beberapa lambang terlarang lainnya. Tidak diperbolehkan menggunakan variabel dari kata kunci Java sendiri
- ✓ Dari ketentuan di atas, penulisan variabel dalam bahasa JAVA berlaku sebagai berikut:

Nama Variabel	Status
Nama atau _nama	Betul
Nama_Mhs	Betul
Nama007	Betul
Nama Mhs	Salah
007Nama	Salah
%nama	Salah

Penulisan peubah dalam pemrograman Java Bluej hampir sama dengan bahasa lain, yaitu mencantumkan tipe data di awal nama variabel. Aturan penulisan variabel tipe data atau lebih tepatnya format penulisan umum di Java Bluej adalah sebagai berikut:

```
tipe_data nama_variabel;
  // Atau dengan inisialisasi nilai dengan struktur
tipe_data nama_variabel = nilai;
```

Contoh

```
int bilangan;
int bilangan = 7;
float lebar = 4.5;
```

Variabel adalah konsep dasar dalam pemrograman. Secara umum, kita mengatakan variabel adalah lokasi bernama dalam memori komputer, dan nilai yang disimpan di lokasi itu dikendalikan selama eksekusi program. Variabel adalah nama yang terkait dengan sebagian memori komputer - sebagaian memori yang menyimpan nilai yang dapat digunakan dan diubah oleh program saat dijalankan.
Bahasa pemrograman Java mengharuskan kita untuk mendeklarasikan tipe data menjadi tipe sederhana yang terkait dengan variabel. Java membuat perbedaan antara tipe data sederhana dan tipe data lain yang didefinisikan melalui kelas. Library kelas Java ada banyak kelas yang telah ditentukan sebelumnya, misalnya: String dan System. Bahasa Java berisi delapan tipe data sederhana: byte, short, int, long, float, double, char dan boolean:

- ✓ **byte, short, int, dan long** digunakan untuk data bilangan bulat (angka tanpa komponen pecahan). Misalnya, 33, 498, -100 adalah bilangan bulat. Tipe data ini berbeda dalam hal besarnya angka yang dapat mereka wakili.
- ✓ **float dan double** digunakan untuk data numerik dengan komponen pecahan seperti seperti: 101.5, 26.334, -55.5.
- ✓ **boolean** digunakan untuk kerja dengan nilai logika benar dan salah. Biasanya digunakan dalam bentuk : true, false.
- ✓ **char** digunakan ketika ada karakter individu yang akan ditangani. Contoh karakter individu adalah 'a', 'b', 'q', '$'. Nilai diapit oleh tanda kutip tunggal.

Type Data Bilangan Utuh

Tipe data ini dipakai untuk nilai numerik di mana tidak ada komponen pecahan - semua nilai adalah bilangan bulat utuh. Jenis ini berbeda sehubungan dengan jumlah memori yang digunakan (dan karenanya nilai minimum dan maksimum):

Tabel 3.1 Type data Bilangan Bulat

Type Data	Minimal	Maksimal
Byte	-128	+127
Short	-32768	+32767
Integer	-2147483648	+2147483647
Long	-9223372036854775808	+9223372036854775807

- ✓ **byte:**

  Type byte adalah jenis bilangan bulat bertanda 8-bit. Nilai maksimum adalah +127. Nilai minimum adalah sama dengan -128. Nilai default yang disimpan dalam variabel jenis ini adalah 0. Jenis informasi byte digunakan untuk menghemat ruang dalam tampilan yang luas, terutama pengaturan angka, karena satu byte berukuran empat kali lebih kecil dari sebuah int.
  Contoh: byte x = 200, byte y = -20

- ✓ **Short**

  Type short ini adalah bilangan berlambang 16-bit. Nilai maksimumnya adalah +32768 dan Nilai minimum adalah -32768. Type Short juga dapat dipakai untuk menghemat memori sebagai jenis informasi byte. Pendek adalah 2 kali lebih kecil dari int Nilai default untuk tipe data ini adalah 0.
  Contoh: short x = 2164, short y = -786

- ✓ **int**

  int information sort adalah nomor pelengkap dua berlambang 32-bit. Nilai maksimum untuk tipe data ini adalah 2^31 -1, yaitu sama dengan 2.147.483.647. Angka ini juga termasuk dalam rentang untuk tipe data ini. Nilai minimum untuk tipe data ini adalah -2^31, yang sama dengan - 2.147.483.648. int sebagian besar digunakan sebagai jenis informasi default untuk kualitas yang sangat diperlukan kecuali jika ada kekhawatiran tentang memori. Nilai default untuk tipe data ini adalah 0. Contoh: int x = 826378, int y = -64782

- ✓ **Long**

  Jenis ini adalah bilangan bulat tambahan berlambang dua 64-bit. Nilai maksimum untuk tipe data ini adalah 9.223.372.036.854.775.807. Nilai minimum untuk tipe data ini adalah -9.223.372.036.854.775.808. Jenis ini digunakan ketika diperlukan rentang memori yang lebih luas daripada int.

Contoh: long x = 174636l, long y = -536452l

Type Data Bilangan Pecahan

Type Data Bilangan Pecahan Ini digunakan untuk mewakili nilai yang memiliki tempat desimal. Tetapi tidak dapat menuliskan pecahan dalam bentuk 1/3 secara lengkap (itu adalah desimal berulang 0,33333 dst.) ada pecahan yang tidak dapat direpresentasikan di komputer. Tipe-tipe ini berbeda sehubungan dengan jumlah digit signifikan yang disimpannya (sekitar 7 untuk float dan 16 untuk double) dan besarnya keseluruhan nilai:

Tabel 3.2 Type Data Bilangan Pecahan

Type Data	Minimal	Maksimal
Float	$\pm 1.4 \times 10^{-45}$	$\pm 3.4 \times 10^{38}$
Double	$\pm 4.9 \times 10^{-324}$	$\pm 1.79769 \times 10^{308}$

Seorang programmer Java mendeklarasikan variabel dalam pernyataan deklarasi, dan kemudian menggunakan nama variabel nanti dalam program untuk menetapkan nilai, untuk mengubah nilai saat ini, dan untuk referensi nilai saat disimpan.

- ✓ **Float**

  Float adalah tipe data untuk bilangan pecahan. float sebagian besar digunakan untuk menghemat memori dalam penyimpanan besar. Jenis informasi float tidak pernah dipakai untuk nilai yang tepat, misalnya, uang. Contoh: float x = 254.3f

- ✓ **Double**

  Double adalah float dengan 64-bit. Tyep double ini sebagian besar digunakan sebagai jenis informasi default untuk kualitas desimal. Contoh: double x = 322,7

Type Char

Karakter adalah karakter tunggal yang didefinisikan oleh tanda kutip tunggal ('). Jenis karakter mengikuti aturan Unicode, jadi Anda dapat menggunakan kode yang diikuti dengan angka antara 0 dan 65535, tetapi angka heksadesimal dari 0000 hingga FFFF biasanya digunakan. Contoh: teks karakter = 'a'

Boolean

Tipe data Boolean merupakan tipe yang paling mudah dalam bahasa pemrograman. Namun demikian Boolean penting dan ada dalam di setiap bahasa pemrograman komputer yang banyak menggunakan Tipe data ini. Boolean merupakan jenis data yang hanya bisa diisi dengan salah satu dari dua nilai: benar atau salah. Jenis data ini sering dipakain untuk pemilihan pada kode program atau memutuskan apa yang harus dilakukan kerjakan ketika suatu kondisi terjadi.

Misalnya, menulis source code untuk menentukan apakah suatu bilangan gasal atau genap berdasarkan masukan user. Untuk melakukan ini, pertama-tama kita perlu memeriksa apakah bilangan tersebut habis dibagi 2 (bilangan genap) atau tidak habis dibagi 2 (ganjil). Tipe data Boolean dapat digunakan untuk menyimpan kondisi seperti itu, yaitu benar atau salah (true or false). jenis informasi boolean yang berbicara dengan beberapa data. Setiap variabel Boolean dapat mengambil salah satu dari dua nilai: benar atau salah Jenis data ini digunakan dalam standar sederhana yang mengikuti kondisi benar/salah. Nilai default untuk tipe data ini adalah false. Contoh: cek boolean = benar;

String

Konsep umum dalam pemrograman, string adalah tipe data yang merupakan kumpulan dari huruf "I", "You", atau "Semarnag". Array juga valid sebagai tipe data string, mis. B. "Saya sedang belajar Java di Semarang".

Di Java, tipe data String dibuat menggunakan kata kunci String. String teks ini juga harus diapit dengan tanda kutip (""). Tipe data string ini dinamis, artinya tidak ada panjang maksimum yang harus ditentukan untuk isi variabel string ini.

Format umum untuk menulis stringString nama_variabel =”isi dari string”

Mendeklarasikan Variabel

Sintaks dasar :

- ✓ [tipe data] [nama variabel]

Perhatikan data dari variabel berikut :

- ✓ int angka;
- ✓ char huruf;
- ✓ float angkadesimal;
- ✓ boolean pernyataan;

{Selanjtnya dideklarasikan variebel tersbut dengan tipe yan sesuai seperti contoh diatas , setelah semau selesai jangan lupa berikan tanda sama dengan (=) }

- ✓ Angka = 20;
- ✓ Huruf = ‘H;
- ✓ Angkadesimal = 22.2f;
- ✓ Pernyataan = true;

{Membuat pernyataan pada tipe data serta memberi kan nilai peubah}

- ✓ int Angka = 20;
- ✓ char Huruf = ‘H’;
- ✓ boolean Pernyataan = true;

{Membuat peubah menjadi nilai tetap hingga tidak dapat diubah lagi isi dari nilainya dengan menambahkan kata kunci sebelumnya}

- ✓ Final int a= 10;
- ✓ Final float bunga = 2,7

{Memberikan visibilitas agar nilai tetap tersebut dapat diakses oleh kelas berbeda tanpa harus membuat objek terlebih dulu, untuk itu dilakukan penambahan modifier umum dan kata kunci static}

- ✓ Public static final a = 10;

Bentuk Umum Variabel

tipedata namaVariabel;

Contoh Deklarasi Variabel

```
int panjang;
double luas;
float kelLing, LsLing;
Int Lbr, Pjg;
```

Tanggal dan Jam Pada Java

Java tidak memiliki kelas Tanggal bawaan, tetapi dapat mengimpor java.time paket untuk bekerja dengan API tanggal dan waktu. Paket ini mencakup banyak kelas tanggal dan waktu.

Berikut adalah tabel java.time untuk memudahkan dalam import:

Tabel 3.3 Java.Time

Class	Keterangan
LocalDate	Represents a date (year, month, day (yyyy-MM-dd))
LocalTime	Represents a time (hour, minute, second and nanoseconds (HH-mm-ss-ns))
LocalDateTime	Represents both a date and a time (yyyy-MM-dd-HH-mm-ss-ns)
DateTimeFormatter	Formatter for displaying and parsing date-time objects

Untuk menampilkan tanggal saat ini, impor java.time.LocalDatekelas, dan gunakan now() metodenya.

Contoh Program

```
import java.time.LocalDate;
public class Utama {
  public static void main(String[] args) {
    LocalDate myObj = LocalDate.now();
    System.out.println(myObj);
  }
}
```

Untuk menampilkan waktu saat ini (jam, menit, detik, dan nanodetik), impor java.time.LocalTimekelas, dan gunakan now()metodenya.

Contoh Program

```
import java.time.LocalTime;
public class Utama {
  public static void main(String[] args) {
    LocalTime myObj = LocalTime.now();
    System.out.println(myObj);
  }
}
```

Untuk menampilkan tanggal dan waktu saat ini, impor java.time.LocalDateTimekelas, dan gunakan now()metodenya:

Contoh Program

```
import java.time.LocalDateTime;
public class Utama {
  public static void main(String[] args) {
    LocalDateTime myObj = LocalDateTime.now();
    System.out.println(myObj);
  }
}
```

Metode ofPattern()menerima segala macam nilai, jika ingin menampilkan tanggal dan waktu dalam format yang berbeda. Berikut adalah tabelnya.

Tabel 3.4 Pattern

Nilai	Contoh
yyyy-mm-dd	"1988-09-29"
dd/mm/yyyy	"29/09/1988"
dd-mmm-yyyy	"29-Sep-1988"
E, mmm dd yyyy	"Thu, Sep 29 1988

Contoh Program

```
import java.time.format.DateTimeLocal;
import java.time.format.DateTimeFormatter;
public class Main {
public static void main(String[] args) {
LocalDateTime myDateObj = LocalDateTime.now();
System.out.println("Sebelum Formatting: " + myDateObj);
DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MMM-yyyy
HH:mm:ss");
String formattedDate = myDateObj.format(myFormatObj);
System.out.println("Sesudah Formatting: " + formattedDate);
 }
}
```

Praktikum Type Data pada BlueJ

Contoh Program

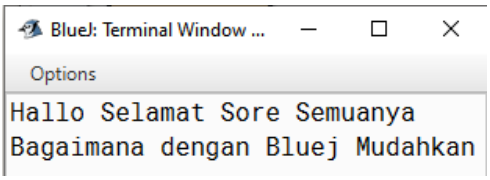
1. Praktikum Variabel

Praktikum dengan menggunakan varibel dan menampilkan variabel tersebut ke layar monitor. Variabel Pesan “Hallo Selamat Sore Semuanya” dan menampilkan data secara langsung “Bagaimana dengan Bluej Mudahkan”

```
public class Salam_Sore
{
    public static void main( String [] args) {
        String pesan ="Hallo Selamat Sore Semuanya";
        System.out.println(pesan);
        System.out.println("Bagaimana dengan Bluej Mudahkan");}

}
```

- ✓ Buat class dengan nama salam_Sore, Jika sudah masuklah ke dalam editor programnya , hapus yang tidak perlu pada tampilan awal, selanjutnya ketikan kode program diatas.
- ✓ Jika semua kode program sdh di ketikan , selanjutnya cek dengan perintah **Compile**

- ✓ Jika di cek dengan tombol perintah compile dan hasilnya adalah “"Class compiled - no syntax erors.", berati program sudah tidak ada yang salah.
- ✓ Selanjutnya bisa di eksekusi dengan klik kanan pada nama class tersbut dan pilih void main, jika benar hasilnya akan tampil sperti gambar dibawah ini.

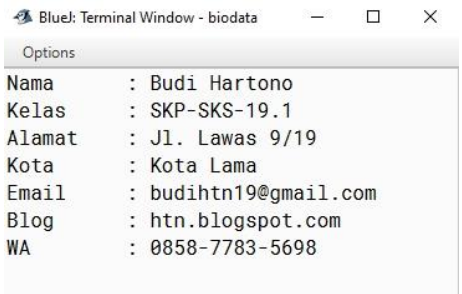


2. Praktikum Data Diri

menampilkan data secara langsung kelayar monitor dengan perintah system out print atau println. Println menampilkan data dengan baris baru sedangkan print menampilkan data dan data ke-2, ke-3, dst ditempatkan disebelah kanan. Untuk hasil program di atas dapat dilihat pada gambar dibawah.

```
public class Datadiri
{
    // instance variables - replace the example below with your own
    private int x;

    public Datadiri()
    {
        // initialise instance variables
        x = 0;
        System.out.println("Nama      : Budi Hartono");
        System.out.println("Kelas     : SKP-SKS-19.1");
        System.out.println("Alamat    : Jl. Lawas 9/19");
        System.out.println("Kota      : Kota Lama");
        System.out.println("Email     : budihtn19@gmail.com");
        System.out.println("Blog      : htn.blogspot.com");
        System.out.println("WA        : 0858-7783-5698");
    }
}
```

- ✓ Buat class dengan nama Data_Diri, Jika sudah masuklah ke dalam editor programnya , hapus yang tidak perlu pada tampilan awal, selanjutnya ketikan kode program diatas.
- ✓ Jika semua kode program sdh di ketikan , selanjutnya cek dengan perintah **Compile**
- ✓ Jika benar hasilnya dapat dilihat seperti contoh dibawah ini



3. Praktikum Variabel dan Konstanta

a) Hitung Luas Segi Empat

Program latihan menghitung luas segi empat (4) dengan menggunakan variabel secara langsung variabel lebar , panjang dan luas dengan type int semua. Dengan variabel di lebar dan panjang inputkan secara langsung sedangkan variabel luas hasil dari perkalian panjang dan lebar

```
public class Luas
{
    public static void main (String [] args)
    {
      // Variabel Lebar
      int lebar = 10;
      //Variabel Panjanng
      int panjang = 20;
      //Variabel Luas
      int luas;

      System.out.println("Masukan Berapa Lebar    : " + lebar);
      System.out.println("masukan Berapa Panjang) : " + panjang);
      luas = lebar * panjang;
      System.out.println("Jadi luas segi Empat adalah :" + luas);
    }
}
```

Buat class baru dengan nama Luas dan selanjutnya bisa Anda ketikan kode program seperti contoh diatas, setelah selesai cek program tersebut dengan compile sampai muncul no-syntac error, jika sudah jalankan program tersebut dan lihat hasilnya sperti contoh dibwah ini

b) Hitung Luas Lingkaran

```
public class Lingkaran
{
    public Lingkaran()
    {
        // Deklarasi konstanta
        final double PI = 3.14159;
        // Variabel standart
        int R = 10;
        double Luas, Kel ;
        System.out.println("Masukkan Nilai Jari-2  :" + R);
        Luas = PI * R * R;
        Kel  = 2 * PI * R;
        System.out.println("Jadi Luas Lingkaran Adalah      : " + Luas);
        System.out.println("Jadi Keliling Lingkaran Adalah  : " + Kel);
    }
}
```

Program latihan menghitung luas dan keliling lingkaran ini hampir sama dengan program sebelumnya yaitu luas segi empat, vaiabelnya adalah jari-jari dengan type int, yang membedkan pada luas dan keliling lingkaran ini ada deklarasi konstanta (nilai tetap) yaitu PI dengan nilai 3.14
.

c) Hitung Luas Segi Tiga

```
public class Segitiga
{
    public Segitiga()
    {
    int Alas = 10, Tinggi = 15;
    double Luas ;
    System.out.println ("Berapa Tinggi Segitiganya.....=" + Tinggi );
    System.out.println ("Berapa Alas Segitiganya.....=" + Alas);
    Luas = 0.5 * Alas * Tinggi;
    System.out.println ("Jadi Luas Segitiga Adalah.....=" + Luas);
    }

}
```

Pada hitung luas segi tiga ini hampir mirip dengan yang sebelumnya maksdunaya adalah segi empat dimana pada variabel tinggi dan alas dengan type integer sedangkan pada variabel luas dengan type double ini yang membedakan dengan luas pada segi empat yang typenya integer. Pada luas segitiga ini hasilnya pada luas aka ada desimalnya 75.0 beda dengan yang sebelumnya 200 dimana pada akan 200 tidak ada angka dibelakang koma alias tidak ada desimalnya.

BAB IV
OPERATOR

Operator merupakan suatu lambang yang dipakai dalam memberikan instruksi kepada komputer untuk melaksanakan perintah kerja pada suatu atau beberapa operand. Atau Operator merupakan lambang tersendiri untuk memberikan instruksi kepada penerjemah untuk melaksanakan suatu tindakan terhadap sejumlah operand. Instruksi suatu operasi didifinisikan oleh operator yang lambang operandnya berupa peubah , variabel tetap operand sendiri merupakan sesuatu yang dioperasikan oleh operator. Operator ini mengikuti skala dari prioritas yang pasti sehingga compeler tahu mana saja dari operator yang ada yang harus didahulukan dalam permaslahan dari beberapa operator yang digunakan secara bersama-sama dalam satu statemen.

Operator sendiri diibagi menjadi 5 yaitu :

- ✓ Operator unary : Operator yang mengolah satu operand
- ✓ Operator prefix : Operator yang ditempatkan sebelum operand
- ✓ Operator biner infiks : Operator yang ditempatkan diantara dua operand
- ✓ Operator postfiks: Operator yang ditempatkan setelah operand .
- ✓ Operator ternary : Operator yang membutuhkan tiga operand

Pengertian Operand dan Operator

Ada baiknya kita masuk telebih dahulu ke bentuk bentuk operator di dalam bahasa Java, dalam bahasa jawa dikenal dengan istilah *operand* dan *operator*.

Operand meruapakan angka sumber dari nilai yang digunakan dalam sebuah proses operasi. Operator merupakan perintah yang dipakai untuk mendapatkan hasil dari suatu proses yang ada. Pada prinsifnya operator dapat berjenis karakter atau matematik atau perintah singkat singkat. Ambil contoh, pada suatu operasi : 12 + 3. Angka 12 dan angka 3 biasa keita sebut dengan operand, sedangkan lambang tambah (karakter +) merupakan operator.

Operator Unary, Binary dan Ternary

Macam macam operand-nya, dapat digolongkan menajdi beberapa diatantaranya adalah : Operator Unary, Operator Binary dan Operator Ternary.

- ✓ Operator Unary merupakan suatu operator yang hanya memiliki satu(1) operand. Umpama operator plus(positif) : +9, +5, +10.15
- ✓ Operator Binary merupakan suatu operator yang terdapati dua(2) operand. Kebanyakan operator yang ada masuk ke dalam golongan operator binary. Contoh operator aritmatika : 5 + 10, 7 * 5 , 9 % 3, dll.
- ✓ Operator Ternary merupakan suatu operator yang terdiri atas tiga (3) operand. Bahasa Java memiliki 1 operator ternary, yakni ” ? : ” misal (a == 1) ? 20: 30.

Jenis Jenis Operator

Berikut ini adalah macam macam operator pada bahasa pemrograman Java :

- ✓ Operator Aritmatika
- ✓ Operator Increment dan Decrement
- ✓ Operator Perbandingan / Relasional
- ✓ Operator Logika / Boolean
- ✓ Operator Bitwise
- ✓ Operator Assignment
- ✓ Operator Type Comparison
- ✓ Operator Ternary

Daftar di atas dapat bervariasi karena metode pengelompokan yang digunakan, misalnya operator penjumlahan atau pengurangan terkadang termasuk dalam operator aritmatika. Kami akan membahas secara singkat arti dari operator ini.

Operator Penugasan / Assignment

Operator Assignment adalah operator yang dipakai untuk meberikan suatu nilai ke variabel. Setelah variabel dideklarasikan, Anda dapat menggunakan operator penugasan untuk memberikan nilai padanya. Pada lambang lambang tertentu seperti lambang sama dengan (=) dipakai sebagai sebuah operator assignment. Umpama operator assignment:

Contoh operator penugasan:

- ✓ int x = 5; // berarti menugaskan 5 kepada variabel x
- ✓ double luas = 2.5; // berarti menugaskan 2.5 kepada variabel luas
- ✓ x = 5 * (4 + 3); // Tetapkan x nilai ekspresi
- ✓ Variabel juga dapat digunakan dalam ekspresi, variabel ini dapat digunakan di dalam suatu ekspresi. Misalkan : x = x + 1;
- ✓ Operator assignment dapat dipakai secara series, misalnya: p = q = r = 15;

Operator Aritmatika

Operator aritmatika merupakan sutau operator yang biasanya dipakai untuk operasi yang bersisifat matematis. Sedangkan aritmatika merupakan bagian ilmu matematika yang mempelajari khususnya perhitungan sederhana seperti perkalian, pembagian, penambahan, dan pengurangan. Selain operasi yang disebutkan diatas, Java juga memiliki operasi pembagian modulus, atau operator %, yang digunakan untuk mencari sisa. dari hasil bagi. Jadi, operator aritmatika adalah:

Perkalian (*), Penjumlahan (+), pengurangan (-),pembagian (/), dan modulus

(%).**Tabel 4.1 Operator Aritmatika**

Operator	Arti	Contoh	Hasil
+	Penjumlahan	5 + 5	10
*	Perkalian	5 * 2	10
-	Penguruangan	5 – 2	3
/	Pembagian	4 / 2	2
%	Modulus	5 / 2	1

Jika dua operan dari operator pembagian adalah bilangan bulat, hasilnya adalah bilangan bulat. Pelanggaran dibatalkan. Misalnya, 5/2 akan sama dengan 2, bukan 2,5.

Operator Increment dan Decrement

Operator menaikan dan penurunan meruapkan nama untuk operasi seperti a++ dan a--. Ini sebenarnya kepanjangan dari operasi a++ (a = a + 1) serta a—(a = a – 1).

Increment dipakai guna menaikan variabel sebanyak 1 point adapun decrement dipakai untuk menurunkan variabel sebanyak satu angka (-1). Pada saat menggunakan, lambang plus dipakai dua (2) kali untuk penambahan dan tanda negatif dua (2) kali untuk pengurangan. Penempatan lambang plus atau negatif bisa dipakai pada awal, seperti ++a dan --a, atau juga dapat dipakai pada akhir variabel, seperti umpama. a++ dan a--.

Kesmipulannya adalah ada empat (4) penambahan dan pengurangan dalam bahasa Jawa.

Tabel 4.2 Increment dan Decrement

Operator	Penjelasan
Pre Increment ++a	Tambah a Sebanyak 1 Angka, lalu tampilkan hasilnya
Pre Increment a++	Tambah nilai a , lalu tambahkan a sebanyak 1 Angka
Pre Increment --a	Kurangi a Sebanyak 1 Angka, lalu tampilkan hasilnya
Pre Increment a--	Kurango nilai a, lalau kurangi a sebanyak 1 angka

Operator Perbandingan / Relasional

Operator Relasional dipakai untuk meng-compare dua(2) nilai dan hasil yang didapatkan bisa sama, lebih kecil dari, lebih besar dari dll. Hasil dari operator compare ini berupa logika benar atau logika salah. Java memiliki beberapa operator pembanding , untuk lebih jelasnya dapat Anda lihat pada tabel berikut sebagai tabel untuk membandingkan dua nilai.

Tabel 4.3 Operator Relasional

Operator	Nama	Contoh	Hasil
<	Lebih Kecil	5 < 5	Salah
>	Lebih Besar	7 > 5	Benar
>=	Lebih Besar sama dengan	5 >= 5	Benar
<=	Lebih Kecil Sama dengan	5 <= 4	Salah
==	Sama dengan	5 == 5	Benar
!=	Tidak sama dengan	5 != 5	Salah

Operator Logika / Boolean

Operator logika / nalar dapat mempunyai satu , dua atau lebih operator Boolean yang dapat menghasilkan nilai logika, yaitu. Operator logika dipakai untuk mengembalikan nilai logika benar atau logika salah dari dua kondisi atau lebih. Terdapat enam (6) operator logika dalam bahasa Java, antara lain : && (logika DAN), & (logika DAN), || (ATAU logika), | (logika OR inklusif), ^ (logika eksklusif ATAU) dan ! (Not logika).

Tabel 4.4 Operator Logika / Boolean

Operator	**Nama**	**Penjelasan**
&&	Logika And	Akan menghasilkan kondisi true jika kedua operand true
&	Boolean logika And	Akan menghasilkan kondisi true jika kedua operand true
||	Logika Or	Akan menghasilkan kondisi true jika salah satu operand true
|	Boolean Logika Or	Akan menghasilkan kondisi true jika salah satu operand true
^	boolean logika inclusive	Akan menghasilkan kondisi true jika salah satu operand true atau false (berbeda)
!	Not	Akan menghasilkan kondisi true jika operand false

Rumus :

- ✓ Operator && hanya mengembalikan true jika kedua operan benar, jika tidak hasilnya salah.
- ✓ Operator || mengembalikan false hanya jika kedua operan bernilai false, jika tidak mengembalikan true.
- ✓ Operator ^ Hasil dari operator eksklusif OR adalah mempunyai nilai benar jika salah satu dari kedua operator tersebut mempunyai nilai logika yang tidak sama.
- ✓ Operator! Balikkan logikanya, !false menjadi true, !true menjadi false.

Operator AND menghasilakan nilai "Benar", jika kedua operan terdapat suatu nilai logika BENAR. Berikut tabel kebenaran AND

Tabel 4.5 Tabel Logika untuk Operator And

Operasi Boolean AND (&&)		
Logika A	Logika B	Hasil = A && B
True	True	True
True	False	False
False	True	False
False	False	False

Operator OR menghasilakan nilai “Benar” jika salah satu operan tersebuat ada yang bernilai benar Berikut adalah tabel logika OR:

Tabel 4.6 Tabel logika untuk Operator Or

Operasi Boolean OR (||)		
Logika A	Logika B	Hasil = A || B
True	True	True
True	False	True
False	True	True
False	False	False

Hasil dari operator OR eksklusif akan bernilai “Benar” jika dari kedua operan tersebut memiliki nilai nilai logika boolean yang berbeda. Dibawah ini disajikan suatu tabel logika untuk OR eksklusif:

Tabel 4.7 Tabel logika untuk Operator Exclusive

Operasi Boolean Exclusive (^)		
Logika A	Logika B	Hasil = A ^ B
True	True	False
True	False	True
False	True	True
False	False	False

Operatot pembalika atau yang lebih dikenal dengan sebutan NOT logika biasanay dipakai dalam pernyataan di mana argumen bisa deklarasi, peubah dan peubah tetap. Berikut merupakan tabel logika untuk NOT

Tabel 4.8 Tabel kebenaran untuk Operator Not

Operasi Boolean NOT (!)		
Logika A	Logika B	Hasil = NOT
! True		False
! False		True

Operator Bitwise

Operator Bitwise merupakan operator tersendiri yang melakukan operasi logika suatu bilangan biner dalam bentuk bit. Bilangan bit sendiri merupakan macam bilangan yang terdiri dari 2(dua) bentuk, yaitu 0 dan 1. Jika nilai sumber yang dipakai bukan biner, penerjemah Java secara otomatis mengkonversi menjadi biner. Contoh: 8 desimal = 1000 dalam format biner.

Dalam praktiknya, operator ini jarang digunakan kecuali jika Anda membuat program yang perlu memanipulasi bit komputer. Operator ini juga cukup kompleks dan membutuhkan pemahaman tentang sistem bilangan biner.

Tabel 4.9 Daftar lengkap BItwise

Operator	Nama	Contoh	Binner	Hasil Binner	Hasil Desimal
&	And	10 & 12	1010 & 1100	1000	8
|	Or	10 & 12	1010 & 1100	1110	14
^	Xor	10 ^ 1	1010 ^ 1100	0110	6
!	Not	!10	!1010	0101	-11

Operator Type Comparison

Operator Type Comparison merupakan operator perbandingan tipikal adalah nama untuk operator khusus yang dipakai untuk memeriksa objek. Di Java, operator ini menggunakan perintah instanceof.

Operator Ternary

Operator ternary meruapkan suatu operator dengan tiga (3) operan. Di Java, operator ternary ini menggunakan lambang ? : dan ini meruapakan kepanjangan dari kondisi if-else. Operator ini menggunakan lambang tanya (?) dan titik dua (:) untuk memisahkan tanggapan.

Operator Bersyarat (? :):

Operator kondisional adalah merupakan suatu operator ternary yang berisi tiga operan. Pada prinsifnya, operator ini dipakai untuk evaluasi ekspresi boolean. Operator menguji operan atau kondisi pertama dan jika hasilnya benar, maka nilai kedua diberikan ke peubah. Apabila kondisinya salah, operan ketiga ditugaskan ke variabel.

Sintaks operator ini adalah sebagai berikut:

<ekspresi logis> ? <nilai benar> : <nilai salah>

di mana

<ekspresi logis> adalah beberapa ekspresi logis yang mengevaluasi benar atau salah

<true value> adalah hasil ketika ekspresi logika benar

<false value> adalah hasil ketika ekspresi logika salah

Operator kondisional menghasilkan hasil yang berupa <nilai benar> atau <nilai salah>.

Misalkan kita ingin menetapkan variabel maxValue maksimum dari dua variabel, x dan y. Ini mudah dilakukan dengan pernyataan penugasan di mana sisi kanan dikodekan dengan operator kondisional, seperti pada:

nilai maks = (x>y) ? x : y ;

Misalkan Anda perlu menampilkan yang lebih besar dari dua nilai x dan y. Untuk melakukan ini, kita dapat menggunakan ekspresi di atas di dalam pernyataan tampilan., seperti pada:

System.out.println("terbesar adalah "+ ((x>y) ? x : y))

Prioritas Operator

Java memberikan prioritas kepada setiap operator dan kemudian menggunakan prioritas tersebut untuk mengendalikan urutan evaluasi ekspresi. Operator dengan prioritas lebih tinggi di eksekusi sebelum operator dengan prioritas lebih rendah. Terkadang seorang progrmaer perlu mengasmpingkan prioritas ini dan akan menggunakan sub ekspresi untuk tujuan itu, sebuah sub ekspresi selalu di evaluasi sebelum ekspresi yang memuatnya di evaluasi. Perhatikan tabel prioritas operator berikut ini.

Prioritas Operator
Tinggi ke Rendah
*/ %
+ -

Perkalian diberikan prioritas yang sama dengan pembalian atau modulo dan penambahan diberikan prioritas yang sama dengan pengurangan, namun prioritas perkalian, pembagian,dan modulo lebih tinggi dari pada penjumlahan dan pengurangan. Tabel berikut menunjukan ekspresi , urutan evaluasi yang di tunjukan dengan sub ekspresi yang setara dan hasil akhir.

Ekspresi Java yang melibatkan prioritas		
Ekspresi	Evaluasi akhir yang setara	Hasil
9.0 /5.0 + 32.0	(9.0 / 5.0) + 32.0	33.8
105 – 105 % 10	105 – (105% 10) 1	100
1 + 5 * 2	(5 * 2) + 1	11

Penerapan Operator dalam Program

1. Contoh Operator Aritmatika

Contoh Program untuk operator aritmatika untuk seperti penjumlahan, perkalian, pengurangan dan pembagian serta menampilkan hasil dari operator tersebut.

Buatlah class Aritmatika dan selanjutnya tuliskan kode programnya seperti dibwah ini, jika sudah selesai silahkan di compile dan eksekusi maka hasilnya akan terlihat seperti dibawah ini.

```
public class Artimatika
{
    public static void main(String [] args)
    {
        //buat variabel
        int a=10, b= 5, c= 15;
        //Hitung
        int hasil1 = a + b - c;
        int hasil2 = a * b / c;
        int hasil3 = a + b * c;
        int hasil4 = a + b / c;
        int hasil5 = (a + b) * c;
        int hasil6 = (a - b) * c;
        int hasil7 = a * (b - c);
        //Cetak
        System.out.println(hasil1);
        System.out.println(hasil2);
        System.out.println(hasil3);
        System.out.println(hasil4);
        System.out.println(hasil5);
        System.out.println(hasil6);
        System.out.println(hasil7);
    }
}
```

```
BlueJ: Terminal Window - Operator
Options
0
3
85
10
225
75
-100
Can only enter input
```

2. Contoh Operator Increment dan Decrement (Naik Turun)

Contoh Program untuk operator penaikan dan penurunan serta menampilkan hasil dari operator tersebut. Buat class baru dengan nama Class Naik_Turun, selanjutnya ketikan potongan kode program tersebut seperti contoh dibawah dan lihat juga hasilnya.

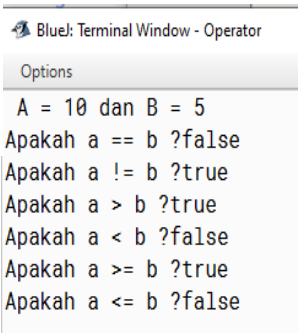
```
public class Naik_Turun
{
    public static void main ( String [] args)
    {
        int a = 8;
        int b = a++;
        int c = ++a;
        int d = a--;
        int e = --a;
        System.out.println("A = " + a);
        System.out.println("B = " + b);
        System.out.println("C = " + c);
        System.out.println("D = " + d);
        System.out.println("E = " + e);
    }
}
```

```
BlueJ: Terminal Window - Operator
Options
A = 8
B = 8
C = 10
D = 10
E = 8
Can only enter input
```

3. Contoh Operator Pembanding

Buat class Pembanding untuk menguji perbandingan dalam operator pembading, membandingkan variabel a dan variabel sama, tidak sama , lebih besar dan lebih kecil serta menampilkan hasil dari operator pembanding tersebut.

```
public class Pembanding
{
    public static void main (String [] args){
    System.out.print('\u000c');
    int a = 10;
    int b = 5;
    boolean hasil;
    hasil = a == b;

    System.out.println (" A = 10 dan B = 5");
    System.out.println ("Apakah a == b ?" + hasil );

    hasil = a != b;
    System.out.println ("Apakah a != b ?" + hasil );

    hasil = a > b;
    System.out.println ("Apakah a > b ?" + hasil );

    hasil = a < b;
    System.out.println ("Apakah a < b ?" + hasil );

    hasil = a >= b;
    System.out.println ("Apakah a >= b ?" + hasil );

    hasil = a <= b;
    System.out.println ("Apakah a <= b ?" + hasil );
    }
```



4. Operator Boolean Logika

Buat class BelajarJava untuk menguji boolean logika dalam operator boolean logika, membandingkan variabel a dan variabel b, tidak sama serta menampilkan hasil dari operator pembanding tersebut.

```
class BelajarJava {
  public static void main(String args[]){

    boolean a = true;
    boolean b = false;
    boolean hasil;

    hasil = a && b;
    System.out.println("Hasil dari a && b : " + hasil );

    hasil = a || b;
    System.out.println("Hasil dari a || b : " + hasil );

    hasil = !b;
    System.out.println("Hasil dari !b : " + hasil );

  }
}
```

5. Operator Bitwise

Buat class BelajarJava untuk menguji operator bitwise, membandingkan variabel a dan variabel b, tidak sama , lebih besar dan lebih kecil serta menampilkan hasil dari operator pembanding tersebut.

```
class BelajarJava {
  public static void main(String args[]){

    int a = 10;
    int b = 12;
    int hasil;

    hasil = a & b;
    System.out.println("Hasil dari a & b : " + hasil );

    hasil = a | b;
    System.out.println("Hasil dari a | b : " + hasil );

    hasil = a ^ b;
    System.out.println("Hasil dari a ^ b : " + hasil );

    hasil = ~a;
    System.out.println("Hasil dari ~a : " + hasil );

    hasil = a << 1;
    System.out.println("Hasil dari a << 1 : " + hasil );

    hasil = a >> 1;
    System.out.println("Hasil dari a >> 1 : " + hasil );

  }
}
```

Kesimpulan

- ✓ Operan adalah nilai sasal yang digunakan dalam pelaksanaan operasi.
- ✓ Operator adalah perintah yang digunakan untuk memeperoleh hasil dari suatu proses.
- ✓ Macam - macam operator dalam bahasa pemrograman Java ada Operator penugasan, operator aritmatika, operator penjumlahan dan pengurangan, operator perbandingan/relasi, operator logika/logika, operator dua arah, operator pembanding biasa, operator ternary
- ✓ Adapuan prioritas urutan dalam operator adalah *, /, +, -

BAB V
INPUT DARI KEYBOARD

Susutu sistem dari program komputer pada umum terdiri atas Elemen : Inputan , elemen Proses, dan elemen Cetak atau keluaran.

- ✓ Masukan : meruapakan suatu nilai yang diinputkan ke dalam program komputer
- ✓ Proses: merupakan urutan dari suatu tindakan yang dilakukan untuk memenajemen masukan menjadi sesuatu yang bernilai
- ✓ Keluaran t: adalah meruapak suatsu dari hasil proses pengolahan yang bernilai

Oleh karena itu, input menyediakan data melalui keyboard, mouse, pemindai, dan mikrofon. Bab ini hanya mencakup input keyboard. Di Java Blues, ada beberapa cara untuk mendapatkan input keyboard, antara lain:

Kelas pemindai, kelas BufferedReader, dan GUI

- ✓ Masukan berorientasi pada tDOS memakai Scanner
- ✓ Masukan berorientasi memakai BufferedReader
- ✓ Masukan berorientasi pada GUI memakai JOptionPane

Perbedaan Scanner, BufferedReader dan GUI

- ✓ Pada scanner, untuk melakukan proses matematis, peubah yang dipakai dalam perhitungan tidak perlu diubah lagi, dapat langsung diproses atau dihitung.
- ✓ Fungsi pada (InputStreamReader + BufferedReader) masih harus dilakukan sedikit perubahan terlebih dahulu sebelum melakukan perhitungan matematis dengan peubah yang digunakan.
- ✓ JOptionPane, paket javax.swing yang dipakai untuk masukan dan keluaran berbasis GUI Swing. Tampilannya memudahkan user , karena tampilannya kotak dialog atau messagebox

Kelas Scanner

Kelas Scanner merupakan sebuah kelas yang disertakan dalam paket java.util dan bertindak sebagai kelas untuk untuk proses masukan dari mode DOS. Kelas Scanner sendiri ada dalam paket java.util, jadi ketika kita akan menggunakannya maka wajib mengambilnya terlebih dahulu atau menuliskan perintah sebelum deklarasi kelas dengan rumus sebagai berikut:

```
Import java.util.Scanner;
```

Langkah-langkah menggunakan kelas Scanner :

- ✓ Pertama adalah dengan cara import kelas Scanner yang ada pada pada paket java.util
  - o import java.util.Scanner;
- ✓ Langakah kedua adalah dengan membuat objek referensi sebagai alat untuk memasukan data.
  - o Scanner objekReferensi = new Scanner(System.in);
  - o Contoh:
    - ▪ Scanner input = new Scanner(System.in);
    - ▪ Scanner scan = new Scanner(System.in);
- ✓ Selanjutnya adalah menggunakan method tersendiri untuk melakukan masukan data melalui objek referensi yang sebelumnya sudah dibuat.

Bentuk pernyataan secara umum pemakaian kelas Scanner adalah seperti tampilan dibawah ini :

```
Scanner bacaInput = new Scanner(System.in);
```

Sintaks Pemindai (System.in) baru membuat objek bertipe Pemindai. sintaks pemindai readInput memberikan pernyataan bahwa peubah readInput adalah variabel bertipe Scanner. Objek ReadInput bisa memanggil salah satu dari metode kelas Scanner yang akan digunakan untuk read dari berbagai jenis data inputan. Untuk informasi selengkapnya, lihat tabel berikut:

Tabel 5.1 Method input pada kelas Scanner

Metode	Penjelasan
nextBigDecimal()	menampung Nilai bilangan pecahan BigDecimal
nextBigInteger()	() Nilai bilangan bulat BigInteger
nextBoolean()	Nilai berupa boolean
nextByte()	Nilai bilangan bulat byte
nextDouble()	Nilai bilangan pecahan double
nextFloat()	Nilai bilangan pecahan float
nextInt()	Nilai bilangan bulat int
nextLine()	Data berupa String
nextLong()	Nilai bilangan bulat long
nextShort()	Nilai bilangan bulat short
next()	Membaca suatu string yang berakhir dengan karakter spasi

Contoh Penggunaan method input

```
Scanner input = new Scanner(System.in);

String nama = input.nextLine();

int alas = input.nextInt();

float tinggi = input.nextFloat();

boolean status = input.nextBoolean();
```

Kelas BufferedReader

Kelas BufferedReader ada di paket java.io untuk menerima masukan dari keyboard dengan rumus : berikut: *BufferedReader dataIn = new BufferedReader (new InputStreamReader(System.in));*

BufferedReader merupakan sebuah kelas yang bekerja mirip dengan kelas Pemindai, ini merupakan kelas input keyboard berorientsi pada text. Tetapi konsep BufferedReader tidak sama dengan kelas Scanner.

Langkah-langkah menggunakan BufferedReader:

- ✓ Pertama adalah dengan mengambil kelas InputStreamReader, IOException dan BufferedReader.
- ✓ Berikutnya dengan melakukan pembuatan objek referensi sebagai media untuk entri data.
- ✓ Selanjutnya adalah dengan memanggil metode readLine().
- ✓ Entri pada poin 2 dan 3 di blok try-catch.

Langkah-langkah memakai penggunaan kelas BufferedReader :

- ✓ Pertama adalah dengan mengambil kelas InputStreamReader, IOException dan BufferedReader.
  - import java.io.BufferedReader;
  - import java.io.IOException;
  - import java.io.InputStreamReader;
- ✓ Langkah selanjutnya dengan membuat objek referensi
  - BufferedReader objekReferensi = new
  - BufferedReader(new InputStreamReader(System.in));

  Contoh:
  - BufferedReader input = new
  - BufferedReader(new InputStreamReader(System.in));
  - BufferedReader read = new
  - BufferedReader(new InputStreamReader(System.in));
- ✓ Panggil metode ReadLine() dengan objek referensi. Yang perlu diperhatikan adalah, hasil dari kelas buffer adalah dalam bentuk text . Jadi jika ingin mengubahnya menjadi format dengan tipe data yang lain, maka harus mengkonversi data tersebut.
- ✓ Di blok try-catch, tulis poin 2 dan 3, yang akan ditangkap oleh IOException.

```
try {
BufferedReader input = new
BufferedReader(new InputStreamReader(System.in));
System.out.print("Input Nilai X : ");
double x = Double.parseDouble(input.readLine());
} catch (IOException e) { }
```

5.2 Tabel konversi data (parsing)

Konversi ke	Cara konversi
boolean	Boolean.parseBoolean(String)
float	Float.parseFloat(String)
double	Double.parseDouble(String)
byte	Byte.parseByte(String)
short	Short.parseShort(String)
int	Integer.parseInt(String)
long	Long.parseLong(String)

Contoh Penggunaan parsing

- ✓ double x=Double.parseDouble(input.readLine());
- ✓ int y=Integer.parseInt(input.readLine());
- ✓ Boolean z=Boolean.parseBoolean(input.readLine());

Menggunakan GUI (JoptionPane)

- ✓ Manfaat dari kelas JOptionPane tidak hanya untuk laporan, tetapi bisa juga digunakan sebagai fungsi entri. Langkah-langkah menggunakan JOptionPane:
- ✓ Impor atau ambil kelas JOptionPane dari dalam paket javax.swing.
- ✓ Buat referensi objek dari kelas string tersebut, kemudian panggil metode showMessageDialog() tadi dengan referensi objek.
- ✓ Hasil dari pemanggilan ShowInputDialog() adalah sebuah text. Jadi jika ingin mengkonversinya menjadi jenis tipe data lain, maka harus melakukan konversi atau pengalihan bentuk data tersebut.
- ✓ Masukan melalui JoptionPane Impor kelas JOptionPane ke dalam paket javax.swing.
  - ~ impor javax. mengayun JOptionPane;
  - ~ Pembuatan objek referensi
  - ~ String objectReference = JOptionPane.showInputDialog(null, String);
- ✓ Contoh:
  - ~ Input string = JOptionPane.showInputDialog(null, "Contoh teks");
  - ~ String readJOp = JOptionPane.showInputDialog(null, "Masukkan data");
- ✓ Hasil dari pemanggilan ShowInputDialog() adalah sebuah string. Jadi jika kita ingin mengubahnya menjadi bentuk tipe data yang lain, kita harus melakukan beberapa analisis (transform data).
  - ~ Input string = JOptionPane.showInputDialog(null, "Input angka");
  - ~ double x = Double analysisDouble(result);

Output Terformat

Pemformatan harus dilakukan agar tampilan nilai mudah dibaca oleh pengguna. Pemformatan tampilan nilai dapat menggunakan metode printf ini. Sintaks untuk menulis metode printf dari kelas System di Java adalah sebagai berikut:

System.out.printf(strFormat, penentuFormat);

Argumen ke parameter StrFormat merupakan test yang bisa berisi subtext dan atribut. Hasil akhir dari format tersebut menentukan jenis format objek yang akan dihasilkan. Bagian ini dapat berupa nilai seperti nilai numerik, nilai karakter, nilai logika atau string. Pada tabel di bawah ini adalah tabel definisi format yang dapat dipakai untuk menentukan jenis pembentukan untuk item yang ditampilkan.

5.3 Tabel Format (Specifier)

Penentu Format (Specifier)	Tampilan	Contoh
%b	Nilai Boolean	true atau false
%c	Karakter	'c'
%d	Desimal integer	350
%f	Pecahan decimal	243.45
%e	Numerik dalam notasi ilmiah	3.457000e+01
%s	String	"Selamat Datang Java"

5.4 Tabel Format untuk Angka

Nilai	Pattern	Output	Keterangan
123456.789	###,###.###	123,456.789	Tanda pagar (#) menunjukan digit , koma adalah tempat penamung untuk pemisah pengelompokan dan titik adalah penampung pemisah desimal
123456.789	###,###	123456.789	Nilai memiliki tiga digit disebelah kanan koma desimal, tetapi polanya hanya memiliki dua. Metode ini mengangani dengan pembualatan
123.789	000000.00	000123.789	Format dengan penetuan 0 di depan dan di belakang , karena format 0 digunakan sebagai pengganti #
12345.67	$###,###.###	$12,345.67	Format dengan menampilkan tanda $ dan menyesuaikan dengan formatnya

Jika **Anda hanya** ingin menampilkan dua angka **setelah koma desimal, mari** kita **tulis kodenya** seperti ini:

- ✓ double x = 2.0 / 3;
- ✓ System.out.println("x sama dengan = " + (int) (x * 100) / 100.0);
- ✓ System.out.println("Hello World");

Hasilnya adalah

- ✓ x sama dengan 0.66

Untuk melihat hasil pemformatan yang lebih baik sebagai hasilnya, inisialisasi output dapat menggunakan salah satu metode printf. Rumus untuk memanggil metode ini dengan menuliskan :

System.out.printf(format, item1, item2, ... , item-n);

Di mana bentuk adalah meruapakan sebuat text yang berisi subtext dari penentu bentuk tersebut. Penentu format menjelaskan bagaimana objek dihasilkan. Komponennya dapat berupa nilai numerik, nilai karakter, nilai logika, atau string. Kualifikasi sederhana berisi tanda persen (%) diikuti dengan kode konversi.

Contoh penerapan keluaran terformat:

```
int hitung = 5;
double pi = 3.1415;
System.out.printf("hitung adalah = %d dan PI adalah %f", hitung, pi);
```

Nilainya adalah = 15 dan Phi adalah 3.14

Masukan dalam daftar harus sesuai dengan urutan atribut berdasarkan nama dan jenis. menyukai misalnya, definisi dari phi adalah %f. Secara baku, pecahan ditampilkan dalam enam (6) tempat setelah titik desimal. Anda dapat diatur lebar dan resolusi sesuai dengan tabel di bawah ini:

5.5 Tabel Format untuk Angka

Format	Keterangan
%4c	Keluaran karakter dan menambahkan empat spasi sebelum karakter tersebut
%5d	Keluaran item integer dengan lebar minimum 5. Jika jumlah digit dalam item < 5, akan ditambahkan beberapa spasi sebelum angka item. Jika jumlah digit da lam item >5, lebar akan ditambahkan secara otomatis
%10.2f	Keluaran item pecahan dengan lebar minimum 10, termasuk satu titik desimal dan dua (2) digit setalah titik desimal. Jadi dialokasikan 7 digit sebelum titik d esimal. Jika jumlah digit dalam item 7, lebar akan ditambahkan scara otomatis
%6c	Menampilkan karakter dan menambah lima spasi kosong sebelum nilai karakter
%8b	Menampilkan nilai desimal dan menambahkan tiga spasi sebelum nilai false dan dua spasi sebelum nilai true
%5d	Menampilkan nilai integer dengan lebar sedikitnya lima digit. Jika jumlah angka kurang dari lima , manambahkan spasi sebelum angka
%11.2f	Menampilkan nilai floating point (pecahan desimal) dengan lebar sedikitnya sebelas termausk tanda desimal titik dan dua digit stelah tanda desimal. Ada delapan angka di alokasikan sebelum tanda desimal, Jika digit sebelumnya tanda desimal lebih sedikit, menambahkan spasi sebelum digit
%11.2e	Menampilkan nilai floating point (pecahan desimal) dengan lebar sedikitnya sebelah termasuk tanda desimal (titik) , dua digit (angka) setelah tanda desimal dan bagian eksponen, Jika tampilan jumlah dalam notasi ilmiah kurang dari sebelas menambahkan spasi sebelum notasi ilmiah tersebut
%13s	Menampilkan string dengan lebar sedikitnya tiga belas karekter , jika karekter string kurang dari tiga belas menambhkan spasi sebelum string

Pernyataan pada larik yang paling akhir dapat dibentuk atau dimodifikasi, umpama Anda hanya menginginkan dua spasi sebelum titik pecahan dan tiga digit setelah titik pecahan. Berikut adalah konversi untuk batch terakhir:

```
System.out.printf("Nilai integer = %d dan nilai double = %9.3f", nilaiInt, nilaiDouble);
```

Suatu statemen yang diedit di bagian definisi terakhir menghasilkan layar berikut:

Nilai integer-nya adalah = 70 dan nilai double = 243.450

```
public class PraktikumFormat {
   public static void main (String[] args) {
      int angka = 86;
      double nilai = 49.3456;
      //persen n (%n) digunakan untuk pindah baris
      System.out.printf("variabel angka %d%n", angka);
      System.out.printf("variabel angka dengan 2 digit desimal %.2f", nilai);
    }
}
```

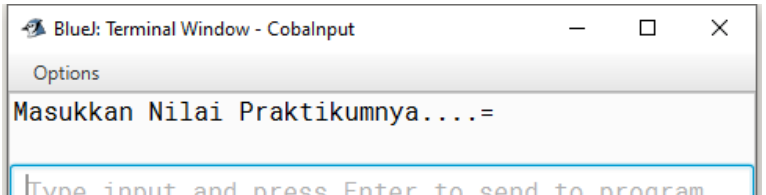
Praktikum Program Input dengan Keyboard

Contoh program dengan menggunakan kelas scanner , dimana kita akan membuat kelas jumlah bilangan, kelas ini akan menjumlahkan dua buah bilangan yaitu nilai pratikum dan nilai prakteknya , kemudian di total nilai tersebut dan di buat rata2nya.

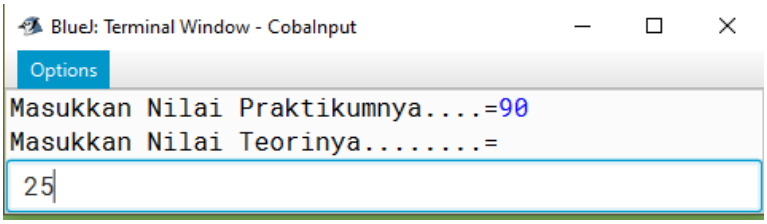
1. Contoh Program Input dengan Keyboard untuk memasukan nilai teori dan praktek

   Buatlah kelas JumlahBilangan dimana program ini kan menghitung nilai rata rata dari nilai masukan teori dan masukan nilai praktek. Selanjutnya silahkan Anda ketikan potongan kode program tersebut, jika sudah jangan lupa untuk di cek dulu aapak ada yang salah dengan klik tombol compile. Jika benar maka hasil program tersbut jika dijalankan akan menghasilkan tampilan seperti berikut. Meminta memasukan nilai pratikumnya, contoh kita isi dengan nilai 90, kemudia sistem minta lagi untuk memasukan nilai teorinya kita isi dengan 25.

```
import java.util.Scanner;
public class JumlahBilangan
{
    public static void main (String[]args){
        int nilait , nilaip, jumlah;
        double rata;
        Scanner baru = new Scanner(System.in);
        System.out.print("Masukkan Nilai Praktikumnya....=");
           nilaip = baru.nextInt();
        System.out.print("Masukkan Nilai Teorinya........=");
           nilait = baru.nextInt();
           jumlah = nilaip + nilait ;
           rata = jumlah/2;
        System.out.println("Jadi Total Nilai adalah.......=" + jumlah);
        System.out.println("Jadi Nilai Rata-rata..........=" + rata);}
}
```

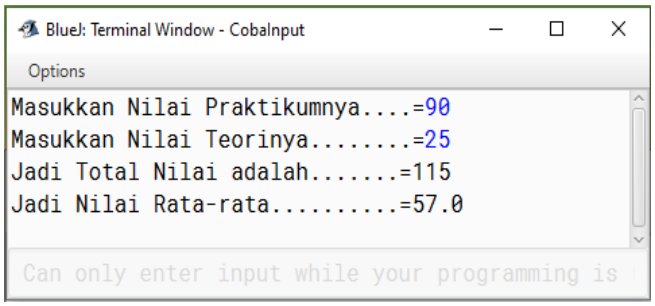


Inputkan nilai Praktikumnya : kinta input dengan angka 90



Inputkan nilai teorinya : kinta input dengan angka 25

Hasil proses untuk nilai terbut adalah seperti tampilan dibawah ini.

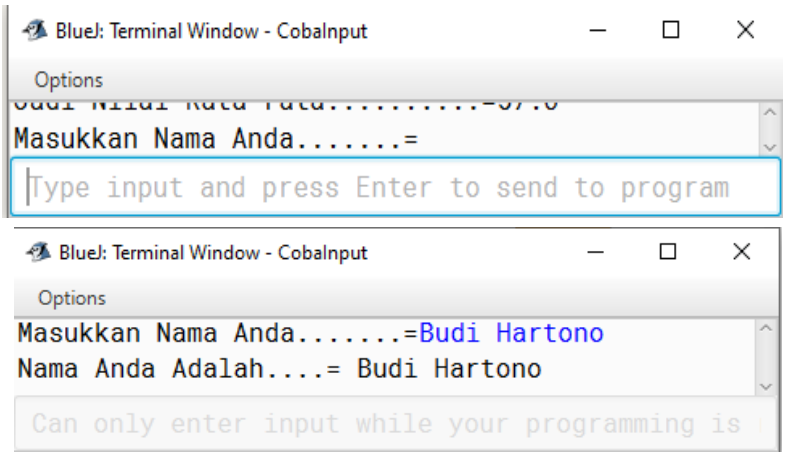


2. Contoh program Input dari keyboard dengan try-out

Contoh program input dari keyboard dengan menggunakan fungsi try-out, pada contoh program ini di minta untuk memasukan nama Anda dan hasil input nama tersbut akan ditampilkan.

Silahkan Anda buat class dengan nama Class Penyangga, selanjutnya Anda dapat menuliskan potongan kode program di bwah ini, jika sudah silahkan di compile dan lihat hasil compilenya apakah masih ada kesalahan jika maish ada betulkan sampai semua benar, jika semu benar maka hasilnya dapat dilihat sperti contoh dibawah ini.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
public class Penyangga
{
    public static void main (String [] args ){
      BufferedReader dataInput = new BufferedReader (new InputStreamReader(System.in));
      String nama =" ";
      System.out.print ("Masukkan Nama Anda.......=");

      try {
          nama = dataInput.readLine();}
          catch(IOException e){
             System.out.println("Terjadi Kesalahan Input...!");}
             System.out.println ("Nama Anda Adalah....= " + nama);}
}
```



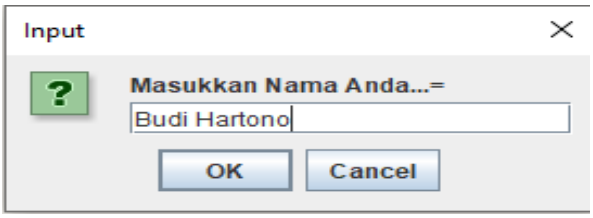
3. Contoh program Input dari keyboard dengan model input dialog box

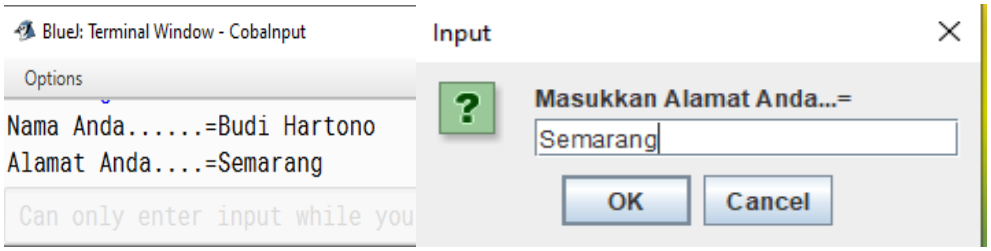
Contoh program input dari keyboard berikut adalah dengan menggunkan JOptionPane, yaitu input dari keyboard dengan menggunakan model input dialog.

Silahkan buat class baru dengan nama Class Pilihan, selanjutnya Anda dapat menuliskan penggalan kode program di bwah ini, jika sudah silahkan di compile dan lihat hasil compilenya apakah masih ada kesalahan jika maish ada betulkan sampai semua benar, jika semu benar maka hasilnya dapat dilihat sperti contoh dibawah ini.

```
import javax.swing.JOptionPane;
public class Pilihan
{
   public static void main (String [] args ){
       String nama, alamat;

       nama = JOptionPane.showInputDialog("Masukkan Nama Anda...=");
       alamat = JOptionPane.showInputDialog("Masukkan Alamat Anda...=");

       System.out.println("Nama Anda......=" + nama );
       System.out.println("Alamat Anda....=" + alamat);
    }
}
```

Hasil dari program tersebut jika di jalankan adalah akan menampilakn hasil seperti dibawah ini , masukan nama anda dengan model input dialog box, jika sdh dinputkan silahakan Anda klik Ok. Selanjutnya akan meminta inputan kembali atau menanyakan masukan alamat Anda. Jika sudah kita input selanjutnya klik oke. Dan hasil komplitnya dapat dilihat pada gambar dibawah.



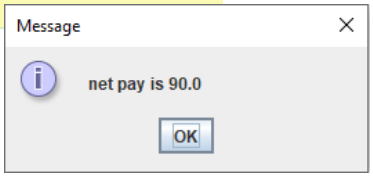


4. Contoh program Input dari keyboard dengan model input dialog box

Contoh program input dari keyboard dengan menggunakan fungsi JoptionPane yaitu input dengan model dialogbox, pada contoh program ini menampilkan messagebox untuk menampilkan suatu pesan.

Buatlah class UserDialog dan ketik kode program seperti tampilan dibawah ini. Compile class tersebut sampai muncul tulisan no syntac error dan jalan program tersebut, maka akan menampilkan pesan net pay is 90.0 dan selajutnya kita klik Ok.

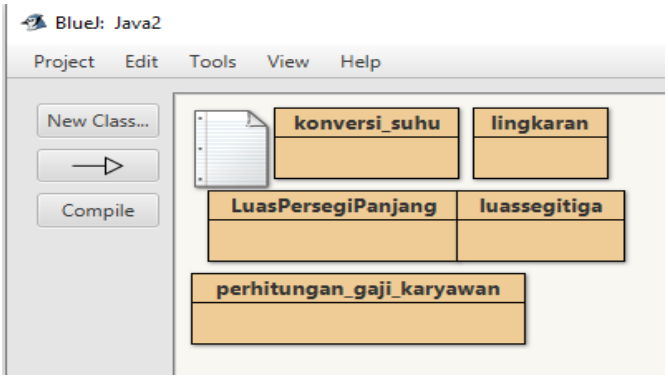
```
import javax.swing.JOptionPane;

public class UserDialog
{
  public static void main ( String [] args )
  {
    double netPay , grossPay , deductions ;
    grossPay = 100.00;
    deductions = 10.00;
    // Calculate net pay
    netPay = grossPay - deductions ;
    JOptionPane . showMessageDialog (null , " net pay is " + netPay ) ;
}
}
```



Selanjutnya Anda dapat mengerjakan latihan latihan seperti yang di minta dibwah ini.

Latihan -1

Buatlah input dari keyboard untuk menghitung luas persegi panjang, luas segitiga, luas lingkaran, konversi suhu dari celcius ke farehait dan reamur dan menghitung gaji karyawan.



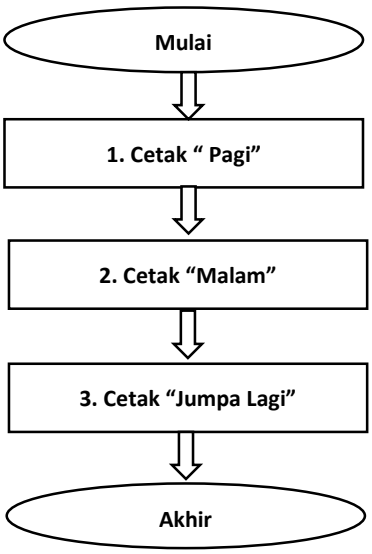
Latihan -2

1. Untuk latihan dirumah silahkan Anda buatlah sebuah program untuk mencari luas dan keliling empat persegi panjang. Dimana pada lebar dan panjang di inputkan dengan menggunakan inputan bukan dengan variabel input, selanjutnya tampilkan hasil dari proses untuk luas dan keliling empat persegi panjang :
   - ✓ Inputan untuk empat persegi panjang dari Lebar adalah = 8
   - ✓ Inputan untuk empat persegi panjang dari panjang adalah = 5
   - ✓ Jadil Luas empat adalah persegi panjang = 40
   - ✓ Jadi Keliling empat persegi panjang adalah = 36

2. Buat program Java menghitung luas segi tiga dan menampilkan hasilnya!, diman pada sisi tinggi dan alas Anda inputkan dari keyboard, dan Hasilnya kurang lebih seperti berikut:
   - ✓ Masukkan tinggi segitiga = 8
   - ✓ Masukkan alas segitiga = 5
   - ✓ Luas segitiga = 20
3. Buat sebuah program Java yang menerima masukan dari keyboard untuk suhu Celcius dan selanjutnya Anda konversi kedalam suhu Reamur dan Fahrenheit dan tampilkan hasilnya!
   - ✓ Fahrenheit = (9/5) * Celcius + 32
   - ✓ Reamur = 4/5 * Celcius
   - ✓ Hasil yang diharapkan:
   - ✓ Suhu pada derajat Celcius = 10
   - ✓ Suhu pada derajat Fahrenheit = 50
4. Ulangi pembuatan program diatas dengan menggunakan kelas BufferedReader dan JOptionPane. Apa kesimpulannya?

BAB VI
KONTROL PROGRAM (PERCABANGAN)

Instruksi atau petunjuk dalam source code program umum dijalankan secara squensial dari pertama sampai pada tahan yang terakhir, yaitu dengan alur pelaksanaan kode program dilakukan secara berurutan pertama samai pada baris terakhir. Mesin komputer membaca baris demi baris, lalu komputer melakukan apa yang diperintahkan.



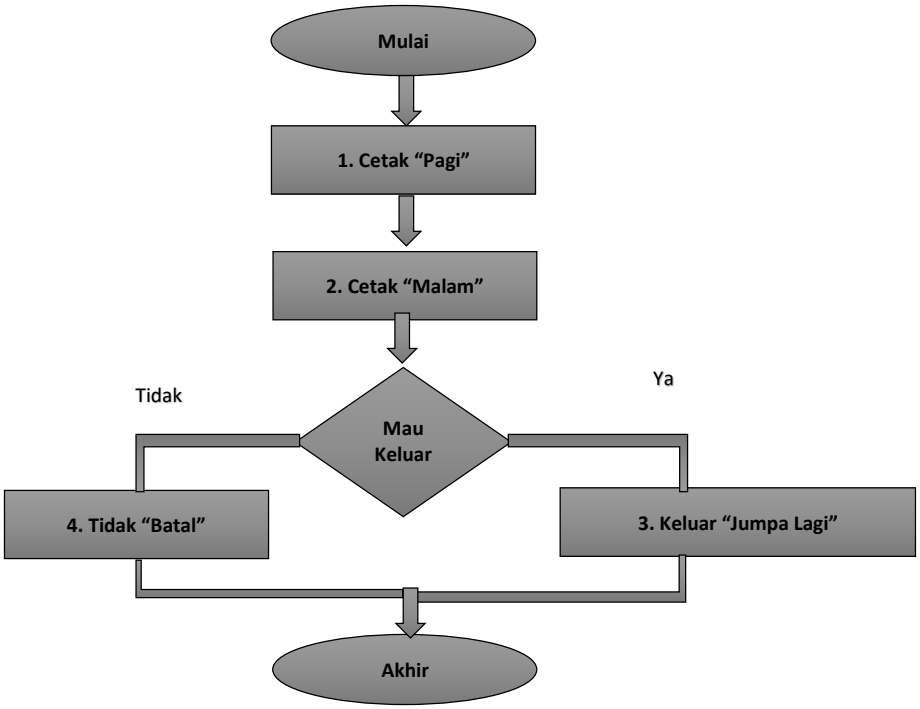
Namun ada kalanya suatu instruksi akan dieksekusi dalam situasi tertentu. Persyarata situasi tertentu yang harus dipenuhi suatu program untuk mengeksekusi suatu perintah, Andai persyaratan dan ketentuan tersebut terlaksana, maka program akan melakukan eksekusi instruksi tersebut, tetapi jika tidak, program tidak akan mengeksekusi atau melewatkan instruksi tersebut dan melihat situasi eksekusi lainnya atau berhenti sama sekali.

Definisi Percabangan

Percabangan adalah tugas (instruksi) yang memungkinkan tugas (instruksi) dieksekusi andai suatu kondisi dinyatakan terpenuhi atau dinyatakan tidak terpenuhi. Jika kondisi terpenuhi, perintah dijalankan. Jika kondisi tidak terpenuhi, perintah lain dijalankan.

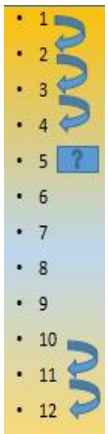
Percabangan dalam pemrograman dipakai oleh komputer untuk menentukan tahapan tahapan kerja instruksi. Cabang menggunakan operator bersyarat yang mengembalikan nilai Boolean (benar/benar atau salah/salah). Jika nilai hasil benar, perintah (instruksi) dijalankan, sedangkan jika salah, instruksi tidak dijalankan atau instruksi lainnya dijalankan. Daerah ini perlu mendapatkan perhatian jika Anda baru belajar pemrograman, tetapi sebenarnya adalah wajar saja dan merupakan dasar dasar pemrograman. Dalam pemrograman Java terdapat 4 jenis percabangan yang dapat kita kelompokkan berdasarkan bentuk atanra laian if, if-else, if else if else dan yang terakhir adalah switch-case.

Tentu saja dari ke empat(4) percabangan ini masing-masing memiliki aturan dan cara penggunaannya tersendiri. Misalnya saat membuat program untuk pengujian: masuk, ketika Anda masuk, Anda akan dimintai kata sandi pengguna. Apakah pengguna memasukkan kata sandi yang benar? Jika demikian, izinkan pengguna untuk terus mengakses sistem, jika tidak, maka pengguna tidak mendapatkan hak aksesnya. Kesimpulannya adalah bahwa bahasa pemrograman membutuhkan cara untuk mengambil suatu kondisi program ke salah satu dari dua arah. Dalam hal ini disebut kondisi pernyataan IF.



Konsep Percabangan

- ✓ Tidak semua instruksi program akan di jalanakan atau di terjemahkan
- ✓ Suatu pernyataan akan dieksekusi jika memenuhi ketentuan yang berlaku atau kondisi tertentu dan pada dasarnya kondisi umumnya adalah Boolean
- ✓ Kondisi Boolean merupakan suatu ekspresi hubungan yang bernilai betul atau salah tergantung pada nilai dari jenis operasi yang terlibat didalamnya
- ✓ Pada kondisi boolean dan aksi yang akan dijalankan semua terpatokan pada jumlah percabangan atau permasalahan yang sedang dihadapi tersebut apakah terdapat satu pilihan, dua pilihan, ataukah terdiri atas banyak pilihan



Tidak semua baris dalam urutan ini akan di eksekusi, hanya akan di eksekusi bila memenuhi syarat

Contoh pada baris 6,7,8 dan 9 akan di eksekusi dengan melihat kondisinya benar atau salah.

Baris 1,2,3,4 di Eksekusi baris ke – 5 menanyakan kondisi , jika kondisni yang di ujia bernilai benar maka eksekusi baris 6,7 dan melewatkan pada baris ke 8 dan 9 dan akan mengeksusi kembali pada baris 10,11, 12 dst.

Baris 1,2,3,4 di Eksekusi baris ke – 5 menanyakan kondisi , jika kondisi yang di uji nilainya salah maka akan melewatkan baris 6, 7 dan akan mengeksekusi pada baris 8,9,10,11, 12 dst

IF Tunggal

Seperti namanya, jika tunggal hanya memiliki satu blok keputusan berdasarkan hanya satu kondisi. Sederhananya, pilihan ini berarti keputusan untuk mengeksekusi blok pernyataan atau hanya pernyataan, dan hanya jika kondisinya benar, maka akan dieksekusi, dan jika kondisi tidak terpenuhi,

maka suatu kelompok pernyataan akan dilewati. . Rumus pernyataan IF yang adigunaka adalah dengan melihat struktur dibawah ini :

```
if (kondisi)
    statement;
atau
if (kondisi){
    statement 1;
    statement 2;
    ...
    statement – n;
}
```

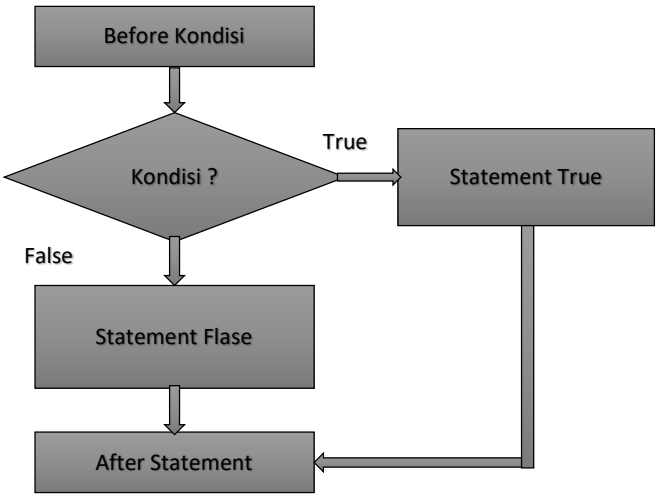
Jika hasil penilaian suatu kondisi benar, maka pernyataan pada blok akan dieksekusi. Kurung kurawal dapat dihilangkan. jika kondisi pasti satu pernyataan yang akan dieksekusi jika kondisinya benar.
Dalam prakteknya, dicission tree selalu mengandung operator relasional dan operator logika. Hingga dua operator sering dipakai secara bersama sama dalam suatu situasi kondisi untuk menentukan keabsahan suatu batas nilai, khususnya untuk nilai data bertipe bilangan (bilangan bulat/pecahan) atau bertipe teks (string atau karakter). Untuk kejelasan, penggunaan pemilih if tunggal ini ditunjukkan pada contoh di bawah ini
Contoh penulisan statemen if:

```
if (nilai > 60){
    ket = "Lulus";
}
System.out.println("Nilai   = " + ket);
```

Pada contoh fragmen kode sumber di atas, dapat divisalisasikan bahwa jika variabel nilai lebih besar dari > 60 maka dinyatakan lulus, tetapi jika nilainya < 60 maka tidak mengerjakan apapun.

IF – ELSE

Pohon keputusan ini pada hakekatnya adalah sebuah pilihan ketika sebuah blok program (pernyataan) terdiri dari dua pilihan, dengan hanya satu pilihan diantara dua pilihan yang dieksekusi (dieksekusi). Pada blok klausa where kondisi paling awal akan dieksekusi jika nilai kondisi pertama terpenuhi maka akan di eksekusi, tetapi jika keadaan awal tidak terpenuhi, maka baris pernyataan diabaikan, dan akan mengeksui pada blok pernyataan yang laian.



Berikut adalah sintaks suatu percabangan if ... else :

```
if (kondisi) {
    statement – statement true
} else {
    statement – statement false
}
```

Ketika sistem mengeksekusi pernyataan if, pertama-tama sistem akan mengevaluasi kondisi logis. Jika kondisi benar, ekspresi-1 (ekspresi) dijalankan; jika kondisi salah, pernyataan 2 dijalankan. Jika pernyataan secara kondisional mengeksekusi salah satu pernyataan-1 atau pernyataan-2, tidak pernah keduanya.

Sebagai contoh, perhatikan kode berikut:

```
if (nilai >= 65) {
    System.out.println("Selamat Anda Lulus");
} else {
    System.out.println("Mohon Maaf Anda Tidak Lulus");
}
```

Penjelasan dari contoh di atas:

Jika nilai pada variabel lebih besar atau sama dengan enampuluh lima (>= 65) maka pengujian kondisinya adalah benar, maka akan tampil kalimat “Selamat atas kelulusan” ditampilkan, sebaliknya jika kondisi salah maka akan menampilkan pesan “Maaf Anda tidak lulus).

IF Majemuk (if ... else if … else)

Pola if ini adalah evolusi dari pilihan dengan dua pernyataan blok atau pernyataan, bentuk pilihan if banyak ini sering disebut dengan if berjenjang seperti tangga. Secara struktural, pola ini membentuk rangkaian tangga yang berkesinambungan satu sama laian. Cara kerja dari model pola ini adalah dengan menguji nilai awal keadaan atau kondisi pertama, jika kondisi ini terpenuhi maka baris pernyataan pertama akan dijalankan, tetapi jika kondisi awal tidak terpenuhi maka blok pernyataan akan diabaikan dan dilanjutkan pada program untuk menguji kondisi selanjutnya dalam hal ini adalah yang kedua, jika nilai kondisi kedua terpenuhi, maka pernyataan di dalamnya baris perintah akan dijalankan, tetapi jika kondisi kedua juga tidak terpenuhi, baris pernyataan pertama dan kedua dilewati

dan yang selanjutnya dalah hali ini adalah yang ke-3, ke-4, dst. .diuji sampai kemudian tiba di tempat lain. Struktur lain adalah pilihan jika tidak semua kondisi terpenuhi. Kata "lain" berarti "selain" atau "berbeda" dari "pertama", "kedua", "ketiga", dll. sebelum yang lainnya. Artinya, jika tidak semua kondisi terpenuhi, blok pernyataan else dijalankan.

Struktur umum dari rumus if majemuk ini adalah seperti bentuk dibawah ini:

```
if (kondisi-1) {
    aksi jika kondisi-1 benar
} else if (kondisi-2){
   aksi jika kondisi-2 benar
} else if (kondisi-3){
   aksi jika kondisi-3 benar
} ...
else if (kondisi-N){
    aksi jika kondisi-N benar
} else {
   aksi jika seluruh konisi tidak terpenuhi
}
```

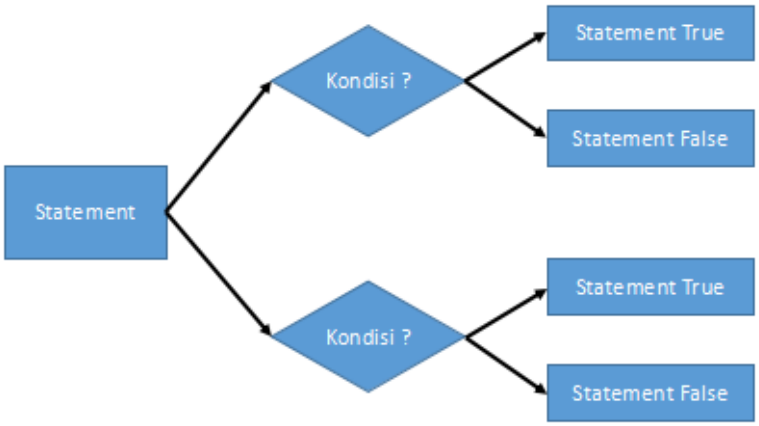
Perhatikan contoh berikut:

```
int testscore = 76;
char grade;
if (testscore >= 86) {
grade = 'A';
} else if (testscore >= 75 {
grade = 'B';
} else if (testscore >= 65) {
grade = 'C';
} else if (testscore >= 55) {
grade = 'D';
} else {
grade = 'F';
}
System.out.println("Grade = " + grade);
```

Jalankan pernyataan if di atas menghasilkan keluaran Grade = b.
Proses eksekusi pertama mengecek apakah nilainya (hasil tes >= 86), jika benar maka gradenya adalah “A”, jika kondisi salah maka kondisi tes kedua diperiksa (hasil tes >= 75). Jika nilai kondisinya benar, nilainya adalah "B". Jika kondisinya salah, kondisi selanjutnya diperiksa hingga kondisi tersebut terpenuhi (benar). Jika semua kondisi ternyata salah, nilainya adalah "F".

NESTED IF

Terkadang struktur if dibangun. Jika struktur if-select berisi if-select. Struktur ini disebut nested if atau nested if structure. Dalam bentuk lain , pemilihan if bagian dalam harus bersarang di dalam pemilihan if bagian luar. (pilihan demi pilihan).



Kondisi percabangan dalam percabangan atau nested if ditulis dengan bentuk seperti berikut ini :

```
if( ekspresi_boolean1) {
    statemen_if_utama;
if(ekspresi_boolean2) {
    statement_if_within_if;
}
}
```

Perhatikan contoh Nested If Berikut ini :

```
int a = 5, b = 3, c = 2;
int maks;
if (a > b) {
if(a > c) {
maks = a;
} else {
maks = c;
}
} else if (b > c){
maks = b;
} else {
maks = c; }
```

Dari contoh diatas dimana kondisi pertama akan menguji nilai apakah nilai a > b , jika uji kondisinya terpenuhi maka baris statemen didalam if tersebut akan dieksekusi dan tentunya juga menguji perbandingan antara nilai a dan c. Jika nilai a > c, maka baris statemen pada if kondisi (anak) pertama yang akan dieksekusi, tetapi jika kondisi tersebut tidak terpenuhi, maka baris statemen pada bagian else yang akan dieksekusi

Latihan

1. Buatlah suatu program dengan memperhatikan data tabel dibawah ini dengan nilai input berupa usia dan program dapat memperlihatkan dari sebuah kelompok atau kategori dari usia tersebut berdasarkan keterangan yanga ada.

Usia	Kategori
1-16 thn	Anak-anak
17-25 thn	Remaja
>25	Dewasa

2. Buatlah program untuk menentukan potongan dan total bayar dari sejumlah pembelian dengan aturan diskon sebagai berikut :

NO.	Jumlah Beli	Potongan
1	>300,000.00	40%
2	>200,000.00	30%
3	>100,000.00	20%
4	>50,000.00	10%
5	<50,000.00	5%

3. Buatlah sebuah program untuk menentukan bearan gaji yang didapatkan dari seorang karyawan berdasarkan ketentuan sebagai berikut:
   - ✓ Gaji pokok ditentukan berdasarkan dari golongan karyawan
   - ✓ Uang transport dihitung per hari dari kehadirannya.
   - ✓ Pendapatan per bulan adalah = gaji pokok + (jumlah masuk * uang transport)

Gol	Gaji Popok	Uang Transport
A	4.000.000,00	25.000,00
B	3.000.000,00	15.000,00
C	2.000.000,00	10.000,00
D	1.000.000,00	5.000,00

4. Buat program untuk menentukan harga rental berdasarkan status peminjam dari sebuah rental buku komik
   - ✓ Yang retalnya status pelajar, maka harga rentalnya buku adalah Rp. 500,-/hari.
   - ✓ Yang rentalnya statusnya mahasiswa, maka harga rentalnya buku adalah Rp. 1000,-/hari.
   - ✓ Yang rentalnya statusnya masyarakat umum, maka harga rental buku adalah Rp. 1500,-/hari.
5. Rental tersebut, memberikan diskon kepada peminjam berdasarkan banyak buku yang dipinjam.
   - ✓ Banyak buku yang dipinjam lebih dari 5, maka akan mendapatkan potongan sebanyak 5% dari total harga rental buku.
   - ✓ Banyak buku yang dipinjam kurang dari 5, maka akan mendapatkan potongan 2,5% dari total harga rental.

6. Rental tersebut memiliki peraturan tentang batasan banyak buku yang boleh dipinjam dan durasi peminjaman. Banyak buku yang boleh dipinjam maksimal 10 buku, dan lama peminjaman maksimal adalah 7 hari.
7. Silahkan Anda buat programnya dengan menggunakan kelas scanner untuk menghitung berapa harga rental yang harus dibayarkan oleh peminjam!

Switch – Case

Penyataaan atau penyelesaian if – yang banyak tentu akan menimbulkan masalah tersendiri bagi programernya selain instruksi instruksinya jadi panjang juga akan sulit untuk dibaca. Untuk penyelesaian if banyak ini Java sudah menyediakan dengan pernyataaan yaitu switch case. Switch Case digunakan untuk penganan dari berbagai alternatif secara lebih singkat. Beda dengan if, switch hanya diperbolehkan untuk pilihan kondisi yang bertipe bilangan bulat atau char. Char sebenarnya adalah sebuah angak yang memliki rental nilai anatara 0 sampai dengan 65535.
Rumus dari percabangan switch – case dapat Anda lihat dibawah ini:

```
switch (switch_expression){
case case_selector_1:
statement 1; //
statement 2; // Blok 1
... //
break;
case case_selector_2:
statement 1; //
statement 2; // Blok 2
... //
break;
default:
statement 1; //
statement 2; // Blok n
... //
break;
}
```

- ✓ Pokok kerja dari bentuk struktur pernyataan switch case adalah , saat pernyataan switch ditemukan pada penggalan kode program, maka pertama kali akan memeriksa switch_expression dan menuju ke case yang akan menyamakan nilai yang dimiliki oleh switch_expression.
- ✓ Kemudian program akan mengeksekusi satatement pada kode tersebut setelah case yang dijumpai sampai menemukan pernyataan break. Apabila tidak ditemukan case yang cocok, maka program akan mengeksekusi standart yang telah di tetapkan.
- ✓ Bagian standart atau baku ini adalah sebuat pilihan (opsional) artinya boleh dipakai tidakpun tidak masalah. Sebuah pernyataan switch bisa jadi tidak memiliki kode standart.
- ✓ Ada dua cara dalam 2 penggunaan keyword break,
  - o Saat keluar dari kendali percabangan switch,
  - o Saat keluar dari kendali perulangan.

- ✓ Dengan menggunkan kata kunci ini maka percabangan/perulangan akan diakhiri, selanjutnya eksekusi dilanjutkan ke pernyataan setelah blok percabangan/perulangan tersebut.

Catatan:

- ✓ Tidak sama dengan pernyataan pada if, beberapa pernyataan switch –case pada struktur pernyataan switch akan dijalankan tanpa memerlukan tanda kurung kurawal ({}).
- ✓ Saat pernyataan case pada switch menjumpai kesamaan, selanjutnya semua pernyataan pada case tersebut akan dijalankan. Pernyataan lain yang berada pada case yang sesuai juga akan dijalankan.
- ✓ Untuk menghindari program mengeksekusi pernyataan pada case selanjutnya , kita menggunakan pernyataan break sebagai pernyataan akhir dari setiap blok case.

Contoh Penggunaan Switch

```
int nHari = 5;
    switch (nHari){
    case 1:
    System.out.println("Hari Senin");
    break;
    case 2:
    System.out.println("Hari Selasa");
    break;
    case 3:
    System.out.println("Hari Rabu");
    break;
    case 4:
    System.out.println("Hari Kamis");
    break;
    case 5:
    System.out.println("Hari Jumat");
    break;
    case 6:
    System.out.println("Hari Sabtu");
    break;
    case 7:
    System.out.println("Hari Minggu");
    break;
    default:
    System.out.println("No Hari Tidak ada yang sesuai");
    }
```

Potongan source code diatas adalah memeriksa apakah nHari cocok dengan nilai 1,2,3,4,5,6, dan 7. Jika cocok, akan ditampilkan nama hari seusai dengan case nya. Jika tidak ada satu pun yang cocok, maka nilai default akan dikerjakan. (No Hari Tidak ada yang sesuai)

Latihan-2

1. Buatlah program yang terdapat struktur switch – case didalamnya untuk menentukan gaji bulanan seorang pekerja berdasarkan golongannya. (Lihat tabel)

Gol	Gaji
1	1.500.000,00
2	2.000.000,00
3	3.000.000,00
4	3.500.000,00

Praktikum Kontrol Program Percabangan

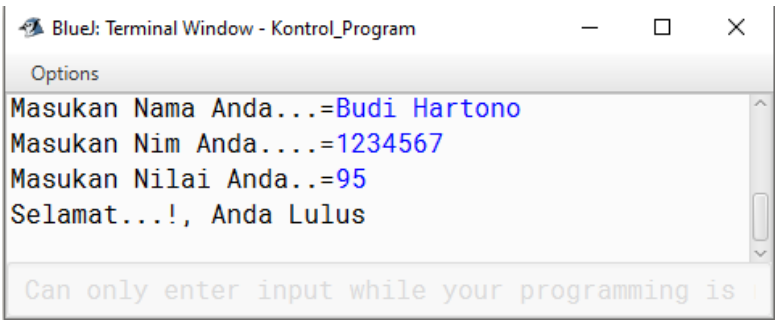
1. Untuk IF-ELSE

Untuk lebih jelas pemahaman mengenai percabangan berikut ditampilkan contoh contoh program percabangan

Buatlah class dengan nama percabangan, selanjutnya Anda bisa membaerika kode program sperti contoh dibwah ini, Program percabangan ini akan memenita inputan nama Anda, nim dan nilai Anda, selanjutnya inputan nilai inilah yang akan di ujia apakah anda lulus tau tidak, jika nilai >= 60 maka anda dinyatakan lulus, jika nilai anda <60 maka Anda dinyatakan tidak lulus alias harus melakukan remisi.

```
import java.util.Scanner;
public class Percabangan
{
    public static void main(String[] args) {
        Scanner kontrol = new Scanner(System.in);
        System.out.print("Masukan Nama Anda...=");
        String nama = kontrol.nextLine();
        System.out.print("Masukan Nim Anda....=");
        String nim = kontrol.nextLine();
        System.out.print("Masukan Nilai Anda..=");
        int nilai = kontrol.nextInt();

        if (nilai > 60) {
            System.out.println("Selamat...!, Anda Lulus");}
        else {
            System.out.println("Maaf....Anda Harus Remidi");}
        }
}
```

Program diatas akan menguji sebuah nilai, jika nilai pada variabel tersebut lebih dari atau sama dengan (>= 60) , (kondisi benar), maka akan menampilkan kalimat “Selamat Anda Lulus”, sebaliknya (keadaan salah) akan menampilkan kalimat “Anda Harus Remidi” (else). Lihat hasilnya seperti gambar dibawah ini.



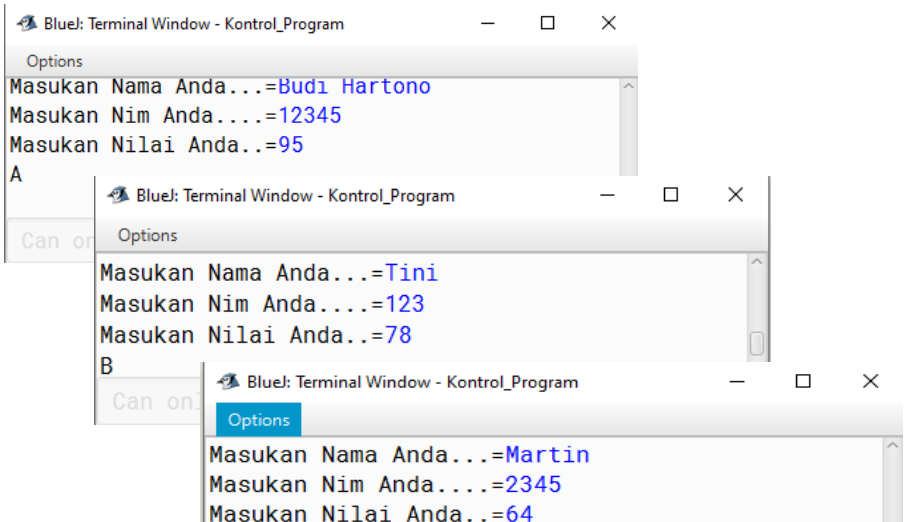
2. Untuk IF-Majemuk

If bertingkat atau if majemuk bisa dikatakan percabangan dengan banyak arah atau if bertingakat yang terdiri atas beberapa pernyataan if, bila kondisi tidak dipenuhi, maka akan menyakan kembali kondisi yang lain dan seterusnya. Jika semua kondisi dalam keadaan false alias tidak terpenuhi, maka akan mengerjakan statement di bawah else (paling terakhir).

Berikut adalah contoh penerapan pada program If Majemuk.
Contoh kasus dibawah adalah untuk penyelesaian if mejemuk, untuk penyelesaian nilai dengan batas batas tertentu, nilai >= 86 maka gradenya adalah "A", nilai >= 81 maka gradenya adalah "AB", nilai >= 73 maka grade adalah "B", nilai >= 66 maka gradenya adalah "BC", nilai >= 55 maka gradenya adalah "C" sedangkan nilai kurang dari 55 maka gradenya adalah "E".
Buatlah class dengan nama Majemuk jangan lupa untuk import java until scanner , ketikan kode programnya seperti tampilan dibawah. Jika sudah selesai cek dengan compile apakah amasih ada yang salah, jika maasih betulkan smapai benar dan selanjutnya jalan program tersebut.

```
import java.util.Scanner;
public class Majemuk
{
    public static void main(String[] args) {
        Scanner kontrol = new Scanner(System.in);
        System.out.print("Masukan Nama Anda...=");
        String nama = kontrol.nextLine();
        System.out.print("Masukan Nim Anda....=");
        String nim = kontrol.nextLine();
        System.out.print("Masukan Nilai Anda..=");
        int nilai = kontrol.nextInt();

        if (nilai >= 86) {
            System.out.println("A");}
        else if (nilai >=81){
            System.out.println("AB");}
        else if (nilai >=73){
            System.out.println("B");}
        else if (nilai >= 66){
            System.out.println("BC");}
        else if (nilai >= 55){
            System.out.println("C");}
        else
            System.out.println("E");}
}
```

Hasil program setelah dijalankan akan menampilkan seperti tampilan dibwah ini. Masukan Nama Anda, Masukan Nim Anda dan Masukan Nilai Anda. Jika nilai sudah di inputkan maka grade akan keluar secara otomatis sesuai dengan gradenya. Input nilai 95 maka otomatis gradenya adalah A, masukan nilai 78 maka gradenya adalah B, begitu dan seterusnya.



3. NESTED IF (Percabangan Tersarang)

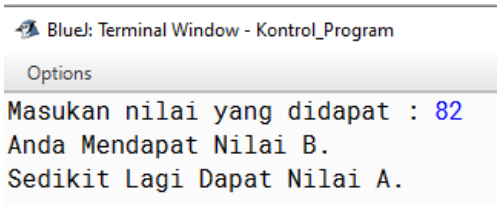
Contoh kasus dibawah adalah untuk penyelesaian if tersarang atau if di dalam if, untuk penyelesaian nilai dengan batas batas tertentu :

- ✓ Jika nilai >= 86 maka gradenya adalah “A”, dan jika nilai dapat 100 maka ada tambahan yaitu “ Luar biasa nilai sempurna”
- ✓ Jika nilai >= 70 maka gradenya adalah “B”, dan jika nilai >= 80 maka ada tambahan yaitu “Sedikit lagi Nilai Agrade adalah “A”,
- ✓ Jika nilai >= 55 maka gradenya adalah “C”, dan jika nilai >= 65 maka ada tambahan grade “ Kurang sedikit lagi nilai B”
- ✓ Jika nilai >= 40 maka gradenya adalah “D” sedangkan nilai kurang dari 40 maka gradenya adalah “E”.

Buatlah class dengan nama IF_if , import java until scanner , ketikan kode programnya seperti tampilan dibawah. Jika sudah selesai klik tombol compile apakah amasih ada yang salah, jika maasih betulkan smapai benar dan selanjutnya jalan program tersebut.

```
import java.util.Scanner;
public class If_if
{
    public static void main(String[] args) {
       System.out.print("\u000c");
        float nilai;
        Scanner masukan= new Scanner(System.in);
        System.out.print("Masukan nilai yang didapat : ");
        nilai = masukan.nextFloat();
        if(nilai > 86){
            System.out.println("Anda Mendapat Nilai A. ");
            if(nilai==100){
                System.out.println("Luar Biasa Nilai Sempurna 100.");
            }
        }
        else if(nilai > 70){
            System.out.println("Anda Mendapat Nilai B. ");
            if(nilai>=80){
                System.out.println("Sedikit Lagi Dapat Nilai A.");
            }
        }
        else if(nilai > 55){
            System.out.println("Anda Mendapat Nilai C. ");
            if(nilai>=65){
                System.out.println("Kurang Sedikit Lagi Dapat Nilai B.");
            }
        }
        else if(nilai > 40){
            System.out.println("Anda Mendapat Nilai D. ");
        }
        else {
            System.out.println("Anda Mendapat Nilai E. ");
        }
    }
}
```

Hasil program ketika di ajalankan adalah akan tampil seperti gambar dibawah



4. Switch – Case

Latihan switch case untuk menentukan hasil dari input

- ✓ jika nilai yang di inputkan adalah “A” maka hasilnya adalah “Pertahankan”,
- ✓ Jika input nilai sama dengan “B” maka haislnya adalah “Harus lebih Baik Lagi”,
- ✓ jika nilai yang diinputkan adalah nilai sama dengan “C” maka hasilnya adalah “Perbanyak Belajar”,
- ✓ jika yang diinputkan adalah nilai “D” maka hasilnya adalah “Jangan Keseringan Main” dan
- ✓ jika yang dinputkan adalah nilai “E” maka hasilnya adalah “Kebanyakan Bolos”.
- ✓ Bagiamana jika yang dinputkan tersebut tidak ada pada data yang disebut diatas, maka bisa dikaishkan format atau hasilnya adalah “Maaf , format nilai tidak Sesuai”

```
import java.util.Scanner;

class BelajarJava {
  public static void main(String args[]){

    char nilai;
    Scanner input = new Scanner(System.in);

    System.out.print("Input Nilai Anda (A - E): ");
    nilai = input.next().charAt(0);

    switch (nilai) {
      case 'A':
        System.out.println("Pertahankan!");
        break;
      case 'B':
        System.out.println("Harus lebih baik lagi");
        break;
      case 'C':
        System.out.println("Perbanyak belajar");
        break;
      case 'D':
        System.out.println("Jangan keseringan main");
        break;
      case 'E':
        System.out.println("Kebanyakan bolos...");
        break;
      default:
        System.out.println("Maaf, format nilai tidak sesuai");
    }

  }
}
```

```
Input Nilai Anda (A - E): A
Pertahankan!

Input Nilai Anda (A - E): D
Jangan keseringan main

Input Nilai Anda (A - E): E
Kebanyakan bolos...

Input Nilai Anda (A - E): F
Maaf, format nilai tidak sesuai
```

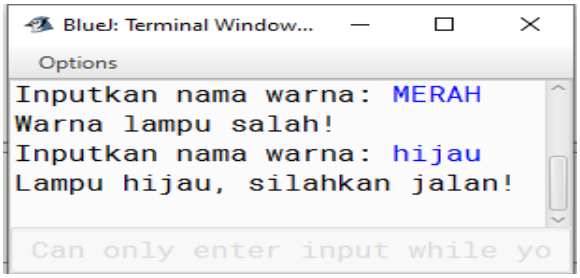
Latihan switch-case yang lain adalah sebuah inputan dalam bentuk type string dimana pada inputan string ini juga menampilkan hasil dalam bentuk string juga. Program ini meminta inputan string yang berupa lampu lalulintas yang itu “Merah”, “Kuning” dan “Hijau”.

- ✓ Jika inputan lampu warna “Merah” maka keterangannya adalah berhenti,
- ✓ Jika inputan warna “Kuning” maka keterangan adalah “Lampu Kuning Harap hati-hati”,
- ✓ jika inputan lampu warna “Hijau” maka keterangan adalah “ Lampu Hijau , Silahkan Jalan”.
- ✓ Bagaimana jika inputan warna selain warna merah, kuning dan hijau atau inputan kedua lampu menyala, apakah hasilnya?.
- ✓ Jika yang diinputkan selain ketiga warna tersebut maka akan muncul keterangan “Warna Lampu Salah”.

Untuk penerapan programnya dapat dilihat pada source code dibawah ini.

```
import java.util.Scanner;
public class Lampubangjo
{
    public static void main(String[] args) {
        // membuat variabel dan Scanner
        String lampu;
        Scanner scan = new Scanner(System.in);
        // mengambil input
        System.out.print("Inputkan nama warna: ");
        lampu = scan.nextLine();
        switch(lampu){
            case "merah":
                System.out.println("Lampu merah, berhenti!");
                break;
            case "kuning":
                System.out.println("Lampu kuning, harap hati-hati!");
                break;
            case "hijau":
                System.out.println("Lampu hijau, silahkan jalan!");
                break;
            default:
                System.out.println("Warna lampu salah!");
        }
    }
}
```

Hasil program percabangan dengan menggunakan swicth - Case



Latihan Kontrol Program Percabangan

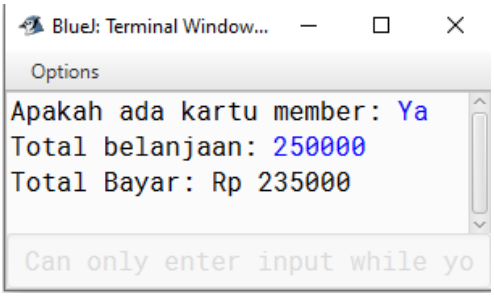
1. Buatlah skrip program yang menerima input nilai berupa data integer dan menghasilkan output berupa nilai huruf dan keterangan dengan ketentuan sebagai berikut.

Rentang nilai input	Output	Ket
✓ 0 sampai 50	E	Remidi
✓ 51 sampai 60	D	Cukup
✓ 61 sampai 75	C	Baik
✓ 76 sampai 85	B	Baik Sekalai
✓ 86 sampai 100	A	Unggul
✓ Input lebih Kecil dari 0 atau lebih besar dari 100 Error		

2. Contoh sebuah supermarket menerapakan kebijkan untuk model bisnisnya dengan membrikan diskon sesuai dengan ketentuan total bayarnya. Ketika orang membayar di kasir, biasanya ditanya ada kartu member untuk mendapatkan diskon dan sebagainya

Apakah anda punya kartu member?
- Ya
  - Apakah belanjaan anda diatas 500.000?
    - ✓ ya : mendapatkan diskon 50.000
    - ✓ tidak : tidak mendapatkan potongan=0
  - Apakah belanjaan anda diatas 100.000?
    - ✓ ya : mendapatkan diskon 15.000
    - ✓ tidak: tidak mendapatkan potongam =0
- Tidak
  - Apakah belanjaan anda lebih dari 100.000?
    - ✓ ya : mendapatkan diskon 10.000
    - ✓ tidak: tidak mendapatkan diskon

Hasilnya kurang lebih seperti ini



3. Contoh sebuah supermarket menerapakan kebijkan untuk model bisnisnya dengan membrikan diskon sesuai dengan ketentuan total bayarnya. Ketika orang membayar di

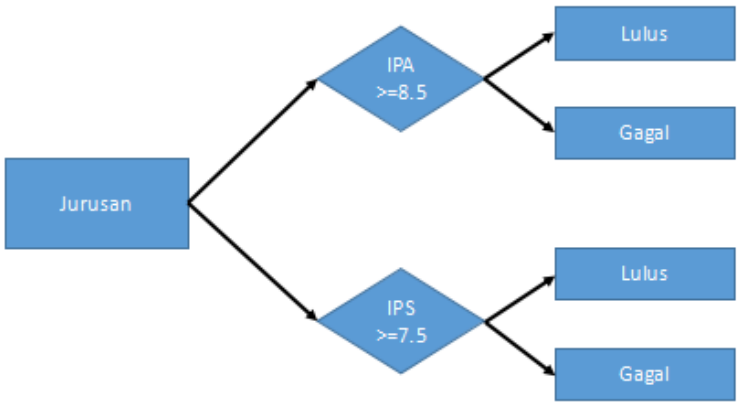
Pilihan	Nama Pemesan	Jenis Makanan	Harga	Jumlah Pesan	Total Bayar
1	Input	Mie Ayam	12.000	Input	Proses
2	Input	Bakso Urat	25.000	Input	Proses
3	Input	Nasi Goreng	20.000	Input	Proses

4	Input	Bakmi Goreng	15.000	Input	Proses
5	Input	Bakmi Kuah	17.500	Input	Proses

- ✓ Untuk Jenis makanan dan Harga hasil dari input nilai yang sesuai dengan daftar
- ✓ Untuk Nama Pemesan dan Jumlah Pesan Silahakan Di Inputkan
- ✓ Total Bayar Proses dari Harga dikalikan (*) Jumlah Pesan

4. Buatlah program dengan algoritma sebagai berikut :
   - ✓ Input melalui keyboard : NPM, Nama, Nilai Tugas-1, Nilai Tugas-2, Nilai UTS dan Nilai UAS
   - ✓ Cari rata-rata nilai Tugas
   - ✓ Hitung nilai akhir dengan ketentuan : 40% UTS + 30% UAS + 30% Tugas
   - ✓ Hitung nilai mutu dengan ketentuan :
     - hasil akhir > 86 nilai mutu = A
     - hasil akhir 80 - <85, nilai mutu = AB
     - hasil akhir 73 - <79, nilai mutu = B
     - hasil akhir 60 - <72, nilai mutu =BC
     - hasil akhir <59 nilai mutu D
   - ✓ Tampilkan NPM, Nama, Nilai Rata-rata tugas, nilai UTS, nilai UAS dan Nilai Akhir serta Mutu

5. Sebagai contoh kriteria lulus seleksi untuk siswa jurusan IPA adalah 8,5 dan IPS 7,5. Apabila tidak memenuhi kriteria tersebut maka tidak lulus.



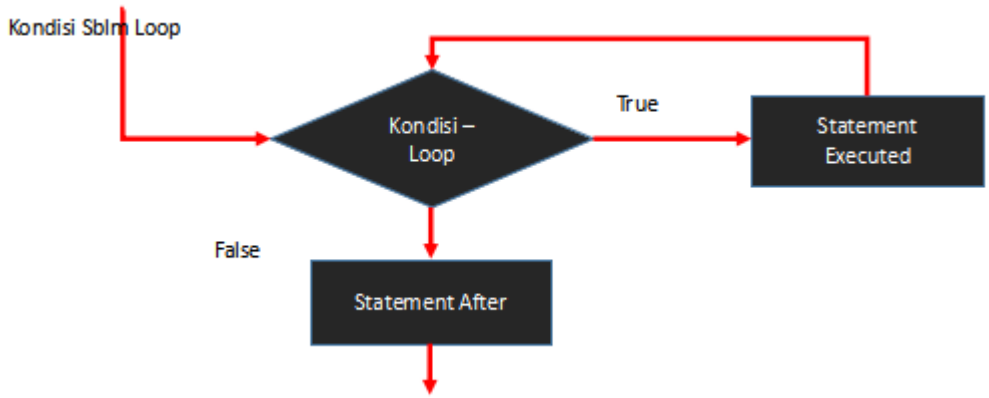
```
if(jurusan == "IPA") {

        if(nilai >= 8.5) {

           System.out.println("Anda lulus "); }

        else { System.out.println("Anda tidak lulus "); } } else
if(jurusan == "IPS") {

        if(nilai >= 7.5) {

           System.out.println("Anda lulus "); }

        else {

         System.out.println("Anda tidak lulus "); } }
```

BAB VII
KONTROL PROGRAM (PER-ULANGAN)

Struktur perulangan mirip pada struktur percabangan sama sama menggunakan simbol yang sama. Tetapi dari arti kata yang ada bahwa perulangan dan percabangan adalah beda strukturnya. Pada perulangan Struktur merupakan sesuatu untuk mengulang pernyataan sebanyak n nilai tertentu atau sampai suatu keadaan tidak memenuhi untuk dilakukan perulangan. Dari penjelasan diatas dapat diambil kesimpulan bahwa struktur perulangan menggunakan prinsip yang diterapkan pada struktur seleksi.
Walaupun percabangan dan perulangan itu mirip tetapi mereka juga memiliki perbedaan dimana pada percabangan mengulangan keadaan untuk memilih satu diantara beberapa pilihan yang sesuai dengan situasi yang diminta. Perulangan mengulang pernyataan sebanyak n tertentu. Dimana Nilai n adalah nilai bilangan utuh yang ditentukan sesuai kebutuhan. Yang perlu diingat adalah bahwa perulangan, apapun bentuknya, tidak mampu menangani data bertipe teks char maupun string. Penerapan perulangan dalam kenyataan yang real terlihat pada perulangan bilangan mundur dari nilai tertinggi ke nilai rendah 0 seperti pada contoh lampu merah di persimpangan.

Struktur kontrol pengulangan ini memiliki tiga (3) macam perulangan yaitu **while**, **do-while**, dan **for-loops**.
Ketiga struktur pengulangan ini memiliki pengulangan di dalamnya. Model struktur ini disebut dengan perulangan bersarang nested looping.

Flowchart Perulangan dapat dilihat dalam gambar dibawah ini.



Kontrol program perulangan terdapat dua jenis yaitu *Counted loop* dan *Uncounted loop*.

- ✓ *Counted loop*: merupakan perulangan dimana jumlah pengulangan sudah pasti naik satu atau turun satu.
- ✓ *Uncounted loop*: merupakan perulangan dimana jumlah pengulangan belum pasti tergantung dari counter yang dipakai dan bisa lebih dari Satu.
  - *Counted loop* untuk perulangan *For*
  - *Uncounted loop* untuk perulangan *While* dan *Do/While*

Perulangan FOR

Kontrol program pengulangan dengan model for ini adalah model pengulanan yang jumlah counter atau stepnya sudah pasti serta jumlah yang diulang juga sdh pasti. Kontrol program pengulangan for ini wajib memberikan batas nilai awal dan nilai akhir pengulangan, pengulangan for ini tidak memerlukan step karen stepnya sudah di pastikan diawal yaitu naik satu dan turun datu.

Bentuk umum pengulangan for sebagai berikut:

```
for (InitializationExpression; LoopCondition; StepExpression)
    { statement1;
     Statement2;
```

Keterangan

- InitializationExpression – inisialisasi dari variabel loop.
- LoopCondition - membandingkan variabel loop pada nilai batas tertentu
- StepExpression - melakukan update pada variabel loop.

Proses iteratif terus berlanjut selama keadaan loop benar. Proses iteratif hanya berhenti ketika kondisinya salah atau tidak lagi terpenuhi. Untuk konstruksi loop biasanya menggunakan variabel untuk mengontrol seberapa sering badan loop dieksekusi dan untuk menentukan kapan loop berhenti. Inilah yang disebut dengan variabel kontrol. Nilai inisialisasi merupakan variabel kontrol, proses inisialisasi inisialisasi hanya dilakukan satu kali. Fungsi iterate menambah (increments) atau decrements (mengurangi) nilai variabel kontrol, dan kondisi loop memerikasa apakah kondisi loop benar atau salah.

Untuk memudahkan pemahaman perhatikan contoh program berikut:

```
public class CetakAngka {
public static void main(String [] argumen){
for(int i=0; i<=10; i++) {
    System.out.print(I + " "); }
}
}
```

Contoh di atas, melakukan perulangan for dengan inisialisai nilai awal i dengan 1, dan meng-evaluasi kondisi loop (i <= 10), jika true maka secara berulang mengeksekusi statement pada tubuh loop . Aksi setalah iterasi (i++), adalah suatu statement yang memperbaharui variabel kontrol. Statement ini dijalankan setelah (akhir) iterasi. Statemen ini menaikan (increment) variabel kontrol. Pada saatnya, nilai variabel kontrol memaksa kondisi loop bernilai false. Sebab jika tidak ada pembehentian karena kondisya yang terus benar , maka perulangan akan menjadi tak berhingga (**infinite loop**).

Contoh program terjadinya **infinite loop**

```
public class latfor {
public static void main(String args[]){
for(int i=4; i>=0; ) {
    System.out.print(i); }
} }
```

Kesalahan pada pengulangan program yang terjadi adalah karena loop dalam keadaan selalau benar dan tidak pernah berhenti (infinite loop). Yaitu, program tidak bisa menghentikan loop karena kondisi-kondisi pengulangan selalu bernilai benar. Hal ini di karenakan tidak adanya increment atau decrement.

Contoh Lain untuk perulangan for yang turun atau pengurangan

```
public class latfor {
public static void main(String args[]){
for(int i=4; i>=0; i--) {
    System.out.print(i); }
}
}
```

Contoh di atas, melakukan perulangan for dengan inisialisai nilai awal i dengan 4, dan meng-evaluasi kondisi loop (i >= 0), jika true maka secara berulang mengeksekusi statement pada tubuh loop . Aksi setalah iterasi (i--), adalah suatu statement yang memperbaharui variabel kontrol. Statement ini dieksekusi setelah (akhir) tiap iterasi. Dengan i—artinya akan dimulai dari hal ini adalah 4 sampai turun ke 0.

Pengulangan While

Kontrol program pengulngan dengan while adalah pengulangan blok perintah yang diulang terus menerus selama kondisi masih bernilai benar. Sintaks perulangan while adalah sebagai berikut:

```
while (kondisi-loop){
//tubuh loop
statement-statement ;
}
```

Pernyataan di dalam while loop (badan loop) dieksekusi berkali-kali. Setiap kondisi loop adalah ekspresi boolean yang memeriksa isi loop. Kondisi dievaluasi untuk menentukan apakah tubuh loop akan dieksekusi atau tidak. Jika hasil evaluasi benar, maka isi loop dieksekusi, sedangkan jika salah, seluruh loop berhenti dan mengeksekusi perintah di bawah while loop (jika ada).
Perhatikan contoh source code berikut:

```
int hitung = 1;
while (hitung < = 10 ) {
System.out.println("Belajar Java Menyenangkan");
hitung++;
}
```

Pernyataan pengulangan while adalah pernyataan atau ekspresi yang diulang. Contoh di atas menunjukkan bahwa variabel count diinisialisasi ke 1, kemudian kondisi loop (count <= 10) dievaluasi, jika benar pernyataan tersebut menunjukkan Easy Learn Java dan kemudian menambahkan variabel count sebanyak satu (count++). Jika nilainya salah, loop berhenti. Output dari potongan kode sumber di atas adalah pernyataan Belajar Java Fun sebanyak 10 kali. Pastikan nilai kondisi loop diakhiri dengan false untuk menghentikan program.

Infinite Loop

Kesalahan pada pemrograman yang terjadi karena kondisi pengulangannya tidak pernah berhenti disbabkan karena kondisi yang selalau benar. Dengan kata lain, program tidak dapat keluar dari perulangan karena kondisi untuk melanjutkan perulangan selalu benar.

Perhatikan contoh berikut:
Contoh program ini merupakan kondisi pengulangan yang tidak akan pernah berhanti (**infinite loop**). Hal tersebut dikarenakan nilai hitung akan selalu bernilai <= 10 (**true**).

```
int hitung = 1;
while (hitung < = 10 ) {
System.out.println("Mudah Belajar Java");
}
```

Perulangan Do While

Kontrol program pengulangan do-while ini sebenarnya sama dengan kontrol pengulangan while. Yang membedakan adalah bahwa perulangan do-while melakukan pemeriksaan status perulangan pada akhir blok perulangan. Dampaknya adalah bahwa pada perulangan yang menggunakan struktur do-while, proses perulangan dijalankan minimal satu kali meskipun kondisi tidak terpenuhi (false)
Sintaks untuk perulangan **while** adalah sebagai berikut:

```
do {
//tubuh loop
statement-statement ;
} while (kondisi-loop);
```

Pernyataan di dalam do (body of the loop) dieksekusi terlebih dahulu. Kemudian keadaan loop dievaluasi. Jika hasil evaluasi benar, isi loop akan dieksekusi lagi; jika itu layak salah, loop berhenti. Dan lanjutkan kalimat berikutnya di bawah loop.Perhatikan contoh berikut:

```
int hitung = 1;
do {
//tubuh loop
System.out.println(hitung + " ");
hitung++;
} while (hitung <= 10);
```

Contoh di atas menunjukkan bahwa variabel count diinisialisasi ke 1, pernyataan dalam loop body dieksekusi (menampilkan nilai variabel count), dan kemudian ditambahkan 1 (count++). Kondisi loop dievaluasi setelah badan loop dieksekusi. Jika nilainya benar, badan loop dieksekusi lagi. jika nilainya salah maka putaran berhenti.
Output dari potongan kode sumber tersebut adalah angka 1 samai dengan 10 : 1 2 3 4 5 6 7 8 9 10

Untuk lebih memahami perbedaan antara konstruksi while dan do-while, lihat khususnya dua bagian dari kode sumber dari program berikut:

```
int i = 6;
while (i < 5) {
System.out.println("Belajar Java Menyenangkan");
i++;
}
```

Pada contoh di atas, blok while loop tidak dieksekusi. Hal ini di karenakan nilai Variabel i lebih besar dari 5, yang menyebabkan keadaan pengulangan tidak terpenuhi (nilai salah). Pertimbangkan contoh berikut, di mana perulangan menggunakan struktur do-while:

```
int i = 6;
do {
System.out.println("Belajar Java itu Mudah");
i++;
} while (i < 5 );
```

Pada contoh di atas, program mengeksekusi pernyataan pada blok repeat walaupun nilai variabel i lebih besar dari 5 (artinya kaeadaan pengulangan bernilai false). 1 kali. Ini karena struktur perulangan do-while memeriksa keadaan perulangan setelah pernyataan di badan perulangan dieksekusi. Jadi, meskipun pemeriksaan kondisi loop salah, pernyataan tetap ada di dalam Body loop tetap berlangsung (1 kali).

Perulangan Tersarang (Nested Loop)

Saat Anda membuat loop, ternyata itu berisi loop lain yang disebut loop bersarang. Loop bersarang berarti struktur loop berisi struktur loop lain. Ini dapat dilakukan dalam for, while, atau do..while loop. Jenis struktur loop ini disebut loop bersarang. Ini adalah loop dengan loop lain yang mungkin terhubung atau tidak terhubung. Rumus selajutnya menunjukkan struktur perulangan bersarang menggunakan struktur for, direpresentasikan dalam bentuk berikut:

```
for(inisialisasi; syarat; pengubah){
for(inisialisasi; syarat; pengubah){
statement ; }
}
```

Untuk memahami cara kerja loop bersarang, pertimbangkan untuk mengimplementasikan konstruksi loop bersarang menggunakan konstruksi for sebagai berikut:

```
public class Coba {
public static void main(String args[]) {
for(int i=1; i<=3; i++){
for(int j=1; j<=3; j++) {
    System.out.format("%d * %d = %d\n",i,j, i*j); }
    System.out.println(); }
}
}
```

Contoh Lain perulangan tersarang

```
public class ForBersarang{
public static void main(String[] args){
int i;
for (i = 1; i <= 5; i++){
    for (int j = 1; j <= i; j++){
      System.out.print(i + " "); }
      System.out.println(); }
}
```

Hasil dari program diatas ketika di jalankan akan menampilakan hasil seperti dibawah:

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Contoh program diatas bisa juga menggunakan struktur perulangan while maupun do while, seperti yang dicontohkan pada kode program berikut:

```
public class WhileBersarang{
public static void main(String[] args){
int i = 1;
int j;
while (i <= 5) {
  j = 1;
  while (j <= i) {

 System.out.print(i + " ");
    j++; }
    System.out.println();
     i++; }
}
}
```

Yang berikut dengan menggunakan struktur do-while

```
public class WhileBersarang{
public static void main(String[] args){
int i = 1;
int j;
do {
    j = 1;
    do {
        System.out.print(i + " ");
        j++; }
while (j <=i);
      System.out.println();
       i++; }
while (i <= 5);
}
}
```

Latihan

1. Menampilkan angaka genap kurang dari 20
   - ✓ Output:
   - ✓ 2 4 6 8 10 12 ...
2. Menampilkan bilangan sederet bilangan 1 s.d 100 dengan ketentuan setiap angka kelipatan tiga (3) yang ditampilkan kata "Tiga", setiap kelipatan lima (5) yang ditampilkan adalah kata "lima" dan setiap angka yang merupakan kelipatan 3 dan 5 yang ditampilkan adalah "Unindra"
   - ✓ Output:
   - ✓ 1 2 tiga 4 lima tiga 7 8 tiga lima 11 tiga 13 14 unindra ...

Praktikum Program Pengulangan dalam BlueJ

1. Mengulang kata Belajar Java sebanyak 10 kali dengan while-do

   Buatlah sebuah class dengan nama Pe_While , untuk menampung instruksi perintah pengulangan dengan menggunakan while, perintah untuk mengulang 10 x belajar java. Untuk lebih lengkapnya source code-nya dapat dilihat pada gambar dibawah. Serta hasil program pada gambar disampingnya.

```
public class Pe_While
{
    public static void main (String[]args){
        int hitung = 1;
        while (hitung <= 10){
        System.out.println("Belajar Java" + hitung);
        hitung++;}
    }
}
```

```
BlueJ: Terminal Window
Options
Belajar Java1
Belajar Java2
Belajar Java3
Belajar Java4
Belajar Java5
Belajar Java6
Belajar Java7
Belajar Java8
Belajar Java9
Belajar Java10
```

2. Sama dengan soal pertama menuliskan kata Belajar Java sebanyak 10 kali dengan while-do tetapi dengan counter atau skip sebanyak 2

```
public class P_While1
{
   public static void main (String[]args){
        int hitung = 1;
        while (hitung <= 10){
        System.out.println("Belajar Java" + hitung);
        hitung = hitung + 2;}
    }
}
```

```
BlueJ: Terminal Window
Options
Belajar Java1
Belajar Java3
Belajar Java5
Belajar Java7
Belajar Java9
```

Untuk contoh berikut adalah program pengulangan dengan menggunakan while, tetapi perulangan dalam bentuk negatif dan jarak pengulangan sebanyak 2 perintah untuk mengulang 10 x belajar java. Untuk lebih lengkapnya source code-nya dapat dilihat pada gambar diatas. Serta hasil program pada gambar disampingnya.

3. Sama dengan soal pertama tetapi dengan menggunakan pengulangan yang lain yaitu do-while, dengan penghitungan mulai dari 1 dengan counter 2 sampai pada batas nilai 15. Hasil yang di dapat sama dengan program sebelumnya.

```
public class P_Dowhile
{
    public static void main(String[]args){
    int hitung = 1;
    do {
        System.out.println("Java " + hitung );
        hitung = hitung + 2;}
    while (hitung <=15);
    }
}
```

```
BlueJ: Terminal
Options
Java 1
Java 3
Java 5
Java 7
Java 9
Java 11
Java 13
Java 15
```

Untuk hasil program dapat anda lihat pada gambar disampingnya yaitu java 1, java 3 sampai pada java 15

4. Contoh ke empat adalah sama dengan yang ke tiga dengan pengulangan do-while, dengan counter -5 dimaulai dari 100 sampai turun pada batas nilai diatas sama dengan = 0

```
public class P_Dowhile2
{

    public static void main(String[]args){
    int hitung = 100;
    do {
        System.out.println("Java " + hitung );
        hitung = hitung - 5;}
    while (hitung >=0);
    }
}
```

```
BlueJ: Terminal Window - Kontrol_Program
Options
Java 100
Java 95
Java 90
Java 85
Java 80
Java 75
Java 70
Java 65
Java 60
Java 55
Java 50
Java 45
Java 40
Java 35
Java 30
Java 25
Java 20
Java 15
Java 10
Java 5
Java 0
```

5. Contoh berikutnya adalah sama dengan yang sebelumnya yaitu program pengulangan dengan perintah for, dengan counter 1 dimulai dari 10 sampai pada batas nilai kurang dari 10, perintah ini akan mengulang dengan counter yang sudah pasti selalu naik satu atau turun satu. Dan pada contoh gambar dibawah ini pengulangan dengan for dan hasilnya.

```
public class P_For1
{
  public static void main(String[]args){
    int x;
    for (x=1; x<=10; x++){
        System.out.println("Java-BlueJ " + x);
    }
  }
}
```

```
BlueJ: Terminal Window - K
Options
Java-BlueJ 1
Java-BlueJ 2
Java-BlueJ 3
Java-BlueJ 4
Java-BlueJ 5
Java-BlueJ 6
Java-BlueJ 7
Java-BlueJ 8
Java-BlueJ 9
Java-BlueJ 10
```

6. Contoh ke empat adalah sama dengan yang ke tiga dengan pengulangan do-while, dengan counter -5 dimaulai dari 100 sampai turun pada batas nilai diatas sama dengan = 0

```
public class P_For2
{
   public static void main(String[]args){
       int i, j;
       for (i =1; i<=5; i++){
           for (j =1; j<=5; j++){
               System.out.print("\t" + i*j);
            }
       System.out.println();}
    }
}
```

```
BlueJ: Terminal Window - Kontrol_Program
Options
        1       2       3       4       5
        2       4       6       8       10
        3       6       9       12      15
        4       8       12      16      20
        5       10      15      20      25
```

7. Buatlah sebuah program perulangan untuk menampilkan hasil seperti contoh dibawah ini.
   - ✓ Output : 100 90 80 70 60 50 40 30 20 10
   - ✓ Output : 5 10 15 20 25 30 35 40 45 50
   - ✓ Output : 2 4 6 8 10 12 14 16 18 19
   - ✓ Output : 11 13 16 20 25 31 38 46 55 65

8. Buatlah skrip program Perulangan dan Percabangan untuk mencetak deret angka seperti berikut:
   - ✓ Output : 1 -2 3 -4 5 -6 7 -8 9 -10 11
   - ✓ Output : 11 2 13 4 15 6 17 8 19 10
   - ✓ Output : 1 10 2 9 3 8 4 7 5 6

BAB VIII
ARRAY

Apa yang harus dilakukan ketika Anda perlu menyimpan banyak data dalam sebuah variabel...? Misalnya, kami ingin menyimpan nama teman dalam sebuah variabel. Mungkin begitulah cara kami melakukannya.

- ✓ String namasahabat1 = "Ani";
- ✓ String namasahabat1 = "Ana";
- ✓ String namasahabat1 = "Martin";
- ✓ String namasahabat1 = "Wira";
- ✓ String namasahabat1= "Dira";

Ini wajar dan legal, tapi masalahnya, bagaimana jika datanya banyak, misalnya 500 data? Membuat begitu banyak variabel pasti akan melelahkan. Oleh karena itu kita dapat menyimpan semuanya dalam sebuah tabel.

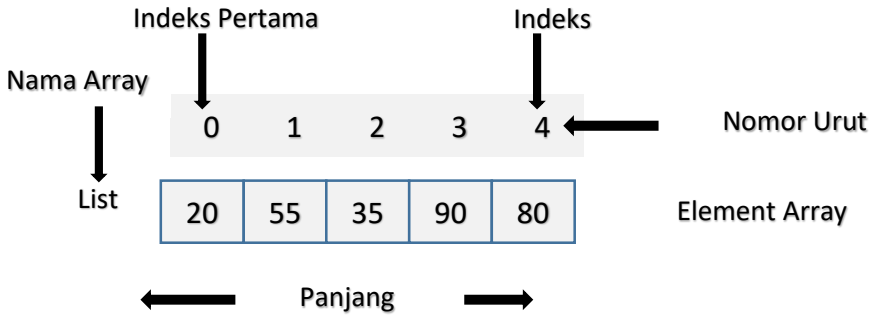
Array, atau sering disebut sebagai array, adalah kumpulan data dengan indeks terurut dari tipe data yang sama. Array dapat dikatakan sebagai sekelompok variabel dengan tipe yang sama, disebut dengan nama yang sama, dimana setiap elemen variabel memiliki nilai indeks dan dapat diakses secara individual secara acak oleh indeks. Setiap elemen array dapat menyimpan tipe data (variabel). Array adalah panjang tetap, artinya jika Anda mendeklarasikan array dengan panjang 10, panjang array tersebut tetap 10 meskipun kita hanya menggunakan 5 elemen. Indeks array selalu berupa bilangan bulat yang dimulai dari nol.

Dataset terdiri dari hal-hal seperti penggaris, buku, pulpen, pensil, dan alat tulis lainnya, dimana datanya menggunakan tipe data String. Data ini dapat dikumpulkan dalam tabel "paperwork" yang tipe datanya adalah string. Tabel dibagi menjadi beberapa struktur tergantung pada representasi dan kompleksitas data. Ketika sebuah array hanya berisi satu string data, itu disebut array satu dimensi. Namun, ketika kumpulan data terdiri dari beberapa atau lebih kumpulan data terurut, itu disebut array multidimensi.
Singkatnya, dengan tabel adalah:

- ✓ Array adalah sebuah variabel yang digunakan untuk menampung banyak data dalam satu variabel.
- ✓ Array merupakan sebuah kumpulan dari tipe data yang sejenis atau sama dan dengan ukuran tetap dan berindeks.
- ✓ Tabel menggunakan indeks untuk menyederhanakan data yang disimpan di dalamnya.
- ✓ Indeks array selalu dimulai dari 0 sampai N... dan perhatikan bahwa indeks tidak harus berupa angka, tetapi bisa juga berupa karakter atau teks.

Array satu dimensi

Jika Anda ingin menggunakan larik dalam suatu program, Anda harus mendeklarasikan variabel yang merujuk ke larik dan menentukan jenis elemen larik. Array satu dimensi adalah kumpulan data serupa yang struktur datanya hanya mewakili satu kolom atau baris. Struktur array satu dimensi seperti yang ditunjukkan pada gambar di bawah ini:



Beberapa komponen yang muncul pada struktur tabel di atas antara lain nama array, komponen , dan indeks array. Nama tabel adalah variabel dari tipe tertentu, yang nantinya berisi data yang tipenya cocok dengan tipe tabel. Pada saat yang sama, elemen adalah kumpulan data yang disimpan dalam array, diatur oleh indeks berurutan. Indeks adalah bilangan utuh dan tidaka bolwh bilangan real yang memainkan peran penting dalam menemukan data. Indeks menunjukkan lokasi data ketika indeks disebutkan. Misalnya, item 55 disimpan dalam tabel daftar di lokasi Indeks 1, di mana Indeks 1 adalah item data kedua.

Deklarasi Larik

Array pertama-tama harus dideklarasikan, baik nama array maupun jenis dan ukurannya. Sehingga array yang dihasilkan dapat menyimpan data. Deklarasi array tidak hanya mendefinisikan tipe dan ukuran, tetapi juga dimensi array. Jika Anda ingin menggunakan larik dalam suatu program, Anda harus mendeklarasikan variabel yang merujuk ke larik dan menentukan jenis elemen larik. Ada dua cara untuk mendeklarasikan array satu dimensi:

Bentuk-1 = Tipe_data [ ] nama_array;

Bentuk-2 = Tipe_data nama_array [ ];

Contoh :

```
String [ ] Nama;
Int Angka [ ];
```

Instansiasi

Proses penerapan instansiasi ini merupakan proses untuk menentukan ukuran array, namun seringkali juga menginisialisasi data array yang dikandungnya. Struktur berikut dapat digunakan untuk mengekspresikan array:

tipedata namalarik[]=new tipedata[ukuran];

Contoh :

String[] nama = new String[5] atau bisa juga

Int [] Angka;

Angka = new int[5];

String[] mahasiswa = {"Caca","Cici","Cucu","Coco","Cece"}

Sekarang kita memiliki contoh struktur tabel yang ditunjukkan pada gambar di atas. di mana nama array adalah daftar tipe integer 5, yang juga merupakan tipe integer. Kemudian acara array dapat ditulis sebagai:int nilai[]= new int[5];
Inisialisasi data dapat ditulis sebagai berikut:

nilai [0]=20;
nilai [1]=55;
nilai [2]=35;
nilai [3]=90;
nilai [4]=80;

Anda juga dapat menginisialisasi anggota data array memakai bentuk yang secara langsung akan memangkas deklarasi array, seperti dalam struktur berikut:

int nilai []={20,55,35,90,80};

Untuk menjelaskan tabel di atas, perhatikan contoh tabel satu dimensi berikut ini:

```
public class ArraySatu {
public static void main(String args[]) {
int nilai[] = {51,66,72,80,91};
for(int i=0; i<list.length; i++){
System.out.println("Elemen ke " +(i+1)+ " : "+nilai[i]); }
}
}
```

Contoh Program Array Satu Dimensi

```
import java.util.Scanner;
public class Larik1 {
public static void main (String args[]) {
Scanner masuk=new Scanner(System.in);
float nilai[]=new float[5];
System.out.println("masukkan 5 buah data nilai");
for(int i=0;i<5;i++) {
System.out.print("Data ke"+(i+1)+": ");
nilai[i]=masuk.nextFloat(); }
System.out.println("data nilai yang dimasukkan");
for(int i=0;i<5;i++)
System.out.println(nilai[i]); }
}
```

Mengakses Elemen Array

Elemen array dapat digunakan baik secara individual dengan menentukan nomor indeks dalam tanda kurung siku "[]" yang mengacu pada indeks data. Harap diperhatikan juga bahwa nama larik harus disertakan sehingga penerjemah mengetahui variabel larik yang mana sesuai dengan permintaan. Misalnya, jika kita ingin menggunakan data dengan nilai 80 dalam daftar tabel, kita melakukannya dengan memanggil nama tabel dan nomor indeks, yang menggunakan struktur sebagai nilai[3] untuk menunjukkan lokasi data. Ditulis secara lengkap dengan struktur sebagai berikut:

```
System.out.println("Elemen ke " + " : " +nilai[3]);
```

Jadi, hasil ekspresi ini menampilkan informasi elemen: 80

Array / Larik dua dimensi

Larik dua dimensi/tingkat berbeda dari array satu tingakat. Dalam array satu dimensi, ia hanya berpatokan pada satu arah vektor saja, baik itu hanya vektor baris atau vektor kolom. Sekarang, dalam larik dua tingkat, dua komponen terdiri dari baris dan kolom yang melengkapi larik tersebut. Dalam array dua tingkat, indeks baris umumnya diketikan pada sebelah kiri, kemudian indeks kolom di sebelah kanan bagian definisi array. Indeks baris dan kolom dalam larik dua tingkat adalah komponen yang merepresentasikan larik tersebut sebagai matriks atau larik. Untuk mendeklarasikan array dua dimensi:

```
Tipedata namalarik[ukuran baris][ukuran kolom];

TipeData[ ][ ] nama_variabel[=new tipeData[jumlah_baris] [jumlah_kolom]];
```

Contoh :

```
int [ ][ ] matrix;

matrix = new int[4][3]
```

Untuk membuat array berbentuk matriks berukuran 3 x 3

Membuat matriks dalam bentuk matriks 3 x 3

Untuk memahami struktur array dua dimensi, berikut adalah struktur matriks pada memori komputer sebagai berikut:

```
int [][] matrix = { {1,2,3} , {4,5,6} , {7,8,9} , {10,11,12} };
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3	10	11	12

Perhatikan pada gambar diatas bahwa kita memiliki matriks yang disebut Matriks [3][3], nilai 3 mewakili jumlah baris dan nilai 3 mewakili kolom. Jadi, tabel adalah tabel yang berbentuk matriks (array) yang mampu menyimpan elemen data dari suatu tabel dengan 3 larik dan 3 lajur. Terlihat juga bahwa elemen dari data array adalah angka-angka di dalam kotak. Jika ingin menggunakan komponen pada data tersebut dengan nilai 9, dapat ditulis dalam format matrix[2][3].

Mengakses Elemen Array Dua Dimensi

Mengakses elemen array dua dimensi sama dengan mengakses elemen array satu dimensi. Namun, dalam array dua dimensi, elemen kolom dan baris harus dicantumkan. Misalnya, jika kita ingin menggunakan item dengan nilai 10, kita harus menyertakan indeks baris dan kolom yang menunjukkan data 10. Kita lihat item 10 berada pada posisi indeks larika ke-4 dan lajur ke-1 , jadi ini ditulis dengan struct list[3][0]; Mengapa? Karena indeks baris dan kolom dimulai dari 0, indeks baris dan kolom ketiga menjadi posisi 4

Duplikasi Array

Jika Anda ingin menyalin sebuah array dari kelas sistem ke yang lain menggunakan metode arraycopy(), atau dengan kata lain, isi array dapat disalin ke array lain menggunakan perintah Arraycopy(). Ada beberapa cara untuk menyalin array di Java, antara lain:

- ✓ Menggunakan metode static arraycopy yang ditemukan di kelas System. Cara yang lebih sederhana dan tidak terlalu intensif kode adalah dengan menggunakan metode static arraycopy, yang melanggar konvensi penamaan Java tetapi berfungsi dengan benar. Untuk menggunakannya, kita harus melakukan hal berikut:

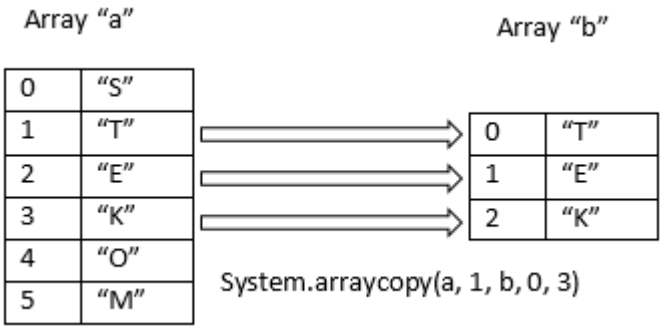
```
char[] karakterSumber = {'a', 'b', 'c', 'd', 'e', 'f', 'g'};
char[] karakterTarget = new char[karakterSumber.length];
for (int x = 0; x <= karakterSumber.length; x++)
  karakterTarget[x] = karakterSumber[x];
```

- ✓ Salin semua nilai elemen array dengan for loop. Ini adalah cara termudah dari semuanya, meskipun tujuannya agak sulit untuk diterapkan dan memungkinkan kita untuk mendapatkan tabel baru yang merupakan salinan dari yang asli dan sepenuhnya independen darinya. Untuk membuat salinan ini, mari lakukan hal berikut:
- ✓ Format penulisan yang umum digunakan menggunakan arraycopy

```
System.arraycopy(array1, p1, array2, p2, n)
```

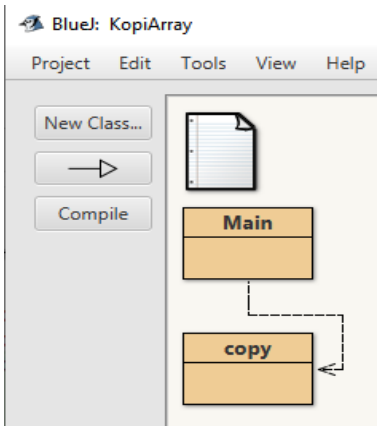
Keterangan
Tabel1 = tabel asal/sumber
Tabel2 = tabel target salinan
P1 = tampilan nilai asli salinan sumber
P2 = tampilan nilai asli salinan target
N = jumlah elemen tabel yang akan disalin (yang akan disalin).

Ilustrasi pada penyalinan array



Dimana pada ilustrasi diatas adalah kata STEKOM sebagai sumber dan selanjutnya akan di copykan hanya pada huruf tertentu, pada contoh diatas adalah huruf T, E dan K. Ini adalah ilustrasi dari array copy.

Contoh Program CopyArray (Dupliaksi Array)



Copy.java

```
import java.util.Arrays;
import java.util.Random;
public class copy
{
    private int[] A;
    private int[] B;
    private int[] C;
    private int[] D;
    private int[] E;
    private int[] F;

    private static Random valueGen = new Random();

    public copy(int newSize)
    {
        A=new int[newSize];
        B=new int[newSize];
        C=new int[newSize];
        D=new int[newSize];
        E=new int[newSize];
```

```
        for (int i=0; i<newSize; i++)
        {
            A[i]=1+ valueGen.nextInt(120);
        }
    }

    public int[] getA()
    {
        return A;
    }

    public void EditA(int value, int index)
    {
        A[index]= value;
    }

    public void setB(int[] newA)
    {
         for(int i=0; i< newA.length; i++)
            {
                B[i] = newA[i];
            }
    }
```

```
    public int[] getB()
    {
      return B;
    }

     public void setC(int[] newA)
    {
        C = Arrays.copyOf(newA, newA.length);
    }
    public int[] getC()
    {
      return C;
    }

     public void setD(int[] newA)
    {
        D = Arrays.copyOfRange(newA, 0, newA.length);
    }
    public int[] getD()
    {
      return D;
    }
```

```
    public void setE(int[] newA)
    {
        System.arraycopy(newA, 0, E, 0, newA.length);
    }
    public int[] getE()
    {
      return E;
    }


    public void testData(int[] data)
    {
        for(int i=0; i<data.length; i++)
        {
            System.out.print(data[i]+" ");
        }
    }
}
```

Main.java

```
import java.util.Arrays;
import java.util.Random;
public class Main
{
    public static void main(String[] args) throws CloneNotSupportedException {
      System.out.println("\u000c");
      copy cp = new copy(10);
      cp.setB(cp.getA());
      cp.setC(cp.getA());
      cp.setD(cp.getA());
      cp.setE(cp.getA());

      System.out.print("-----Nilai awal array A--------\n");
      cp.testData(cp.getA());

      cp.EditA(1000, 0);
      System.out.print("\n---Nilai array A setelah diedit----\n");
      cp.testData(cp.getA());

      System.out.print("\n---Mencopy array dengan for loop----\n");
      cp.testData(cp.getB());
      System.out.print("\n---Mencopy array dengan method static arraycopy---\n");
      cp.testData(cp.getD());}
}
```

Hasil dari copy array dengan inmport java.until.random

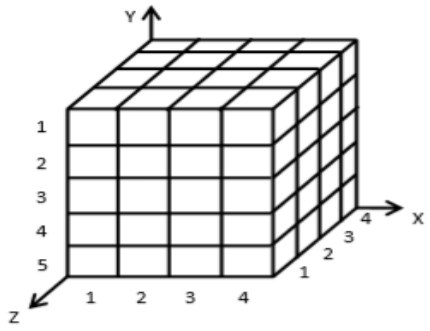
```
BlueJ: Terminal Window - KopiArray
Options

-----Nilai awal array A--------
97 73 34 38 51 70 45 50 17 108
---Nilai array A setelah diedit----
1000 73 34 38 51 70 45 50 17 108
---Mencopy array dengan for loop----
97 73 34 38 51 70 45 50 17 108
---Mencopy array dengan method static arraycopy---
97 73 34 38 51 70 45 50 17 108
```

Array Multidimensi

Properti tabel banyak dimensi adalah memiliki indeks yang terdiri dari larik dan lajur kolom dari halaman yang membentuk ruang Pola sederhana dari array banyak dimensi adalah array dua dimensi dengan hanya indeks baris dan kolom di kedua sisinya. Array banyak dimensi bisa lebih kompleks, dan ujung-ujungnya dapat dilihat dari atas, bawah, kiri atau kanan.

Dalam pemrograman terformat dalam basis teks, array banyak dimensi sulit untuk dilihat dan dipahami karena sisi yang mewakili ruang tersebut tidak mudah kelihatan. Pemrograman Array banyak dimensi lebih mudah dipahami dalam pemrograman komputer grafis karena keluaran dari program dapat dilihat dari perspektif yang berbeda. Berikut ini adalah penerapan array banyak dimensi terhadap tiga dimensi. Untuk larik 3D, gunakan konstruksi Java berikut:

Tipedata namalarik[ukuran baris][ukuran kolom][Tinggi]};



Contoh

int list[3][3][4];

Gambar diatas adalah pola pada , kode Java lengkap untuk array tiga dimensi terlihat seperti ini:

```
public class Tes{
public static void main(String args[]){
int list[][][] = new int[3][4][2];
int i, j, k, num=1;
for(i=0; i<3; i++){
for(j=0; j<4; j++){
for(k=0; k<2; k++){
list[i][j][k] = num;
num++; }
}
}
```

```
for(i=0; i<3; i++){
for(j=0; j<4; j++){
for(k=0; k<2; k++){
System.out.print("list[" +i+ "][" +j+
"][" +k+ "] = " +list[i][j][k]+ "\t"); }
System.out.println(); }
System.out.println(); }
}
}
```

Hasil program tersebut jika dijalankan akan mengahasilkan

```
List[0][0][0] = 1        List[0][0][1] = 2
List[0][1][0] = 3        List[0][1][1] = 4
List[0][2][0] = 5        List[0][2][1] = 6
List[0][3][0] = 7        List[0][3][1] = 8
List[1][0][0] = 9        List[1][0][1] = 10
List[1][1][0] = 11       List[1][1][1] = 12
List[1][2][0] = 1 3      List[1][2][1] = 14
List[1][3][0] = 1 5      List[1][3][1] = 16
List[2][0][0] = 17       List[2][0][1] = 18
List[2][1][0] = 19       List[2][1][1] = 20
List[2][2][0] = 21       List[2][2][1] = 22
List[2][3][0] = 23       List[2][3][1] = 24
```

Kedua format cetak ini sama-sama sulit dibedakan dengan dimensi yang memiliki ruang. Oleh karena itu, pemrograman terformat banyak dimensi lebih baik ditampilkan dalam dua dimensi karena halamannya yang mudah dipahami. Namun, pemahaman array multidimensi harus dipahami sebagai bentuk awal dalam prmrograman data disimpan didalam memori komputer.

Akses Array Multidimensi

Mengakses elemen array multidimensi sama dengan mengakses elemen array dua dimensi. Namun, array multidimensi harus berisi elemen kolom, baris, dan tinggi. Misalnya, jika kita ingin menggunakan item dengan nilai 10, kita harus menyertakan indeks baris, kolom, dan tinggi yang mengarah ke data item 10. Kita melihat bahwa item 10 berada di posisi indeks baris kedua. , kolom ke-1 dan tinggi ke-2, ditulis dengan struct list[1][0][1]; Mengapa? Karena indeks pada baris dan kolom dimulai dari 0, indeks baris dan kolom ketiga menjadi posisi 4.

Array List

Sebenarnya ada beberapa kekurangan dari tabel yang kita lihat sebelumnya, seperti:

- ✓ Data tidak dapat disimpan dalam tipe data yang berbeda
- ✓ Ukurannya tidak dinamis karena sudah ditentukan sebelumnya

Dalam bahasa pemrograman Java, ArrayList adalah kumpulan yang memungkinkan Anda untuk

menampilkan list atau daftar nilai yang bersifat dinamis dan juga dapat diubah. Kita dapat mengubah ukuran ArrayList dengan menambahkan (menambahkan) atau menghapus (menghapus) itu. Berbeda dengan array pada umumnya, kita tidak bisa menggunakan tipe data sederhana seperti int, char, boolean, float, dll. di ArrayList.

Perbedaan antara Array dan ArrayList lainnya adalah karena ArrayList memiliki properti dinamis, kita dapat menetapkan nilai apa pun berdasarkan metode objek yang kita gunakan. Pada materi ini, kita akan belajar bagaimana menggunakan ArrayList dan perbedaannya dengan Array pada program Java.

Tabel Perbedaan Array dan ArrayList 8.1

Array	ArrayList
Tidak dapat menggunakan wildcard	Dapat menggunakan generik untuk menjaga keamanan tipe data dalam array
Sifatnya statis, ukuran data tidak dapat berubah tergantung kapan pertama kali dibuat/didefinisikan.	Ini dinamis, ukuran data dapat diubah dengan menambahkan atau menghapus
Sifat arrat yang Statis.	Sifatnya yang dinamis.

- ✓ **size()**, untuk mencari panjang *ArrayList*
- ✓ **add()**, untuk menambah elemen baru
- ✓ **get()**, untuk mengambil elemen pada indeks tertentu
- ✓ **isEmpty()**, untuk memeriksa apakah *ArrayList* kosong atau tidak
- ✓ **indexOf()**, untuk mengetahui indeks dari suatu nilai
- ✓ **contains()**, untuk memeriksa apakah suatu nilai ada dalam *ArrayList*
- ✓ **set()**, untuk menimpa nilai pada indeks tertentu
- ✓ **remove()**, untuk menghapus nilai pada indeks tertentu

Dengan menggunakan daftar array ini, pertama-tama kita perlu mengimpor kelasnya, seperti:

import java.util.ArrayList;

Kemudian kita dapat membuat objek Array List seperti ini:

ArrayList cobaarraylist = new ArrayList();

Untuk menggunakan array di Java, pertama-tama harus dideklarasikan menggunakan metode berikut:

List<TipeData identifier > objek = new ArrayList<>();

ArrayList dynamicArray = new ArrayList ()

Contoh ArrayList

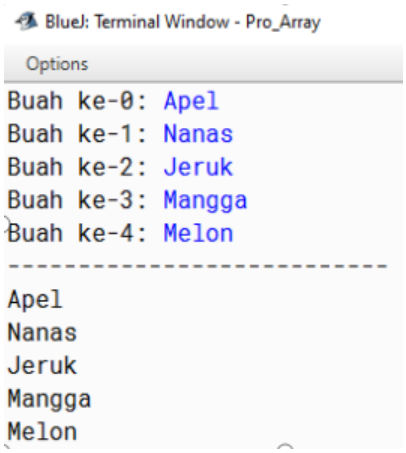
```
import java.util.ArrayList;

public class BangunDatar {

public static void main(String[] args) {

    // membuat objek array list

    ArrayList lingkaran = new ArrayList();

    // Mengisi Lingkaran dengan 5 data

    lingkaran.add("Jari-Jari");

    lingkaran.add(100);

    lingkaran.add("diameter");

    lingkaran.add(3.14);

    lingkaran.add(true);

    // menghapus diameter dari lingkaran

    lingkaran.remove("diameter");

    // Menampilkan isi dari lingkaran

    System.out.println(lingkaran);

    // menampilkan banyak isi data dari lingkaran

    System.out.println("Lingkaran berisi "+ lingkaran.size() +" data");

  }

}
```

Praktikum Program Array dalam Java BlueJ

1. Program Array Satu Dimensi

```
import java.util.Scanner;
public class Array1
{
    public static void main(String[] args) {
        String[] buah = new String[5];
        Scanner scan = new Scanner(System.in);

        for( int i = 0; i < buah.length; i++ ){
            System.out.print("Buah ke-" + i + ": ");
            buah[i] = scan.nextLine();
        }

        System.out.println("--------------------------");
        for( String b : buah ){
            System.out.println(b);
        }
    }
}
```

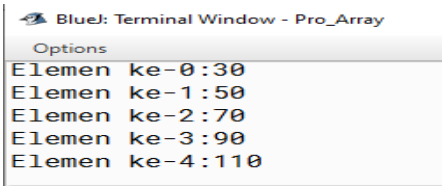
Buatlah class dengan nama Array1 dimana pada kelas ini akan memuat perintah untuk menampilkan program array dalam bentuk satu dimensi. Dimana pada program array ini tetap menggunakan perintah untuk pengulangan dengan For. Silahkan lengkapi kode programnya diatas dan jika benar maka akan seperti tampilan dibawah ini.
Hasil dari program diatas adalah seperti terlihat dibawah ini



Contoh program diatas adalah untuk menampilkan array satu dimensi dengan menggunakan java bluej. Dimana program diatas meminta inputan nama buah sebanyak 5x dan hasilnya akan ditampilkan, karena array ini hanya satu dimensi maka yang ditampilkan adalah baris barisnya saja.

2. Program Array Satu Dimensi

```
public class Array12
{
   public static void main(String[] args){
int myArray[]={30,50,70,90,110};
for (int i=0;i<myArray.length;i++)
System.out.println("Elemen ke-"+i +":"+myArray[i] );
 }
}
```



3. Program Array dua Dimensi

```java
import java.util.Scanner;
public class Duadimensi
{
 public static void main(String[] args) {
    // Membuat Array dan Scanner
    String[][] kursi = new String[2][3];
    Scanner scan = new Scanner(System.in);
    // mengisi setiap meja
    for(int bar = 0; bar < kursi.length; bar++){
        for(int kol = 0; kol < kursi[bar].length; kol++){
            System.out.format("Siapa yang akan duduk di Kursi (%d,%d): ", bar, kol);
            kursi[bar][kol] = scan.nextLine();
        }
       }
        // menampilkan isi Array
        System.out.println("-------------------------");
        for(int bar = 0; bar < kursi.length; bar++){
            for(int kol = 0; kol < kursi[bar].length; kol++){
                System.out.format("| %s | \t", kursi[bar][kol]);
            }
            System.out.println("");
        }
        System.out.println("-------------------------");
    }
}
```

Hasil Program

```
BlueJ: Terminal Window - ArrayDimensi
 Options
Siapa yang akan duduk di Kursi (0,0): Budi
Siapa yang akan duduk di Kursi (0,1): Martin
Siapa yang akan duduk di Kursi (0,2): Tini
Siapa yang akan duduk di Kursi (1,0): Wira
Siapa yang akan duduk di Kursi (1,1): Dira
Siapa yang akan duduk di Kursi (1,2): Dwik
-------------------------
| Budi |        | Martin |      | Tini |
| Wira |        | Dira |        | Dwik |
-------------------------
 Can only enter input while your programming i
```

Contoh program ini adalah untuk menampilkan array dua dimensi dengan menggunakan java bluej. Pada program diatas meminta inputan dalam bentuk baris dan kolom, karena array ini dua dimensi maka yang ditampilkan adalah baris dan kolomnya seperti matrik 2X3.

4. Program Array dua Dimensi

```java
public class Array21
{
   public static void main(String[]args){
    String[][]huruf = {
        {"A","B","C"},
        {"D","E","F"},
        {"G","H","I"}
    };
    //Menampilkan isi array pada indeks ke-[0][1] dan [2][0]
    System.out.println(huruf[0][1]);
    System.out.println(huruf[2][0]);
    }
}
```

```
BlueJ: Terminal Window - Pro_Array
Options
B
G
```

Contoh program ini adalah cara mengakses array dua dimensi, dimana pada array sudah ada daftarnya sehingga ketika akan menampilkan posisi dari isi array tersebut dengan menentukan posisi baris dan kolom, seperti contoh diatas posisi[0][1] dan [2][0] akan menghasilkan abjad pada posisi baris 1 dan kolom 2 yaiti “B” dan menampilkan posisi pada baris ke-3 dan kolom ke-1 yaitu abjad “G”

5. Contoh lain Array dua Dimensi

```
import java.util.Scanner;
public class Array23
{
    public static void main(String[] args){
        Scanner masuk=new Scanner (System.in);
        System.out.print("Jumlah Mahasiswa : ");
        int n=masuk.nextInt();

        String mahasiswa[][] = new String[n][3];
        for(int i= 0; i<n; i++){
            System.out.println("");
            System.out.println("Data Mahasiswa ke "+(i+1));

            for(int j=0;j<3;j++){
                if (j == 0)
                    System.out.print("NIM      :");
                else if (j == 1)
                    System.out.print("Nama     : ");
                else
                    System.out.print("Jurusan  : ");
                System.out.print("");
                mahasiswa[i][j] = masuk.next();
            }
        }
        System.out.println("Data Mahasiswa yang dimasukan");
        System.out.println("-----------------------------");
        System.out.println("NIM \t\t\t  NAMA \t\t  JURUSAN \t");

        for(int i=0;i<n;i++){
                    for(int j=0;j<3;j++)
            {
                System.out.print(mahasiswa[i][j]+"\t\t");
            }
        System.out.println();}
    }
}
```

```
BlueJ: Terminal Window - Pro_Array
Options
Data Mahasiswa yang dimasukan
-----------------------------
NIM                     NAMA            JURUSAN
123             Budi            TI
234             Martini         SI
345             Martin          DG
456             Wira            MI
```

Contoh program array dua dimensi, dimana pada array tersebut di inputkan dengan model matriks 4 baris dan 3 kolom dengan cara di inputkan.

6. Contoh Program dengan fungsi Add dan Delete Array dua Dimensi

```
import java.util.ArrayList;
public class ListArray
{
    public static void main(String[] args) {
        // membuat objek array list
        ArrayList daftar = new ArrayList();

        // Mengisi kantong ajaib dengan 5 benda
        daftar.add("Kelinci");
        daftar.add("Sepatu");
        daftar.add(10000);
        daftar.add(58.50);
        daftar.add(true);

        // menghapus kelinci dari kantong ajaib
        daftar.remove("Kelinci");
```

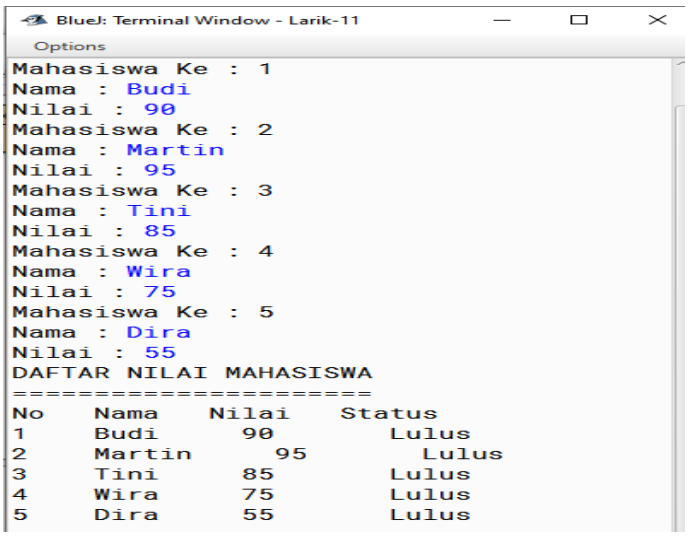
```
BlueJ: Terminal Window - ArrayDimensi
Options
[Sepatu, 10000, 58.5, true]
Daftar 4 item
```

```
ajaib
antong ajaib
"+ daftar.size() +" item");
```

7. Contoh Program Array dengan Percabangan

```
import java.util.Scanner;
public class Larik12
{
    public static void main (String args[])
    {
        System.out.println("\u000c");
    Scanner input=new Scanner(System.in);

    System.out.print("Masukan Jumlah Mahasiswa : ");
    int n=input.nextInt();

    String nama[]=new String[n];
    String status[]=new String[n];
    int nilai[]=new int[n];

    for (int i=0;i<n;i++){
         System.out.println("Mahasiswa Ke : "+(i+1));
        System.out.print("Nama : ");
        nama[i]=input.next();

        System.out.print("Nilai : ");
        nilai[i]=input.nextInt();

        if (nilai[i]<=50) {
                status[i]="Tidak Lulus";
            } else {
                status[i]="Lulus";
            }
    }
    System.out.println("DAFTAR NILAI MAHASISWA");
    System.out.println("======================");
    System.out.println("No   Nama   Nilai   Status  ");

    for (int i=0; i<n;i++) {
        System.out.println((i+1)+"    "+nama[i]+"     "+nilai[i]+"       "+status[i]);
    }

    }
}
```

Hasil Program Array dengan percabangan


BlueJ: Terminal Window - Larik-11
Options
Mahasiswa Ke : 1
Nama : Budi
Nilai : 90
Mahasiswa Ke : 2
Nama : Martin
Nilai : 95
Mahasiswa Ke : 3
Nama : Tini
Nilai : 85
Mahasiswa Ke : 4
Nama : Wira
Nilai : 75
Mahasiswa Ke : 5
Nama : Dira
Nilai : 55
DAFTAR NILAI MAHASISWA
==========
No Nama Nilai Status
1 Budi 90 Lulus
2 Martin 95 Lulus
3 Tini 85 Lulus
4 Wira 75 Lulus
5 Dira 55 Lulus

BAB IX
KONSEP PBO (PEWARISAN, PENGKAPSULAN , POLYMORPHIS)

Ada tiga konsep dasar pada pemrograman berbasis objek paling utama untuk dipahami bila Anda ingin belajar pemrograman. Ketiga konsep tersebut adalah: modulasi/enkapsulasi, pewarisan/warisan, polimorfisme.

Modulasi (Encapsulation) / Pengkapsulan

Enkapsulasi adalah teknik membuat kelas privat (pribadi) dan menyediakan akses melalui metode (publik). Jika dideklarasikan private (publik), tidak tersedia untuk siapa pun di luar kelas, jadi disembunyikan di bidang kelas.

Enkapsulasi merupakan sebuah metode untuk information hiding attribut suatu kelas. Terdapat dua cara untuk memodulasi, yaitu:

- ✓ Sembunyikan informasi
- ✓ Menyediakan perantara (metode) untuk memperoleh informasi

Tujuan utama modulasi adalah untuk memastikan bahwa informasi "sensitif" tetap tersembunyi dari pengguna. Untuk mencapainya adalah:

- ✓ Mendeklarasikan variabel atau atribut kelas bersifat pribadi
- ✓ Menyediakan metode publik untuk mengakses dan memperbarui nilai variabel pribadi.

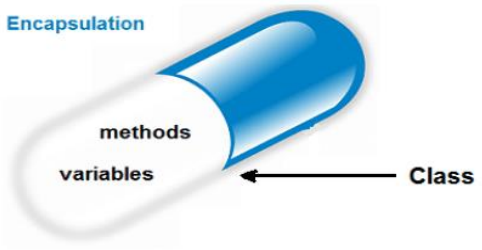
Keuntungan utama enkapsulasi adalah kemampuan untuk mengubah kode yang kita gunakan tanpa merusak kode orang lain yang menggunakan kode kita. Berkat fitur ini, enkapsulasi menawarkan fleksibilitas dalam pengembangan kode program kami.

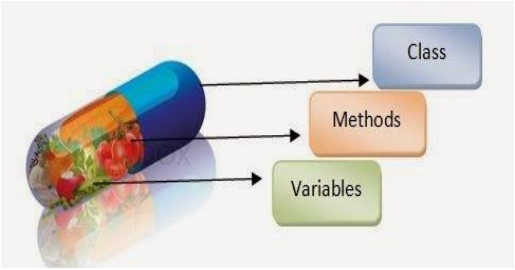
Mengapa enkapsulasi?

- ✓ Manajemen atribut dan metode kelas yang lebih baik
- ✓ Atribut class dapat dibuat read-only (jika hanya menggunakan getmethod) atau read-only (jika hanya menggunakan setmethod).
- ✓ Fleksibel : Seorang programmer dapat mengubah bagian dari kode tanpa mempengaruhi bagian lain
- ✓ Meningkatkan keamanan informasi

Untuk mengimplementasikan enkapsulasi, kami tidak ingin objek apa pun mengakses data kapan saja. Untuk melakukan ini, kami mendeklarasikan atribut kelas sebagai pribadi. Namun, terkadang kami ingin objek lain mengakses data pribadi. Dalam hal ini kami menggunakan metode yang bermanfaat. Metode aksesor digunakan untuk membaca nilai variabel kelas, apakah itu variabel instan atau statis. Metode akses biasanya dimulai dengan mengetik get. Metode ini juga memiliki nilai balik.

Jika kita ingin objek lain mengubah data, kita dapat membuat metode yang dapat mengatur atau mengubah nilai peubah di dalam kelas, baik secara instance atau tetap. Metode seperti ini disebut sebagai metode mutator. Metode mutator biasanya ditulis sebagai array.





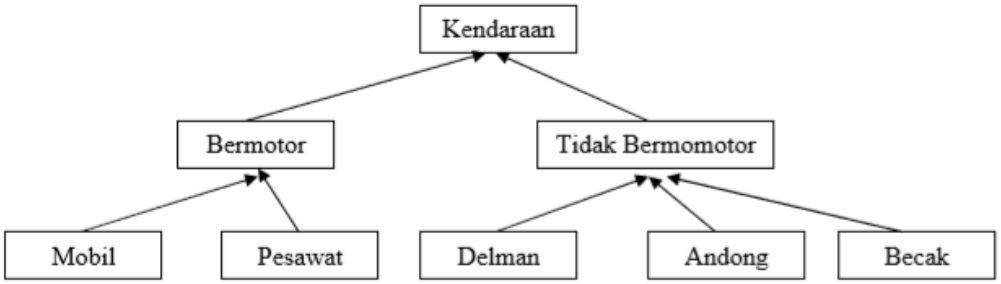
Contoh:
Mobil itu kelas. Seorang pengendara mobil tidak perlu mengetahui cara kerja mesin mobil, cara pembakaran bahan bakar, cara mengganti persneling, dll. Yang paling utama adalah user tahu mana setir untuk mengarahkan mobil, pedal mana yang dipakai menambah kecepatan, dan pedal manapula yang diapakai untuk mengurangi kecepatan, dll. Intinya , yang dia tahu bahwa kendaraan berjalan dengan baik.

Contoh Lain:
Dalam kategori televisi, seorang pemirsa program televisi tidak perlu mengetahui bagaimana program televisi itu ditayangkan untuk menampilkan visual visual di televisi atau untuk berpindah acara dari satu acara ke acara lainnya. Anda perlu mengetahui tombol mana yang digunakan untuk menyalakan TV atau tombol mana yang menggerakkan program agar TV berfungsi sebagaimana mestinya.

Dari visualisasi di atas dengan jelas menampilkan bagaimana suatu proses dari metode kelas selalu tersembunyi, sehingga seseorang hanya tahu bahwa kelas tersebut dapat bekerja, bukan apa yang dilakukan oleh kelas tersebut. Hal di atas cukup penting untuk membangun PBO dalam sebuah aplikasi yang kompleks ketika membangun pemrograman. Misalnya, tim pengemabnagan akan membuat aplikasi akuntansi (tentu saja aplaiaksi ini butuh informasi yang tidak sedikit). Programmer pertam menangani masalah struktur data, Programmer ke-dua menangani masalah tampilan, dan Programmer ke-tiga menangani masalah akuntansi, dll. Sebelum PBO dikenal, setiap programmer harus membuat fungsi dengan argumen input dan output. Jadi jika programmer I ingin memproses program programmer II, dia harus memasukkan dan mendapatkan outputnya.

Pewarisan (Inheritance)
Inheritance adalah proses pembuatan kelas baru dengan mewarisi properti dari kelas yang sudah ada serta properti unik dari kelas baru. Dalam hirarki kelas, jika kelas C adalah turunan dari kelas B dan kelas B adalah turunan dari kelas A, maka atribut dan metode dari kelas A otomatis diwarisi oleh kelas C juga. Setiap subclass mewarisi status (variabel) dan perilaku (metode).
Subkelas kemudian dapat menambahkan keadaa dan perilaku baru yang spesifik, dan juga dapat memodifikasi (mengganti) status dan perilaku yang dikelola oleh superkelasnya.

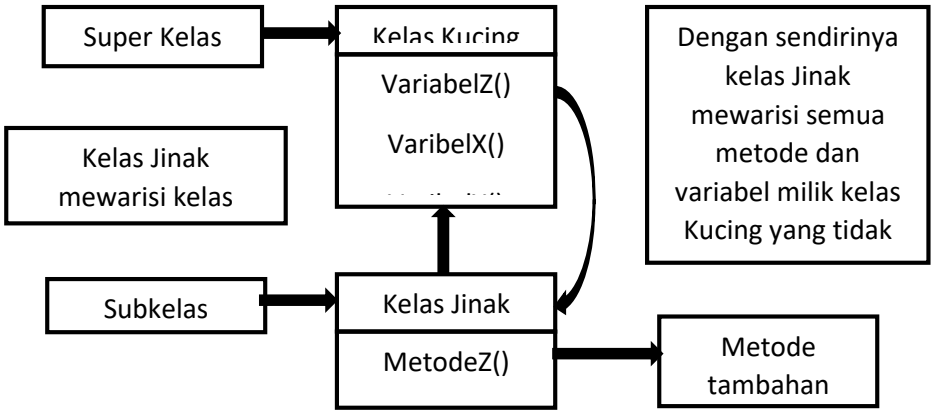
Warisan merupakan bagian penting dari ketiga konsep dasar POO. Konsep pewarisan tersebut mencakup dunia nyata di mana suatu unit/objek dapat memiliki unit/objek turunan. Menggunakan konsep pewarisan, dimungkinkan untuk membuat kelas berdasarkan kelas yang sudah ada sehingga mewarisi semua metode dan variabelnya. Kelas yang mempunyai kelas turunan disebut super class atau kelas dasar. Kelas turunan itu sendiri sering disebut sebagai subclass atau child class.



Child Class dapat mewarisi apa saja dari kelas induk. Karena subclass dapat mewarisi apapun yang dimiliki superclass-nya, kelas anak terdiri dari apa yang dimilikinya serta apa yang diwarisi dari kelas dasar. Jadi, sebuah subclass bisa dikatakan tidak lebih dari perpanjangan kelas induknya dengan menyertakan extend sebagai kata kunci setelah deklarasi nama kelas dan kemudian nama kelas induknya. Kata kunci extens memberi tahu compiler Java bahwa kita ingin memperluas kelas. Biasanya kita hanya perlu menggunakan pewarisan ketika kita menemukan kelas yang dapat diperluas oleh kelas lain. Semua class Java adalah subclass dari superclass yang disebut Object. Selama fase kompilasi, compiler Java membacanya sebagai subclass dari object class.
Berikut adalah tabel kontrol untuk pengaksesan.

Tabel 9.1 Hak Akses

Modifier	Class yg sama	Package yg sama	Sub clas package lain	Class manapun
Private	✓			
Default	✓	✓		
Protected	✓	✓	✓	
Publick	✓	✓	✓	✓



Subclass dan Superclass

Pada java di kenal isitilah pewarisan atau mewarisi atribut dan metode dari suatu kelas ke kelas lainnya. mengelompokkan "konsep warisan" menjadi dua kategori:

- ✓ Kelas Anak (Subclass) – adalah kelas yang mewarisi dari kelas induk
- ✓ Kelas Orang Tua atau Superclass – adalah kelas yang mewarisi atau mewariskan

Gunakan kata kunci extens untuk mewarisi dari kelas induk.

Keuntungan Pewarisan

Kelas anak atau Subclass memberikan status/perilaku spesifik yang membedakannya dari kelas induknya atau superclass. Hal Ini menjadikan pengembang Java lebih suka dan tertarik untuk menggunakan kembali kode sourcer superclass yang ada.

Pemrogram java dapat menjelaskan kelas induk atau uperclass generik khusus, yang dimaksud dari kelas abstrak disini adalah yaitu suatu kelas dengan perilaku dan status generik.

Menjaga kelas dengan data dan metode yang sama memudahkan untuk mengubah data atau metode untuk semua subkelas/subkelas, jadi Anda tidak perlu melakukan perubahan pada setiap subkelas, hanya kelas utama saja. Persyaratan hukum warisan untuk dipertimbangkan:

Bentuk dumum dari deklarasi dalam konsep pewarisan adalah seperti rumus dibawah :

```
[modifier] class namaSubKelas extend namaKelasSuper
{
  // classBody
}
```

Untuk pemahaman secara komplit dapat dilihat pada contoh Program dengan implementasi Enkapsulasi :

```
public class Kapsul{
   private double panjang; // attribute yang disem
bunyikan
   private double lebar;     // attribute yang di
sembunyikan
   private double tinggi;    // attribute yang di
sembunyikan
public Kapsul(){
      double panjang = 0;
      double lebar = 0;
   }
```

Pembuatan kelas kapsul sebagai menu utama dalam pembuatan program ini

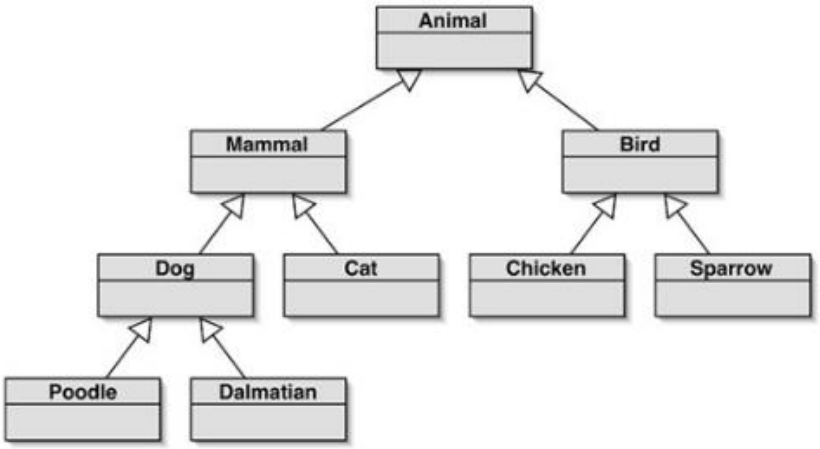
```
private double luas(double p, double l) {   //at
tribute yang di enkapsulasi
      return p*l;
   }
   public void setPanjang(double panjang) {
      this.panjang = panjang;
   }
```

```
  public void setLebar(double lebar) {
    this.lebar = lebar;
  }
  public double getPanjang() {
    return panjang;
  }
  public double getLebar() {
    return lebar;
  }
  public double getLuas() {
    return luas(panjang, lebar);
  }
}

class Encapsulasi{
  public static void main(String[] srgs) {
    Kapsul pp = new Kapsul();
    pp.setPanjang(50);
    pp.setLebar(100);
    System.out.println("Panjang : "+ pp.getPanja
ng());
    System.out.println("Lebar : "+ pp.getLebar()
);
    System.out.println("Luas : "+ pp.getLuas());
  }
}
```

Pada contoh yang telah dihasilkan, Anda dapat melihat bahwa pernyataan pernyataan dari variabel disembunyikan dan ketika perkalian dilakukan untuk menemukan rentangnya, ia dimodulasi. Program di atas membuat 2 kelas, yaitu:
Kelas enkapsulasi dan kelas enkapsulasi, kedua kelas ini berbeda. Dimana kapsul kelasnya adalah kelas untuk membuat dan memanipulasi objek untuk menghitung luas empat persegi panjang. Sedangkan class Encapsulation adalah class untuk menampilkan hasil dari suatu proses.

Polimorfisme Java

Polimorfisme bisa berarti "banyak cara" dan kelas ini terjadi manakala kita mempunyai banyak kelas yang berhubungan satu sama lain melalui konsep dari pewarisan. Warisan memungkinkan kita untuk dapat mewariskan semua atau sebagain atribut dan metode dari kelas lain. Polimorfisme menggunakan cara ini untuk melakukan berbagai tugas. Istilah polimorfisme sering digunakan dalam kaitannya dengan pemrograman berorientasi objek, karena erat kaitannya dengan salah satu pilar seperti kelas, objek, metode, atau pewarisan. Polimorfisme hadir dalam berbagai bentuk atau varian. Dalam bahasa pemrograman, polimorfisme adalah konsep penggunaan antarmuka tunggal untuk entitas yang berbeda. Secara umum, menggunakan satu ikon mewakili beberapa tipe entitas.



Polimorfisme bisa di difinisikan memiliki banyak jenis. Dua atau lebih objek dikatakan polimorfis ketika kedua objek tersebut memiliki antarmuka yang identik tetapi berperilaku beda. Pada pemrograman, polimorfisme dapat diartikan sebagai modulasi dengan nama yang sama tetapi perilakunya beda, sehingga daftar kode implementasinya juga berbeda. Sebagai contoh, pertimbangkan superclass bernama Animal yang memiliki metode bernama bestSound(). Subkategori hewan dapat berupa kucing, burung - dan mereka juga memiliki aplikasi suara binatang sendiri (riasan kucing, kicau burung, dll.)

Kondisi yang harus dipenuhi untuk mengimplementasikan polimorfisme adalah:

- ✓ Metode yang digunakan wajib melewati variabel dari kelas superclass.
- ✓ Metode yang digunakan haruslah dari metode kelas dasar.
- ✓ Signature metode harus sama di kelas induk dan anak.
- ✓ Atribut akses dari metode subclass tidak dapat dibatasi lebih dari kelas dasar.

Tipe polimorphie

Macam atau bentuk dari polimorfisme diantaranya adalah polimorfisme satatis dan polimorphis dinamis . Beda antara keduanya adalah bagaimana polimorfisme dilakukan. Polimorfisme statis menggunakan kelebihan metode sedangkan polimorfisme dinamis menggunakan kelebihan metode diferensial

- ✓ Method overloading terjadi pada class yang mempunyai i nama method yang sama tetapi parameter dan tipe datanya berbeda.
- ✓ Kemampuan untuk meng-override method sama, tetapi behavior atau perilakunya berbeda antara superclass dan subclass ini.

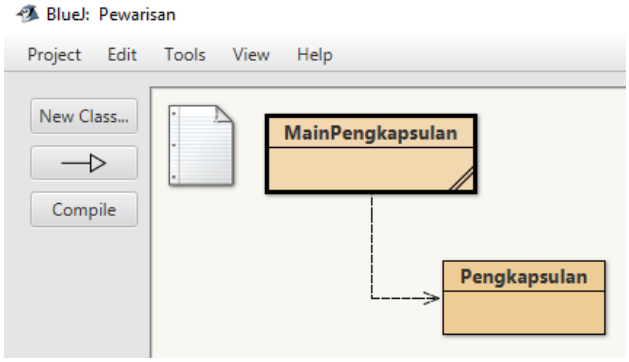
Keuntungan Polimorfik

- ✓ Membantu pengembang menggunakan kembali kode dan kelas setelah ditulis, diuji, dan diimplementasikan
- ✓ Nama variabel dapat digunakan untuk menyimpan variabel dari beberapa tipe data seperti float, double, long, integer dll.
- ✓ Membantu membangun abstraksi yang kuat dan kompleks dari yang lebih sederhana

Praktikum Program Konsep PBO

Praktikum Pengkapsulan

Buatlah class pada folder pewarisan dengan nama Class pengkapusan dan class mainpengkapsulan jadi yang dibuat di sini adalah dua buah kelas.



- ✓ **Class Pengkapsulan**

  Buatlah kelas yang pertama yaitu class pengkapsulan, dan selanjutnya tuliskan kode programnya seperti pada contoh di bawah ini. Pada kode ini yang pertama adalah pendeklarasian variabel panjang dan lebar untuk menghitung luas segi empat.

```
public class Pengkapsulan {
        private double pjg;
        private double lbr;
        public Pengkapsulan(){
            pjg = 20;
            lbr = 10;}
        private double luas(double p, double l){
            return p*l;}
        public void setpjg(double pjg){
            this.pjg = pjg;}
        public void setlbr(double lbr){
            this.lbr = lbr;}
        public double getpjg(){
            return pjg;}
        public double getlbr(){
            return lbr;}
        public double getluas(){
            return luas (pjg, lbr);}
}
```

  Pembuatan kelas pengkapsulan untuk menghitung luas segi empat dimana pada kelas pengkapsulan ini terdiri dari variabel pjg = 20 dan lbr = 10, selanjutnya kita akan cari luas segi empat tersebut

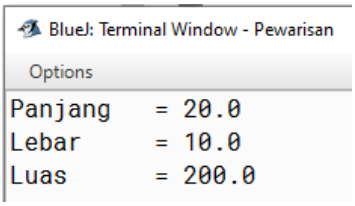
✓ **Class MainPengkapsulan**

Buatlah kelas yang kedua dengan nama class yaitu mainpengkapsulan, dan selanjutnya tuliskan kode programnya seperti pada contoh di bawah ini. Pada kode ini yang pertama adalah Menampilkan panjang dan lebar serta luas empat persegi panjang.

```
import java.util.Scanner;
public class MainPengkapsulan
{
    public static void main(String[]args){
        Pengkapsulan pp = new Pengkapsulan();
        pp.setpjg(20);
        pp.setlbr(10);
        System.out.println("Panjang   = " + pp.getpjg());
        System.out.println("Lebar     = " + pp.getlbr());
        System.out.println("Luas      = " + pp.getluas());
    }
}
```

Jika pada kelas mainpengkapsulan sudah selesai, selanjutnya adalah dengan cek program tersebut apakah ada kesalahan dalam penulisan programnya dengan cara menekan tomobl compile, ulangi demikian terus menerus sampai muncul pesan no syntax error yang berati sudah tidak ada kesalahan kode program selanjutnya jalan program tersebut dengan klik kanan pada class mainpengkapsulan dan vilih void dan selanjutnya lihat hasilnya. Apakah sudah seperti contoh dibawah ini. Jika sudah seperti ini berarti sudah betul.

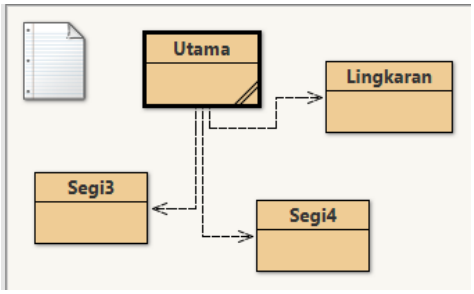
✓ **Hasil Program**



Hasil compile dari program pengkapsulan dan main pengkapsulan dapat Anda lihat sperti pada gambar diatas

Praktikum Pengkapsulan-2

Menghitung Luas dari Segi empat, luas segitiga, dan juga luas persegi panjang. Selanjutnya total luas bangun ini dikurangi dengan luas setengah lingkaran, sehingga kita akan dapatkan wilayah yang diarsir.



✓ **Class Utama**

Buat class dengan nama utama dan selanjutnya anda bisa memberikan potongan kode programnya tersebut seperti pada contoh dibawah ini. Untuk mensetting segi tiga dan segi empat.

```
public class Utama
{
public static void main(String[] args) {
        System.out.print("\u000c");
        // hitung luas segitiga
        Segi3 st1 = new Segi3();
        st1.setAlas(8);
        st1.setTinggi(7);
        double luasADE = st1.getLuas();

        // hitung luas segitiga
        Segi3 st2 = new Segi3();
        st2.setAlas(8);
        st2.setTinggi(7);
        double luasCBF = st2.getLuas();
        // hitung luas persegipanjang
        Segi4 pp1 = new Segi4();
        pp1.setPanjang(14);
        pp1.setLebar(7);
        double luasCDEF = pp1.getLuas();

        // hitung luas setengah lingkaran X
        Lingkaran l1 = new Lingkaran();
        l1.setJejari(7);
        double luasX = 0.5 * l1.getLuas();

        // hitung luas daerah diarsir
        double luasArsir = luasADE + luasCBF + luasCDEF - luasX;
        System.out.println("Luas daerah diarsir: " + luasArsir + " cm2");
    }
}
```

✓ **Class Segi3**

Selanjutnya buat kelas dengan nama Class Segi3 , dimana kelas ini nantinya untuk menghitung luas segi tiga yang didapat dari alas dikalikan tinggi dibagi dengna 2 atau alas dikalikan tinggi dikalikan dengan 0.5

```
public class Segi3
{
 // atribut
    private int alas;
    private int tinggi;
    private double luas;

    public void setAlas(int a){
        if (a > 0){
            this.alas = a;
        } else {
            this.alas = 0;}
    }
    public void setTinggi(int t){
        if (t > 0){
            this.tinggi = t;
        } else {
            this.tinggi = 0;}
    }
    public double getLuas(){
        // hitung luasnya
        this.luas = this.alas * this.tinggi * 0.5;
        return this.luas;}
}
```

✓ **Clas Segi4**

Lanjutkan dengan membuat kelas dengan nama Class Segi4 , dimana kelas ini nantinya untuk menghitung luas segi empat yang didapat dari panjang dikalikan lebar.

```
public class Segi4
{
 private int panjang;
    private int lebar;
    private double luas;

    public void setPanjang(int p){
        if (p > 0){
            this.panjang = p;
        } else {
            this.panjang = 0;}
    }

    public void setLebar(int l){
        if (l > 0){
            this.lebar = l;
        } else {
            this.lebar = 0;}
    }

    public double getLuas(){
        // hitung luasnya
        this.luas = this.panjang * this.lebar;
        return this.luas;}
}
```

✓ **Clas LIngkaran**

Lanjutkan dengan membuat kelas yang lain yaitu Class lingkaran , dimana kelas ini nantinya untuk menghitung luas lingkaran yang didapat dari PHI (3.14) dikalikan Jari-Jari dikalikan jari-jari.

```
public class Lingkaran
{
    private int jejari;
    private double luas;

        public void setJejari(int r){
        if (r > 0){
            this.jejari = r;
        } else {
            this.jejari = 0;
        }
    }

        public double getLuas(){
        // hitung luasnya
        this.luas = Math.PI * Math.pow(this.jejari, 2);
        return this.luas;
    }
}
```

✓ **Hasil Program**

```
BlueJ: Terminal Window - Pewarisan2
Options
Luas daerah diarsir: 77.03097998705007 cm2
```

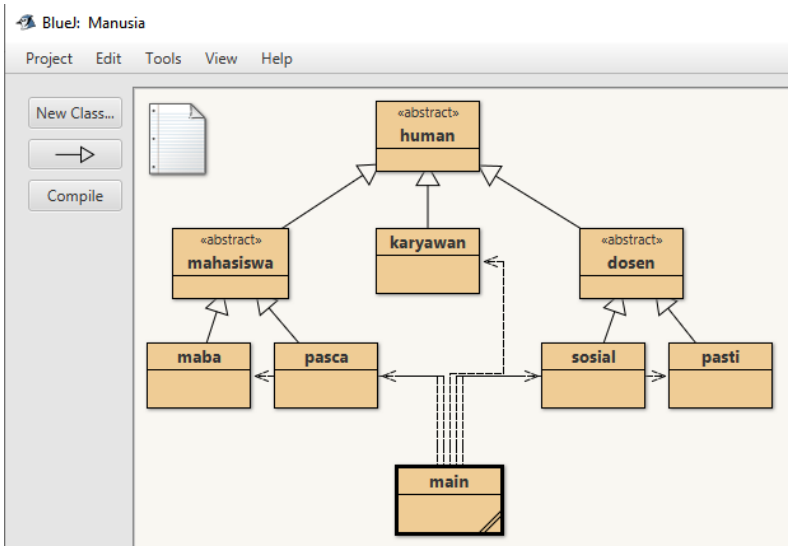


Diagram di atas adalah kelas program warisan. Pada diagram di atas, garis putus-putus merepresentasikan objek instantiation dan garis padat merepresentasikan pewarisan, meskipun masing-masing kotak merepresentasikan kelas. Jadi jika melihat tabel di atas, kategorinya antara lain: manusia, mahasiswa, pekerja, fakultas, mahasiswa baru, mahasiswa pascasarjana, sosial, percaya diri dan menyenangkan.

Class human

```
public abstract class human
{
   public human(){
   };

   void work(){
   };

   public abstract void hobi();
   public abstract void alamat();
}
```

Kelas human menggunakan jenis kelas abract yaitu suatu kelas yang tidak memiliki obyek didalamanya. Pada kelas tersebut terdapat terdapat 4 method yaitu methode alamat, method hobi, method human, Method Mahasiswa.

Kelas mahasiswa

```
public abstract class mahasiswa extends human
{
   public mahasiswa(){
   };

   public void hobi(){
   };

   public void alamat(){
   };
}
```

Kelas mahasiswa juga sama dengan kelas human menggunakan jenis kelas abstrak dan kelas mahasiswa menurunkan sifat method serta atribut dari human. Pada kelas mahasiswa tersebut terdapat 3 method yaitu method hobi, method mahasiswa dan method alamat.

Class karyawan

```
public class karyawan extends human
{
   public karyawan(){
   };

   public void alamat(){
    System.out.println("Semarang");
   };

   public void work(){
      System.out.println("Administrasi");
   };

   public void hobi(){
      System.out.println("Baca");
   };
}
```

Kelas karyawan menurunkan sifat dari kelas human serta terdapat 4(empat) method yaitu method karyawan, methode alamat, mthode work, dan method hobi. Dimana masing masing memiliki nilai keluaran terkecuali pada method karyawan.

Class dosen

```
public abstract class dosen extends human
{
  public dosen(){
   };

  public void hobi(){
   };

   public void alamat(){
   };
}
```

Kelas dosen adalah juga sama dengan kelas sebelumnya yaitu sebagai kelas abstrak juga. Pada kelas dosen ini terdapat 3 method yaitu methode dosen, methode hobi dan method alamat.

Class maba

```
public class maba extends mahasiswa
{
   public maba(){
   }

   public void alamat(){
   System.out.println("Demak");
   }

   public void hobi(){
   System.out.println("Jalan-2");
   }
}
```

Kelas mahasiswa baru atau maba menurunkan sifat dari kelas mahasiswa dan terdapat tiga method yaitu maba, alamat, dan hobi.

Class pasca

```
public class pasca extends mahasiswa
{
   public void namajurusan(){
      System.out.println("Teknik Informatika");
   }

   public pasca(){
   }

   public void alamat(){
      System.out.println("Kudus");
   }

   public void hobi(){
      System.out.println("Makan-makan");
   }
}
```

Kelas pasca menurunkan sifat dari kelas mahasiswa dan pada kelas pasca terebut terdapat 3 method yaitu methode nama jurusan, method alamat, dan method hobi yang diturunkan dari class mahasiswa. Dimana pada masing masing method tersebut mempunyai nilai kelauran kecuali method pasca

Class sosial

```
public class sosial extends dosen
{
   public sosial(){
   }

   public void hobi(){
      System.out.println("Nongkrong");
   }

   public void alamat(){
      System.out.println("Kudus");
   }
}
```

Kelas sosial menurunkan juga sifat dari kelas dosen dan pada kelas tersbut ada 3 (tiga) method yang punyai kelas sosial yaitu methode sosial, method hobi dan method alamat. Dimana pada masing masing method tersebut mempunyai nilai kelauran kecuali pada method sosial.

Class Pasti

```
public class pasti extends dosen
{
  public void namajurusan(){
     System.out.println("FSA:");
  }

  public pasti(){
  }

  public void hobi(){
     System.out.println("Makan");
  }

  public void alamat(){
     System.out.println("Solo");
  }
}
```

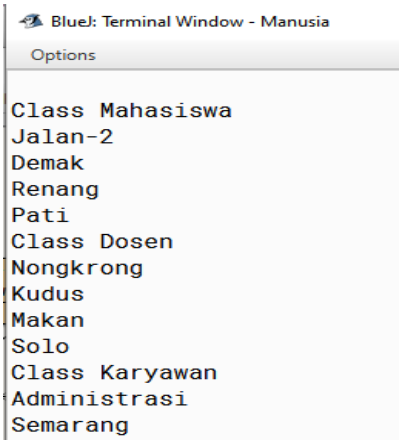
Kelas pasti menurunkan sifat dari kelas dosen dan pada kelas tersebut mempunyai 4 (empat) method yaitu method namajurusan, methode pasti, methid hobi, dan method alamat dimana pada maisng masing method tersebut mempunyai nilai output kecuali method pada metjod pasti.

Class mainclass

```
public class mainclass
{
   public static void main(String [] args){
      maba maba = new maba();
      pasca pasca = new pasca();
      karyawan kr = new karyawan();
      pasti sc = new pasti();
      sosial soc = new sosial();
      System.out.println("Class Mahasiswa ");
      maba.hobi();
      maba.alamat();
      pasca.hobi();
      pasca.alamat();
      //pasca.namajurusan();
      System.out.println("Class Dosen");
      sc.hobi();
      sc.alamat();
      //sc.namajurusan();
      soc.hobi();
      soc.alamat();
      System.out.println("Class Karyawan");
      kr.work();
      kr.alamat();
   }
}
```

Pada Kelas utama ini terdapat method main yang dapat jalankan dan didalamnya terdapat pembuatan obyek atau kelas baru seperti yang kita buat diatas antara laian : maba, pasca, karyawan, pasti, sosial. Setelah pembuatan kelas dari masing masing method tersbut selanjutnya dipanggil dan divisualisasikan pada layar.

Hasil Program diatas adalah



Praktikum Pewarisan-2

Pada program berikut adalah sebuah kelas pewarisan dari kelas utama yang bernama Mobil. Sebagai superkelas dan akan mewariskan ke kelas turunan atau subkelas yaitu Ferrari dan Toyota, karena Ferrari dan toyota adalah turunan dari mobil pasti ferarari dan toyota mearisi merk, warna , jumlah pintu dan juga jenis. Untuk sub kelas seperti ferrari dan toyota punya atribut sendiri yang tidak ada pada kelas utama seprti Bahan bakar, kategori, dll

Class Mobil

Buatlah sebuah class mobil sesuai dengan struktur gambar diatas , dimana pada kelas ferrari dan kelas toyota adalah kelas turunan dari kelas mobil. Pada kelas ini terdiri dari merek , warna, jumlah pintu dan jenis mobil.

```
public class Mobil
{
    private String merek;
    private String warna;
    private int jumlahpintu;
    private String jenis;
    public String getJenis() {
        return jenis;
    }
    public void setJenis(String jenis) {
        this.jenis = jenis;
    }
    public int getJumlahpintu() {
        return jumlahpintu;
    }
    public void setJumlahpintu(int jumlahpintu) {
        this.jumlahpintu = jumlahpintu;
    }
    public String getMerek() {
        return merek;
    }
    public void setMerek(String merek) {
        this.merek = merek;
    }
```

```
    public String getWarna() {
        return warna;
    }
    public void setWarna(String warna) {
        this.warna = warna;
    }
 public void tampilkandata(){
        System.out.println("Merek Mobil  :"+getMerek());
        System.out.println("Warna Mobil  :"+getWarna());
        System.out.println("Jumlah Pintu :"+getJumlahpintu());
        System.out.println("Jenis Mobil  :"+getJenis());

 }
 public void inputData(String m,String w,String j,int jp){
     setMerek(m);
     setWarna(w);
     setJenis(j);
     setJumlahpintu(jp);

 }
}
```

Class MainMobil

Buat juga kelas utama yang akan digunakan untuk menampilkan dari kelas turunan dimana pada kelas mainutama ini beriskan data nama mobil ferrari dan toyota.

```
public class MainMobil
{
    public static void main (String [] args){
    System.out.println("FERARI");

    Ferari f = new Ferari();
    f.tampilkan();

    System.out.println("\nTOYOTA");

    Toyota t = new Toyota();
    t.tampilkan();
  }
}
```

Class Ferrari Turunan Mobil

Selanjutnya buatlah sebuah kelas dengan nama class ferrari turunan dari kelas mobil, dimana pada kelas ferrari ini selain mewarisi dari kelas mobil seperti merek , warna, jumlah pintu dan jenis mobil, kelas ferrari sendiri memiliki anggota seperti silider, jenis bahan bakar dan jenis kategori.

```
public class Ferari extends Mobil
{
     public void tampilkan(){
     double besarsilinder=12;
     String bahanbakar="Pertamax";
     String kategori="Sport";

     Mobil m = new Mobil();

     m.inputData("Ferrari", "Merah", "Mewah", 2);
     m.tampilkandata();

     System.out.println("silinder :"+besarsilinder);
     System.out.println("jenis bahan bakar :"+bahanbakar);
     System.out.println("jenis kategori :"+kategori);
     }
}
```

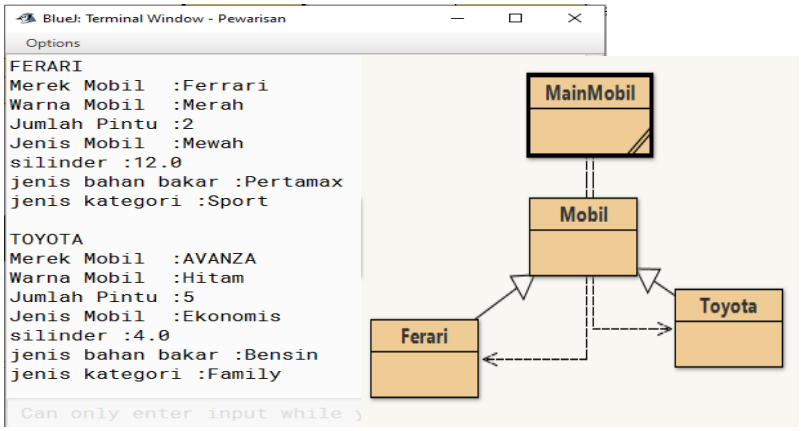
Class Toyota Turunan Mobil

Lanjutkan dengan membuat kelas dengan nama class toyoya turunan dari kelas mobil, dimana pada kelas toyota ini selain mewarisi dari kelas mobil seperti merek , warna, jumlah pintu dan jenis mobil, kelas toyota sendiri memiliki anggota seperti silider, jenis bahan bakar dan jenis kategori.

```
public class Toyota extends Mobil
{
    public void tampilkan(){
        double besarsilinder;
        String bahanbakar;
        String kategori;

        besarsilinder=4;
        bahanbakar="Bensin";
        kategori="Family";

        Mobil m = new Mobil();

        m.inputData("AVANZA", "Hitam", "Ekonomis", 5);
        m.tampilkandata();

        System.out.println("silinder :"+besarsilinder);
        System.out.println("jenis bahan bakar :"+bahanbakar);
        System.out.println("jenis kategori :"+kategori);
    }
}
```

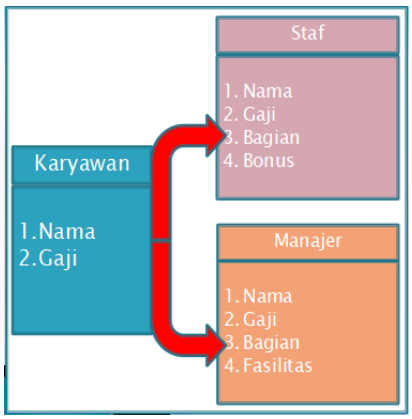
Hasil dari Program Class Mobil

Hasil dari pembuatan kelas ferrari dan toyota sebagai turunan dari kelas mobil dapat dilihat sperti contoh tampilan di bawah ini.



Praktikum Pewarisan-3

Pada program berikut adalah sebuah kelas pewarisan dari kelas utama yang bernama Karyawan dimana pada kelas karyawan mewariskan nama dan gaji pada sub kelas staf dan manajer. Yang namanya manager atau staff pasti punya nama dan gaji yang diterima. Tetapi pada sub kelas staf dan manager tentu memiliki atribut sendiri yang berbeda diantara keduanya, pada sub kelas staf punya bagian dan banus, sedangkan pada bagian manager punya atribut bagian dan fasilitas.



✓ **Class Karyawan**

Buatlah sebuah class karyawan sesuai dengan struktur gambar diatas , dimana pada kelas karyawan memliki anggota Nama dan Gaji. Pada kelas karyawan juga memiliki kelas turunan yaitu staf dan manager. Untuk lebih lengkapnya mengenai kode program kelas karyawan dapat dilihat pada gambar dibawah ini.

```
public class Karyawan
{
    private String nama;
    private double gaji;
    protected Karyawan(String n , double g){
        nama = n;
        gaji = g;
    }
    protected String getDetails(){
        return "Nama Karyawan   =" + nama + "\nGaji    = " + gaji;
    }
    public void cetak(){
        System.out.println("Coba di Karyawan....");
    }
}
```

✓ **Class Manajer Turunan dari Class Karyawan**

Selanjutnya Buatlah sebuah class dengan nama Manager turanan dari kelas karyawan. Kelas turunan mewarisi kelas supernya yaitu karyawan, jadi manager pasti punya nama dan gaji dan juga angota dari dirinya sendiri yaitu bagian dan fasilitas.

```
public class Manajer extends Karyawan{
        private String bagian;
        public Manajer(String nama, double gaji, String bag){
            super(nama, gaji);
            bagian = bag;
        }
        public String getBagian(){
            return bagian;
        }
        public String getDetails(){
            return super.getDetails() + "\nBagian : " + getBagian();
        }
        public void cetak(){
            System.out.println("Coba di Manajer...");
        }
    }
```

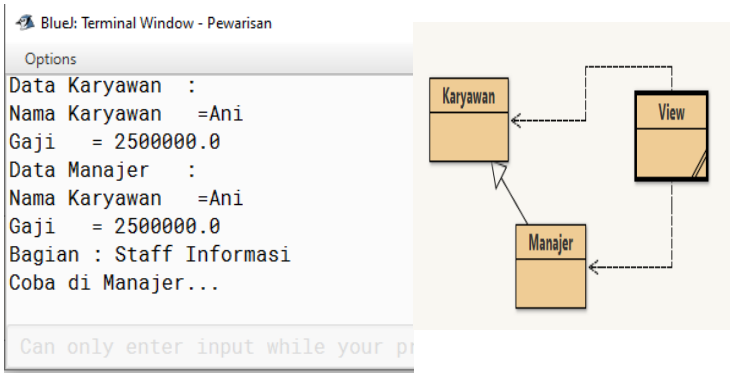
✓ **Class View atau Kelas Cobakaryawan**

Untuk menguji hasil program tersebut maka selanjutnya Anda bisa buat kelas Cobakaryawan untuk mengujinya. Untuk contoh lengakpnya bisa di lihat pada gambar berikut ini. Yang kita kasih nama kelas view

```
public class View
{
    public static void main(String[]args){
        Karyawan Kar = new Karyawan("Ani", 2500000);
        Karyawan Kar1 = new Manajer("Ani", 2500000,"Staff Informasi");
        System.out.println("Data Karyawan  :\n" + Kar.getDetails());
        System.out.println("Data Manajer   :\n" + Kar1.getDetails());
        Kar1.cetak();
    }
}
```

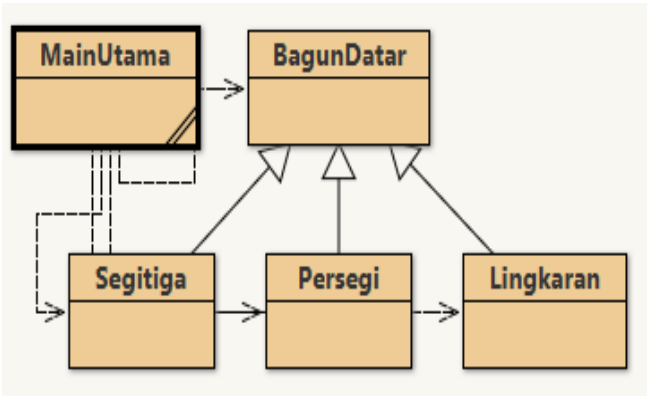
✓ **Hasil dari program diatas ketika dijalankan adalah sebagai berikut**

Setelah semua kelas dan program tersebut Anda buat dan sudah juga di compile dengan hasil non syntac error maka hasilnya akan terlihat seperti contoh dibawah ini.



Praktikum Polymorphis

Untuk lebih jelasnya mengenai program polimorfism dinamis dapat kita lihat pada contoh di bawah ini. Graf akhir datar dengan tiga subkelas, yaitu: persegi, lingkaran dan segitiga. Setiap kelas memiliki metode yang sama yaitu luas dan keliling, namun metode tersebut memiliki isi rumus yang berbeda.



✓ **Kelas BagunDatar**

Contoh program berikut adalah konsep dasar dari pemrograman berorientasi objek pada polimorphis. Dimana kita akan buat bagun datar yang terdiri dari segitiga, persegi, lingkaran dan mainprogram.

Selanjutnya kita buat class bangun datar pada kelas ini digunakan untuk menampilkan luas dan keliling dari bagun datar yang terdiri dari segitiga, persegi dan lingkaran.

```
public class BagunDatar
{
    float luas(){
        System.out.println("Menghitung luas bangun datar");
        return 0;
    }

    float keliling(){
        System.out.println("Menghitung keliling bangun datar");
        return 0;
    }
}
```

✓ **Kelas Persegi**

Berikutnya buat kelas persegi dimana pada kelas persegi ini turunan dari kelas bangun datar. pada turunan kelas bangun datar pasti punya luas dan keliling begitu juga pada kelas persegi nantinya akan menampilkan sisi-sisinya dan menampilkan luas persegi dan keliling persegi. Untuk lebih lengkap mengenai kode program dapat dilihat pada gambar dibawah ini.

```
public class Persegi extends BagunDatar
{
    int sisi;

    public Persegi(int sisi) {
        this.sisi = sisi;
    }

    @Override
    public float luas() {
        return this.sisi * this.sisi;
    }

    @Override
    public float keliling(){
        return this.sisi * 4;
    }
}
```

✓ **Kelas Segitiga**

Langkah selanjutnya adalah buat kelas segitiga dimana pada kelas segitiga ini turunan dari kelas bangun datar. pada turunan kelas bangun datar pasti punya luas dan keliling begitu juga pada kelas segi nantinya akan menampilkan alas dan tinggi serta dan menampilkan luas persegi dan keliling persegi. Untuk lebih lengkap mengenai kode program dapat dilihat pada gambar dibawah ini.

```
public class Segitiga extends BagunDatar
{
    int alas;
    int tinggi;

    public Segitiga(int alas, int tinggi) {
        this.alas = alas;
        this.tinggi = tinggi;
    }

    @Override
    public float luas(){
        return this.alas * this.tinggi;
    }
}
```

✓ **Kelas Lingkaran**

Langkah selanjutnya adalah buat kelas lingkaran dimana pada kelas lingkaran ini juga turunan dari kelas bangun datar. pada turunan kelas bangun datar pasti punya luas dan keliling begitu juga pada kelas lingkaran nantinya akan menampilkan jari jari serta dan menampilkan luas persegi dan keliling persegi. Untuk lebih lengkap mengenai kode program dapat dilihat pada gambar dibawah ini.

```
public class Lingkaran extends BagunDatar
{
    int r;

    public Lingkaran(int r) {
        this.r = r;
    }

    @Override
    public float luas(){
        return (float) (Math.PI * r * r);
    }

    @Override
    public float keliling(){
        return (float) (2 * Math.PI * r);
    }
}
```

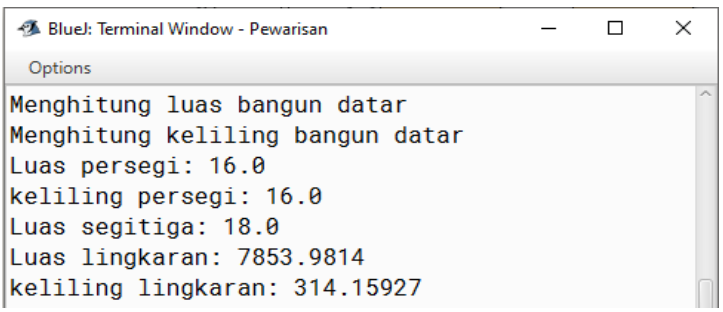
✓ **Kelas Main Utama**

Langkah selanjutnya adalah buat kelas utama yang kita kasih nama MainUtama dimana pada main utama ini yang akan digunakan untuk menguji program yang kita kerjakan sebelumnya. Dimana pada kelas ini akan menampilkan luas persegi , keliling persegi, luas segitiga , keliling segitiga, luas lingkaran dan keliling lingkaran. Untuk lebih lengkap mengenai kode program dapat dilihat pada gambar dibawah ini.

```
public class MainUtama
{
    public static void main(String[] args) {

        BagunDatar bd = new BagunDatar();
        Persegi persegi = new Persegi(4);
        Segitiga segitiga = new Segitiga(6, 3);
        Lingkaran lingkaran = new Lingkaran(50);

        // memanggil method luas dan keliling
        bd.luas();
        bd.keliling();

        System.out.println("Luas persegi: " + persegi.luas());
        System.out.println("keliling persegi: " + persegi.keliling());
        System.out.println("Luas segitiga: " + segitiga.luas());
        System.out.println("Luas lingkaran: " + lingkaran.luas());
        System.out.println("keliling lingkaran: " + lingkaran.keliling());
    }
}
```

✓ **Hasil Program**

Ketika semua kelas pada bangun datar ini selesai dibuat dan sudah di compile dan tidak ada kesalahan maka hasilnya dapat dilihat seperti contoh berikut ini.



BAB X
ABSTRACT CLASS dan INTERFACE

Class Abstract

Kelas abstrak adalah merupakan sutau kelas berisi satu atau lebih suatu metode abstrak. Kelas abstrak ini hanya digunakan untuk membuat metode tanpa implementasi langsung.

Kelas abstrak adalah kelas setengah jadi (abstrak) yang berisi metode dan atribut. Kelas abstrak sebenarnya adalah kelas, sehingga memiliki semua properti dari kelas reguler (memiliki konstruktor). Ini hanya abstrak sekarang karena itulah mengapa metode ini biasanya kosong/belum diimplementasikan. Tetapi kelas abstrak dapat mengimplementasikan metode. Kelas abstrak selalu merupakan superclass/hierarki tertinggi dari subclass-nya.

ClassName kelas abstrak tidak dapat dilemparkan ke objek dengan perintah baru. Itu hanya kelas kerangka dan perlu dipindahkan ke subkelas untuk bekerja. Ketika kelas abstrak diperluas, subkelas harus mengganti beberapa metode kosong dari kelas abstrak, jika tidak, subkelas menjadi kelas abstrak.

Tujuannya adalah menyembunyikan implementasi atau koding sebenarnya dari method-method yang ada sehingga orang lain dapat mengetahui kegunaannya tanpa harus tahu detail bagaimana dia berjalan. Setiap Class anak harus mengimplementasikan method-method yang ada di class induk yang abstract karena hanya berupa nama method tanpa isi kode. Namun tidak semua method di abstract class harus abstract, karena kita bisa memasukkan method yang normal di abstract class.

Untuk memudahkan pemahaman, dibuat dua class, misalnya class component dan class make_food, class component diubah menjadi class abstract, dan class make_food adalah class utama dengan method main. , maka metode kelas komponen (abstraksi) diturunkan di kelas make_food.

Studi kasus lainnya adalah:
Saat desainer pakaian mendesain kemeja lengan panjang, desainer dapat membuat sketsa untuk kemeja tersebut, Berikut adalah format penulisan class abstract, yaitu :

<modifier> abstract class <NamaClass>

Contoh :

```
public abstract class BangunDatar{}
abstract class MahklukHidup{}
```

Berikut adalah format penulisan method abstract, yaitu :

*<modifier> abstract <returnTipe> <namaMethod> (<*parameter>);*

Contoh :

```
public abstract double luas();
public abstract void keliling(double x, double y);
```

Contoh pembuatan abstract class adalah sebagai berikut:

```
abstract class OperasiBilangan {
protected double a;
protected double b;
protected abstract void setA(double a);
protected abstract void setB(double b);
// method biasa (tidak abstract)
protected void setAB(double a, double b) {
this.a = a;
    this.b = b;
}
protected abstract double hitung();
}
```

Apa itu Interface

Antarmuka pengguna konsep PBO adalah modul yang berisi kumpulan deskripsi metode. Dalam hal ini, metode bersifat abstrak atau tidak memiliki badan. Antarmuka pengguna adalah bentuk seperti kelas yang terdiri dari sekumpulan metode dan konstanta kosong. Antarmuka tidak dapat dibuat menjadi objek, hanya diimplementasikan. Biasanya, ketika sebuah kelas mendeklarasikan objek, antarmuka mendeklarasikan properti seperti dapat dibaca, dapat dieksekusi, dapat dibandingkan, dll.

Catatan:
Seperti yang digunakan dalam pemrograman itu, antarmuka pengguna memiliki arti yang berbeda dari "antarmuka pengguna (UI)", di mana antarmuka pengguna adalah antarmuka pengguna yang dibuat untuk memungkinkan pengguna berinteraksi dengan program komputer.

Keberadaan interface dalam sebuah kode program sebenarnya sifatnya hanyalah opsional saja. Namun, meskipun demikian interface ini memiliki kegunaan yaitu bisa berfungsi sebagai semacam validator keberadaan method yang dimiliki oleh sebuah class. Oleh karena itu, apabila sebuah class mengimplementasikan sebuah interface, maka akan terjadi semacam kontrak bahwa class tersebut harus mengoverriding dan menjabarkan detil semua method yang ada di dalam interface tersebut. Apabila tidak, maka class tersebut belum sempurna (error) atau dijadikan class abstrak.

- ✓ Interface tidak diawali dengan kata kunci **class** dan dia merupakan semua kontrak yang harus diimplementasikan oleh semua class yang mengimplementasikannya.
- ✓ Sama dengan method abstract di abstract class, maka method di interface juga hanya berupa nama tanpa implementasi.
- ✓ Dengan interface kita bisa "memaksakan" method-method apa saja yang harus tersedia dalam sebuah class ketika dia mengimplementasikan interface.

Karakteristik Interface

Secara struktur, bentuk interface mirip dengan class, yaitu bisa memiliki atribut dan method. Namun perbedaannya adalah bahwa atribut yang ada di interface haruslah bersifat sebagai konstanta (sudah diassign dengan sebuah nilai), tidak boleh dalam bentuk deklarasi atribut saja. Selain itu, perbedaan lainnya adalah bahwa method yang ada dalam interface haruslah semuanya bersifat abstrak.

Antarmuka memiliki konsep dan operasi yang hampir sama dengan kelas Abstrak, meskipun fungsi keduanya sama, ada beberapa perbedaan yang perlu Anda ketahui.

Beda Kelas Abstrak dan Antarmuka

Perbedaan antara kelas abstrak dan antarmuka misalnya.

- ✓ Kelas abstrak menggunakan ekstensi, sedangkan antarmuka menggunakan metode implementasi
- ✓ Suatu kelas dapat mengimplementasikan lebih dari satu antarmuka, tetapi tidak dapat memperluas lebih dari satu kelas abstrak.

Untuk lebih memahami perbedaan lebih lanjut berikut ini disajikan dalam bentuk tabel supaya mudah dipahami.

Tabel 10.1 Perbedaan atanra abstrak kelas dan antarmuka

Abstract	Interface
Dapat berisi metode abstrak dan non-abstrak.	Antarmuka hanya dapat berisi metode abstrak
Abstrak wajib Kita tulis sendiri modifiernya.	Antarmuka tidak perlu menempatkan abstrak publik sebelum nama metode. Karena pengubah metode pada antarmuka secara implisit bersifat publik dan abstrak.
Dapat mendeklarasikan konstanta dan variabel instan	Hanya dapat melaporkan secara default. Variabel yang dideklarasikan secara implisit dalam antarmuka bersifat publik, statis, dan final
Methode bisa statis.	Metode tidak boleh statis
Metode bisa definitif.	Metode tidak bisa pasti.
Kelas abstrak hanya dapat memperluas kelas abstrak dan mengimplementasikan banyak antarmuka	Antarmuka hanya dapat memperluas antarmuka lain. Dan Anda tidak dapat mengimplementasikan kelas atau antarmuka lainnya
Abstract class hanya dapat melakukan atau meneruskan satu abstract class dan menerapakannya dari beberapa antar muka	Antarmuka hanya dapat meneruskan antarmuka lainya. Antar muka tidak bisa meneruskan class atau interface lainnya.

Tidak seperti kelas abstrak, komponen antarmuka memiliki aturan di mana kita tidak perlu menambahkan properti seperti publik, abstrak, dan final ke variabel atau metode ini karena variabel atau metode ini secara implisit sudah bersifat publik, abstrak, dan final.

Antarmuka adalah mekanisme di Java yang memungkinkan Anda berbagi konstanta atau menentukan format metode yang dapat digunakan oleh banyak kelas.

Kelas dapat mengimplementasikan lebih dari satu antarmuka pengguna

Deklarasi Interface :

Deklarasi dan penggunaan interface:

```
Public interface Nama Interface
{
// beberapa methode
}
Penggunaan interface
```

- ✓ **Contoh Interface**

```
Public Interface MaklukHidup
{
}
```

Contoh deklarasi interface

```
Public class namaclas imlements Nama Interface
{
// beberapa methode
//overide dari beberapa method
}
```

- ✓ **Contoh Interface**

```
Class Hewan Implement MaklukHidup
{
}
```

Jika kelas Anda mencoba menerapkan kelas antarmuka, mohon dipastikan Anda telah menerapkan semua metode antarmuka, jika tidak, Anda akan mendapatkan kesalahan.

Hubungan Antarmuka ke Kelas

Kelas dapat mengimplementasikan antarmuka selama kode penerpannya tersedia untuk semua metode yang ditentukan dalam antarmuka.

Hal yang perlu diperhatikan tentang hubungan antara interface dan class adalah bahwa sebuah class hanya bisa melanjutkan satu superclass, tetapi dapat menerapkannya ke banyak antarmuka..

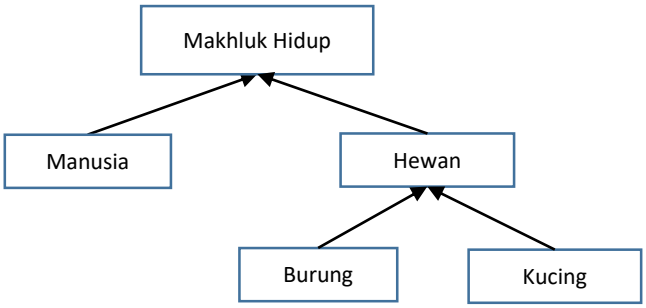
Pewarsian Antar Interface

Antarmuka bukan bagian dari hierarki kelas. Namun, antarmuka dapat memiliki hubungan yang saling diwariskan. Misalkan kita memiliki dua antarmuka pengguna StudentInterface dan PersonInterface. Saat StudentInterface memperluas PersonInterface, ia mewarisi semua definisi metode dari PersonInterface.

- ✓ **Contoh program**

  Kami membuat superclass yang disebut entitas. Kelas ini memiliki metode khusus seperti bernafas, makan, tidur dan berjalan, namun superclass memiliki beberapa metode yang tidak dapat digeneralisasikan. Mari kita ambil contoh, metode berjalan. Tidak semua makhluk hidup berjalan (berjalan) sama. Umpamanya adalah manusia, manusia berjalan dengan dua kaki sementara makhluk lain ada yang berjalan dengan empat kaki seperti binatang ada juga yang

dua kaki seperti burung, maka kita buat dua interface yaitu mamalia dan pikiran, antarmuka mamalia berisi properti metode mamalia dan antarmuka pikiran berisi Metode pikiran . yang membuat perbedaan antara manusia dan hewan.



Penerapan kelas Abstrak dapat dilihat pada bagan struktur dibawah ini, yang terdiri dari Abstract class Maklukhidup, dimana pada kelas abstrak tersbut ada class manusia dan class hewan dan hewan dapat di pecah lagi dengan class hewan berjalan dengan 4 kaki dan berjalan dengan 2 kaki.

Implementasi Interface

Jika kelas dapat diperluas menggunakan kata kunci extends, antarmuka pengguna dapat di implementasikan menggunakan kata kunci implements. Implementasi antarmuka memiliki konsep yang mirip dengan ekstensi di kelas abstrak dengan aturan berikut:

- ✓ Implementasi antarmuka mewarisi semua metode dan konstanta.
- ✓ Setiap class yang mengimplementasikan interface harus menimpa dan mendeklarasikan ulang setiap metodenya untuk membuat instance class.
- ✓ Jika satu atau lebih metode tidak diganti, kelas harus didefinisikan sebagai abstrak.

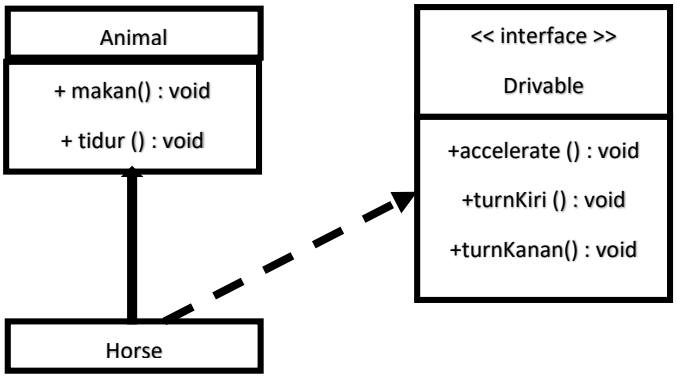
Pewarisan Berganda

Warisan berganda berarti pewarisan properti dalam dua kelas atau lebih. Pada dasarnya, Java tidak mengizinkan sebuah class untuk meng-extend dua class pada waktu yang bersamaan. Di Java, pewarisan berganda dilakukan dengan memperluas kelas dan mengimplementasikan satu atau lebih antarmuka.

Contoh dari beberapa kasus penggunaan yang diwariskan adalah kelas Kendaraan, Hewan, dan Kuda.

- ✓ kelas kendaraan (Vehicle) memiliki fungsi Accelerate (mempercepat), lampu kiri (belok kiri) dan lampu kanan (belok kanan)
- ✓ Kelas hewan adalah fungsi untuk makan (eat) dan tidur (sleep).

Pertanyaannya, bagaimana dengan kelas kuda, dimana kuda adalah hewan yang bisa ditunggangi. Karena kelas kuda tidak dapat memperluas kendaraan dan hewan, solusinya adalah mengimplementasikan pewarisan berganda menggunakan antarmuka Drivable.



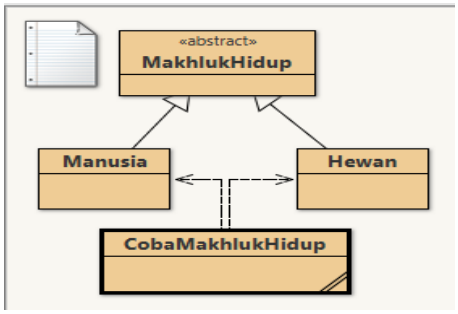
Pada diagram kelas di atas, Anda dapat melihat bahwa kelas Kuda memperluas kelas Hewan dan mengimplementasikan antarmuka Drivable, di mana Kuda mewarisi atribut dan metode dari Hewan dan metode dari Drivable.

Gunakan antarmuka untuk mendefinisikan metode standar yang sama di kelas yang berbeda. Setelah membuat satu set definisi metode standar (abstrak), kita dapat menulis satu metode (tujuan khusus) untuk menangani semua kelas yang mengimplementasikan antarmuka ini. Gunakan kelas abstrak untuk mendefinisikan secara luas properti kelas tertinggi dalam hierarki OOP, dan gunakan subkelasnya (subkelas) untuk melengkapi deskripsi metode kelas abstrak.

Praktikum Abstract Kelas dan Interface

1. Penerpan kelas Abstrak pada Makluk Hidup

Pada program berikut adalah sebuah abstrak kelas dan Interface kita akan coba buat kelas abstrak dengan nama abtrak MakhlukHidup pada kelas abstrak ini memliliki class terurunan yaitu manusia dan hewan. Dan untuk pengujian nantinya kita juga sediakan kelas CobaMakhlukHidup. Untuk dapat memahami lebih jelas bisa dilihat pada diagram berikut ini.



✓ **Class MakhlukHidup**

Langkah pertama Anda buat abstrak kelas MakhlukHidup dimana nantinya pada kelas ini memiliki dua kelas yaitu kelas manusia dan kelas hewan. Pada abstrak makhlukhidup terdapat pertanyaan yang bersifat umum yaitu setiap makhluk hidup bernapas dan setiap makhluk hidup makan. Pada kelas ini tetap menggunakan konsep pewarisan. Untuk lebih jelasnya dapat dilihat kode program pada gambar dibawah ini.

```
public abstract class MakhlukHidup
{
    public void bernapas()
    {
        System.out.println("Makhluk hidup bernapas...");
    }

    public void makan()
    {
        System.out.println("Makhluk hidup makan...");
    }

    public abstract void berjalan();

}
```

✓ **Class Manusia Turunan dari Class MaklukHidup**

Langkah kedua adalah dengan membaut kelas manusia turunan dari makhluk hidup, setiap manusia pasti bernapas dan makan karena merupakan turunan dari abstarak makhluk hidup. Pada kelas manusia ini berjalan dengan 2 kaki.

```
public class Manusia extends MakhlukHidup
{

    public Manusia()
    {

    }

    public void berjalan()
    {
        System.out.println("Manusia berjalan dengan 2 kaki");
    }
}
```

✓ **Class Hewan Turunan dari Class MaklukHidup**

Langkah ketiga adalah dengan membaut kelas hewan turunan dari makhluk hidup, setiap hewan pasti bernapas dan makan karena merupakan turunan dari abstrak makhluk hidup. Pada kelas hewan ini ada yang berjalan dengan 2 kaki dan 4 kaki bahkan lebih dari itu. Untuk penggalan kode program dapat dilihat pada gambar dibawah.

```
public class Hewan extends MakhlukHidup
{

    public Hewan()
    {

    }

    public void berjalan()
    {
        System.out.println("Hewan berjalan ada yang 2 kaki ada 4 kaki");
    }
}
```

✓ **Class CobaMakhlukHidup**

Langkah keempat adalah dengan membaut kelas cobamakhlukhidup, pada kelas ini nantinya akan kita gunakan sebagai bentuk test dari program yang kita buat sudah betul apa belum dengan memanggil kelas manusia dan kelas hewan.

```
public class CobaMakhlukHidup
{
    public static void main (String args[])
    {
        System.out.println("\u000c");
        Manusia mnsia = new Manusia();
        Hewan hwan = new Hewan();

        mnsia.bernapas();
        mnsia.makan();
        mnsia.berjalan();

        hwan.bernapas();
        hwan.makan();
        hwan.berjalan();
    }
}
```

✓ **Hasil Program**

Hasil program yang Anda jalankan pada kelas CobaMkhlukHidup jika semua kode sudah tidak ada yang salah adalah seperti tampilan dibawah ini.

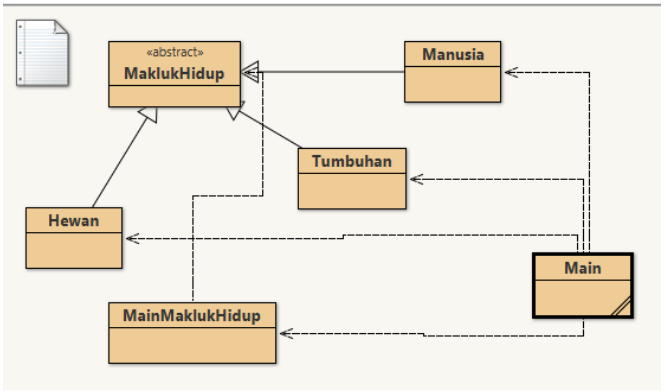
BlueJ: Terminal Window - abstract-interface

Options

```
Makhluk hidup bernapas...
Makhluk hidup makan...
Manusia berjalan dengan 2 kaki
Makhluk hidup bernapas...
Makhluk hidup makan...
Hewan berjalan ada yang 2 kaki ada 4 kaki
```

2. Penerpan kelas Abstrak pada Makluk Hidup

Pada program berikut sebenarnya sama dengan sebelumnya cuman disini kita tambahkan menjadi lebih banyak contoh pada kelas abstrak makhlukhidup yang sifatnya masih abstrak diturunkan beberapa spesifikasi yang lebih jelas bahwa makhlukhidup ada manusia, hewan , tumbuhan , dll. Dan untuk lebih jelas mengenai struktur diagaram rangkaiannya bisa dapat Anda lihat pada struktur dibawah ini.



✓ **Class Abstract Makluk Hidup**

```
public abstract class MaklukHidup{

    public abstract void berdiri ();

    public void oksigen(){
        System.out.println("- butuh Makanan");
        System.out.println("- butuh oksigen");
        System.out.println("- butuh air");
    }
}
```

✓ **Class Manusia**

```
public class Manusia extends MaklukHidup{
    private String duaKaki;

    public Manusia(String duaKaki){
        this.duaKaki = duaKaki;
    }
    public void berdiri (){
        System.out.println("Manusia berdiri menggunakan :  "+duaKaki);
    }
}
```

✓ **Clas Hewan**

```
public class Hewan extends MaklukHidup {

    private String kakiEmpat, kakiDua ;
    public Hewan(String kakiEmpat, String kakiDua){
        this.kakiEmpat = kakiEmpat;
        this.kakiDua   = kakiDua;
    }

    public void berdiri (){
        System.out.println("Kucing berdiri menggunakan  :  " +kakiEmpat);
        System.out.println("Ayam berdiri menggunakan    :  " + kakiDua);
    }
}
```

✓ **Class Tumbuhan**

```
public class Tumbuhan extends MaklukHidup{
    private String Akar;
    public Tumbuhan (String Akar){
        this.Akar = Akar;
    }
    public void berdiri (){
        System.out.println("Tumbuhan berdiri dengan     :  "+Akar);
    }
}
```

✓ **Class MainMaklukhiudp**

```
public class MainMaklukHidup
{
    public void cekMaklukHidup(MaklukHidup mHidup){
        mHidup.berdiri();
        mHidup.oksigen();
     }
}
```

✓ **Class Main**

```
public class Main
{
    public static void main(String[] args) {
        System.out.println("\u000c");
        MainMaklukHidup tMaklukHidup = new MainMaklukHidup();

        tMaklukHidup.cekMaklukHidup(new Manusia("Dua Kaki"));

        System.out.println("----------------------------------------");
        tMaklukHidup.cekMaklukHidup(new Hewan ("Empat Kaki","Dua Kaki"));

        System.out.println("----------------------------------------");
        tMaklukHidup.cekMaklukHidup(new Tumbuhan ("Akar"));

    }
}
```

Hasil Program Abstrak Kelas Makluk Hidup

BlueJ: Terminal Window - AbstakKelas

Options

```
Manusia berdiri menggunakan :  Dua Kaki
- butuh Makanan
- butuh oksigen
- butuh air
----------------------------------------
Kucing berdiri menggunakan  :  Empat Kaki
Ayam berdiri menggunakan    :  Dua Kaki
- butuh Makanan
- butuh oksigen
- butuh air
----------------------------------------
Tumbuhan berdiri dengan     :  Akar
- butuh Makanan
- butuh oksigen
- butuh air
```

BAB XI
PENGANTAR *GRAFICAL USER INTERFACE*
***(* GUI)**

Grafical User Interface (GUI)

Grafical User Interface (GUI) merupakan antarmuka pengguna grafis. GUI merupakan aplikasi desain dengan tata letak beruapa visual untuk memungkinkan pengguna menggunakan aplikasi dengan nyaman. Java Foundation Class (JFC), elemen penting dari Java SDK, berisi kumpulan API yang memfasilitasi pembangunan aplikasi JAVA GUI. JFC termasuk dalam lima komponen utama API, yaitu AWT dan Swing. Tiga bagian lain dari antarmuka aplikasi adalah Java2D, Aksesibilitas, dan Seret dan Jatuhkan. Hal ini mempermudah pengembang mendesaian dan menerapkan aplikasi GUI yang lebih baik.

AWT dan Swing memfasilitasi elemen GUI yang dapat dipakai untuk membangun aplikasi dan applet Java. Tidak semua bagian dari AWT yang menggunakan kode asli, Swing semuanya ditulis dalam bahasa pemrograman Java. Swing menawarkan penerapan platform-independen, di mana aplikasi yang dibagun pada platform yang berbeda dapat mempunyai interaface yang sama. AWT juga menjamin bahwa aplikasi yang berjalan di dua komputer berbeda terlihat sama. Swing API terdiri dari beberapa API yang menerapkan berbagai bagian AWT. Pendeknya, komponen AWT dapat dipakai secara bersama dengan komponen Swing.

Saat ini ada tiga API Java untuk pemrograman grafis: AWT (Abstract Windowing Toolkit), Swing dan JavaFX.

- ✓ AWT API diperkenalkan di JDK 1.0. Sebagian besar komponen UI AWT telah usang dan harus diganti dengan komponen UI Swing yang lebih baru.
- ✓ Swing API, kumpulan pustaka grafis yang jauh lebih komprehensif yang meningkatkan AWT, diperkenalkan sebagai bagian dari Java Foundation Classes (JFC) setelah rilis JDK 1.1. JFC terdiri dari Swing, Java2D, JFC telah diintegrasikan ke dalam inti Java sejak JDK 1.2.
- ✓ JavaFX terbaru, yang diintegrasikan ke dalam JDK 8, dimaksudkan untuk menggantikan Swing. JavaFX dipindahkan dari JDK di JDK 11, tetapi masih tersedia sebagai modul terpisah.

Selain API grafis AWT/Swing/JavaFX yang disediakan di JDK, organisasi/vendor lain juga telah menyediakan API grafis yang bekerja dengan Java, seperti Standard Widget Toolkit (SWT) Eclipse (digunakan di Eclipse), Google Web Toolkit (GWT) (digunakan di Android), 3D Graphics API seperti Java bindings untuk OpenGL (JOGL), Java3D, dan lain-lain. Selain itu, pengembang telah berpindah untuk menggunakan teknologi seperti HTML5 sebagai basis aplikasi web.

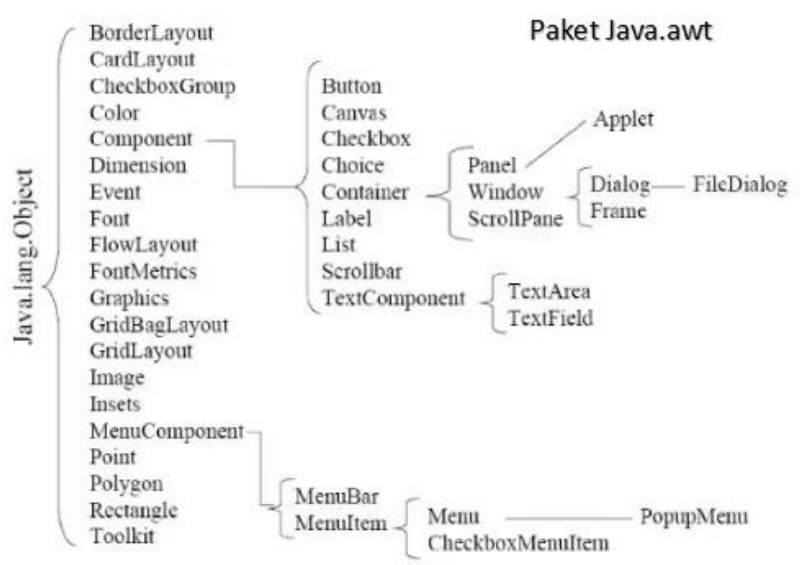
Aplikasi GUI yang dirancang secara teknis di Java dapat diimplementasikan dengan menggunakan dua cara umum. Cara kesatu yaitu dengan menggunakan cara scripting yang biasa dikenal dengan istilah hard coding. Metode ini membutuhkan pemahaman konseptual dan cara praktik yangg cukup baik. Tentunya developer membutuhkan waktu yang cukup untuk mempelajari secara kesemuanya. Demikian pula, seorang pengembang membuatuhkan waktu untuk mendesain aplikasi GUI menggunakan teknik yang ada tidak menghalangi kemampuannya untuk melakukannya dengan cepat. Itu semua tergantung pada keterampilan dan jumlah latihan dan fokus.

- ✓ AWT (Abstract Window Toolkit), toolkit GUI ke satu yang dibuat oleh Microsoft. Toolkit AWT memiliki kelemahan, khususnya kisaran alat yang diperlukan untuk merancang aplikasi GUI;

- ✓ Selain AWT Sun Microsystems, mereka juga mengembangkan sebuah paket yang disebut SWT (Standard Widget Toolkit); dan,
- ✓ Jawa Swing. Fitur Java Swing sangat cocok untuk mengembangkan aplikasi GUI di Java karena bersifat native, artinya aplikasi yang dikembangkan dengan Java Swing dapat digunakan pada beberapa jendela sistem operasi. Beda dengan paket AWT yang dibangun dan mendukung penuh Java Aplikasi pada sistem operasi Windows, SWT dibangun khusus mendukung aplikasi Java pada sistem operasi yang hampir sama dengan Unix.

Komponen GUI di AWT

Distribusi elemen GUI di AWT (paket java.awt) ditunjukkan pada struktur di bawah ini



Berikut adalah keterangan daftar dari beberapa kelas kontainer penting yang disiapkan oleh AWT.

- ✓ Component
  Kelas abstrak untuk objek yang muncul di konsol dan yang dapat berinteraksi dengan pengguna. Tubuh semua kelas AWT.
- ✓ Container
  Subkelas abstrak dari kelas komponen. Sebuah komponen yang dapat berisi elemen atau bagian lainnya.
- ✓ Panel
  Berasal dari kelas kontainer. Bingkai atau jendela tanpa bilah judul, bilah menu tidak berisi bingkai. Bagian atas kelas applet.
- ✓ Windows
  Contoh kelas Container. Jendela tingkat atas, mis. itu tidak dapat disematkan di objek lain, tanpa bingkai dan tanpa bilah menu.
- ✓ Frame
  Berasal dari kelas jendela. Jendela dengan judul, bilah menu, batas, dan ukuran sudut. Ini memiliki empat konstruktor, dua di antaranya menulis:

```
Frame()
Frame(String title)
```

Untuk mengatur ukuran jendela, kita gunakan metode setSize.

```
void setSize(int width, int height)
```

Untuk mengatur ukuran elemen ini dengan perintah width dan height sebagai parameter.

```
void setSize(Dimension d)
```

Untuk meng-edit ukuran kita bisa menggunakan perintah d.width dan d.height ini tentu saja dengna merujuk ketentuan dari spesifikasinya. Dimensi d.Default dari window adalah tidak tampil atau tidak kelihatan sehingga kita perlu mengatur menjadi visibility true agar tmpil atau kelihatan . Untuk rumus metode setVisible.

```
void setVisible(boolean b)
```

Komponen GUI pada Swing

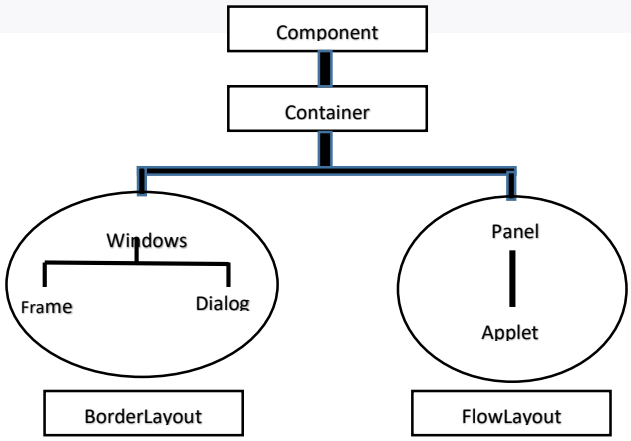
Komponen atau Elemen Swing GUI terdapat pada paket javax.swing. Selanjutna meruapakan daftar beberapa komponen atau elemen Swing:

- ✓ **JComponent**
  Merupakan Kelas utama untuk semua elemen pada Swing kecuali kontainer tingkat lanjut
- ✓ **Jframe**
  Warisan kelas kerangka dan yang setara dalam paket AWT, tetapi keduanya sedikit bertentangan ketika harus menambahkan komponen ke wadah. Anda harus mendapatkan ubin konten terbaru sebelum menambahkan komponen.
- ✓ **Jpanel**
  Turunan komponen-J. Wadah kelas sederhana, tetapi bukan wadah tingkat lanjut.
- ✓ **Japplet**
  Derivatif dan ekuivalen kelas Applet pada paket AWT. Tidak sedikit yang kompatibel dengan kelas applet dalam hal menambahkan komponen ke wadah
- ✓ **Jbutton**
  Tombol Perintah atau Jbutton merupakan elemen berbentuk tombol. Elemen ini sering digunakan untuk melakukan suatu tindakan yang perlulkan. Sebuah Aplikasi komputer umumnya membutuhkan tombol untuk meng-eksekusi perintah.
- ✓ **Jlabel**
  Merupakan elemen yang dipakai untuk membuat teks atau gambar dalam bingkai untuk memberi tahu pengguna tentang program.
- ✓ **JtextField**
  Merupakan komponen untuk masukan berupa string yang selanjutnya dapat dipakai sebagai proses masukan pada proses selanjutnya.
- ✓ **JtextArea**
  Komponennya hampir sama dengan JTextField tetapi dapat berisi lebih dari satu baris.
- ✓ **JcheckBox**
  Merupakan elemen yang dipakai ketika user membutuhkan elemen untuk membuat satu atau beberapa pilihan sekaligus.
- ✓ **JradioButton**
  Merupakan elemen yang dipakai ketika user harus memilih salah satu dari beberapa pilihan yang ada.
- ✓ **JcomboBox**
  Merupakan elemen yang wajib memilih salah satu dari banyak pilihan seperti kolom Teks dengan panah bawah.
- ✓ **JfileChooser**
  Memungkinkan user untuk melaksanakan pemilihan file.
- ✓ **JcolorChooser**
  Merupakan Turunan dari komponen-J. Yang memebrikan kesempatan user untuk memilih warna.

- ✓ **Jtable**
  Jtabel biasanya dipakai pengguna untuk menampilkan atau list data dalam bentuk excel
- ✓ **JscrollPane**
  Merupakan elemen yang berfungsi untuk menampilkan objek ke semua tempat baik ke kesamping, ke atas, ke bawah sehingga semua objek nantinya akan terlihat di layar.
- ✓ **Jmenu**
  Merupakan bagian dalam pembuatan menu.
- ✓ **JinternalFrame**
  Merupakan bingkai yang hanya bisa ada di dalam bingkai lain.
- ✓ **JoptionPane**
  Turunan komponen-J. Ditujukan untuk membuat pop-up lebih mudah dilihat.
- ✓ **Jdialog**
  Warisan dan kesetaraan di jendela kelas paket AWT. Biasanya digunakan untuk memberi tahu pengguna tentang sesuatu atau meminta pengguna untuk memasukkan.

Layout Manager

Pengaturan tata letak yang dipakai guna menata posisi elemen gambar program sesuai dengan tata letak antarmuka pengguna. Java telah menyediakan beberapa pilihan tata letak, dan keputusan untuk menggunakan tipe tata letak tertentu bergantung pada sifat aplikasi dan tingkat kebersihan yang diinginkan.



Berikut adalah beberapa pengelola tata letak Java:

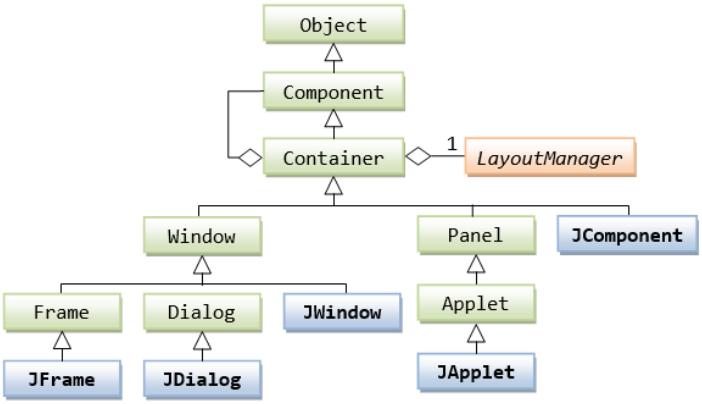
- ✓ **Flow Layout**
  Jenis tata letak paling sederhana, di mana semua komponen disusun dari kiri ke kanan di sepanjang bingkai, bergerak ke bawah saat mencapai tepi bingkai. Standarnya adalah java.awt
- ✓ **Border Layout**
  Tata letak jenis ini bekerja dengan membagi bingkai menjadi lima bagian, yaitu top, left, bawah, right, dan center. Komponen visual dapat ditempatkan di bagian ini
- ✓ **None Layout**
  Jenis susunan ini dapat menciptakan tampilan yang bagus dan serasi, karena kita dapat menyempurnakan posisi komponen menggunakan koordinatnya. Hasil dari pengaturan ini adalah kali ini

proporsional lebih dari tata letak lain karena kita harus menentukan koordinat lokasi setiap komponen.

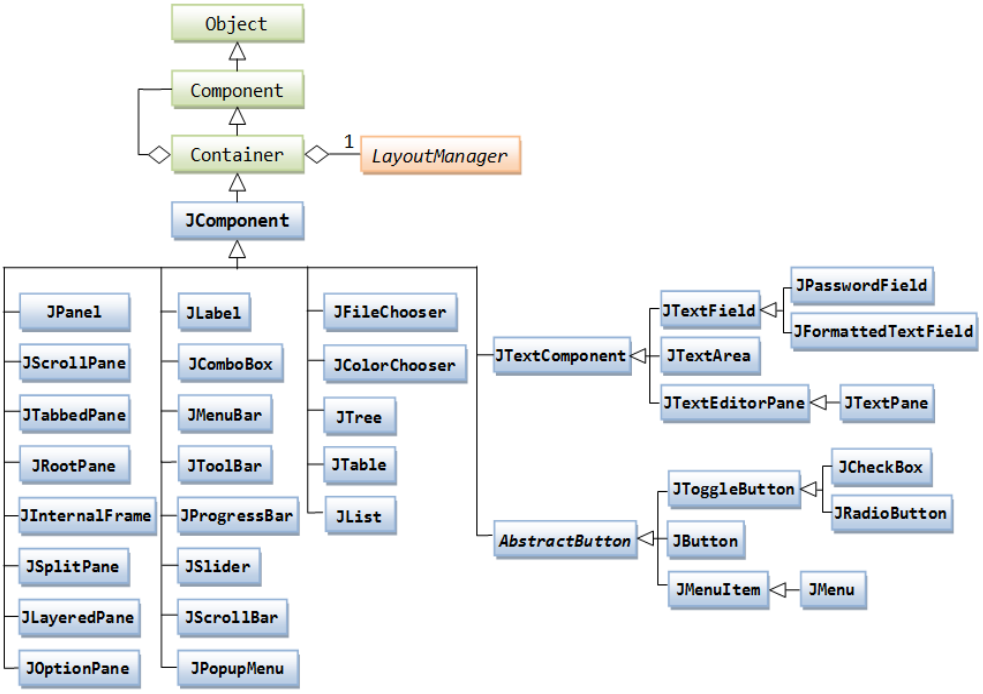
- ✓ **Grid Layout**
  Merupakan bentuk tata letak berdasarkan pada larik dan lajur. Dengan penyiapan ini, kita dapat memberikan pernyataan jumlah larik dan lajur sesuai kebutuhan
- ✓ **GridBagLayout**
  Ukuran kisi dapat bervariasi, kisi dapat menampung lebih dari satu komponen.
- ✓ **Cardlaout**
  Komponen diganti sebagai peta, hanya satu komponen yang terlihat pada satu waktu.
- ✓ **BoxLaout**
  Merupakan elemen atau komponen yang digunakan untuk menyusun tampilan dari posisi Kanan ke kiri atau dari bawah ke atas. Tata letak pengelola dapat diatur menggunakan metode setLayout dari Kelas Kontainer.

Jika Anda tidak ingin memakai pengelola tata letak, Anda bisa meneruskan argumen null ke metode ini. Namun, di masa mendatang, Anda dapat mensetting posisi komponen secara manual memakai metode setBounds dari kelas Komponen.
setBounds kekosongan publik (int x, int y, lebar int, tinggi int), Metode ini menentukan tata letak berdasarkan posisi x dan posisi y dan ukuran berdasarkan anggapan lebar dan tinggi. Ini bisa menjadi sangat rumit dan kurang tepat untuk aplikasi jika ada beberapa elemen objek di dalam objek kontainer. Anda harus menerapkan cara ini untuk setiap komponen.

Sedangkan untuk hirarki kelas Swing Jcomponent adalah sebagai berikut. Jcomponent dan keturunannya adalah komponen yang ringan.



Hirarki kelas wadah tingkat atas Swing (JFrame, JDialog, JApplet) adalah sebagai berikut. Kontainer Swing tingkat atas ini sangat berat, yang bergantung pada subsistem windowing yang mendasari platform asli.



JAVA Swing

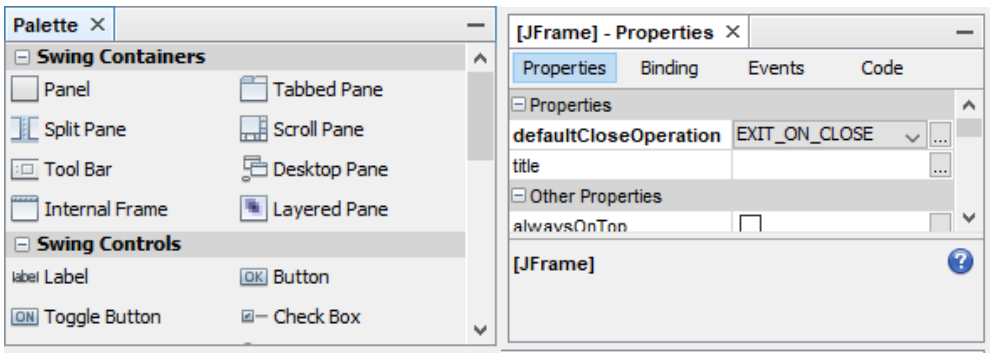
Keuntungan Java Swing adalah menyediakan berbagai alat (palet), yang membuat hidup lebih mudah bagi pengembang, terutama pengembang pemula. Secara umum paket Java Swing berisi beberapa tool tampak dengan jelas dapat Anda lihat pada gambar berikut ini.

Tabel 11.1 Tabel Tools

No	Nama Tools	Fungsi
1	jLabel	Digunakan untuk meletakan *caption* (*labeling*) pada sebuah tools lain yang memiliki
2	jTtextField	Digunakan untuk kotak input dan keluaran
3	jPanel	Umumnya digunakan untuk mengkoleksi tools berdasarkan kategori
4	jComboBox	Digunakan untuk pemilihan secara *full down option*
5	jRadioButton	Digunakan untuk pemilihan secara klik
6	jButton	Digunakan untuk memanggil *script* untuk melakukan proses tersentu

Palette dan Properties

Toolkit Java hampir sama dengan Microsoft Visual Studio. Palet Java Bluej adalah kotak alat yang menyediakan berbagai alat yang disebutkan dalam dokumentasi Java Swing. Secara visual, palet terlihat seperti gambar berikut:



Properti adalah komponen yang digunakan untuk mengubah properti tools dari kotak tools, seperti : Nama variabel, nama properti alat dll.

- ✓ **Frame (Form)**
  Bingkai adalah jendela desain (bentuk) aplikasi Java Swing GUI. Alat-alat ini disimpan di kelas Formulir Java Jframe
- ✓ **Menggunakan Tools**
  Beberapa komponen tool sudah dijelaskan di atas pada anotasi swing JAVA, kemudian bagian ini menjelaskan penggunaan tool, fungsinya dan aplikasinya pada pengkodean program. Perlu dicatat bahwa GUI dan alat digunakan di sini dengan prinsip seret dan lepas. Prinsip drag and drop adalah dengan mengklik dan menahan alat, menyeret alat yang dipilih, lalu meletakkannya di jendela formulir gambar
- ✓ **jLabel**
  jLabel adalah alat yang biasa dipakai untuk menuliskan (nama) label. jLabel tidak menerima masukan tetapi dapat digunakan sebagai jendela keluaran. Dapat dikatakan bahwa jLabel mewakili bidang atau tajuk kolom dalam sebuah tabel. Anda dapat memakainya dengan mengklik alat jLabel dan menyeret dan menjatuhkannya di jendela formulir menggambar. Untuk properti jLabel, cukup mengubah teks biasanya sudah cukup, tetapi jika Anda ingin mengubah latar belakang pembelajaran, ukuran font, dll., Anda dapat mengubahnya di jendela properti, yang dapat diakses di tab Tautan, Acara, dan Kode. Sedangkan nama peubah dapat diatur menggunakan tab Code.
- ✓ **jTextField**
  jTextField adalah alat yang bekerja dengan kolom input. Pada Implementasinya, tool ini membutuhkan peubah komponen teks, nama variabel dan properti background sesuai kebutuhan.
- ✓ **Input Bilangan jTextField**
  Ada beberapa perbedaan saat memasukkan jTextField, terutama karena tipe datanya. Karena input yang dibuat dengan alat ini pada dasarnya dianggap sebagai string, maka perlu untuk mengubahnya menjadi data tertentu tergantung pada tipe datanya.
- ✓ **Menampung Output**
  Hasil yang digunakan oleh komponen tertentu bisa juga divisualkan atau diterima oleh komponen lain. Desainer dapat menggunakan alat yang dikehendaki tenunya yang sesuai dengan kebutuhan mereka. Sifat dari alat ini yaitu hanya mampu menerima data yang bertipe string, sehingga hasilnya harus diubah terlebih dahulu menjadi bentuk string sebelum dapat ditampilkan di tools lainnya. Karena jTextField adalah kolom input, input yang diterima dapat berupa data integer, pecahan, atau karakter dan string.

- ✓ **jComboBox**

  Ada alat yang merekam opsi sepenuhnya. Alat ini biasanya memerlukan pendefinisian properti untuk nama peubah komponen dan list anggota. Ada macam macam metode untuk meregistrasi option di jComboBox, yaitu dengan mengatur properti dan coding dengan metode .addItem(). Dua cara termudah disajikan di bawah ini.

- ✓ **jRadioButton dan jCheckBox**

  jRadioButton adalah merupakan tomobl pilihan yang meminta pengguna untuk memilih satu dari sekian banyak pilihan yang adada, radiobutton sebenarnya dapat dipakai untuk beberapa pilihan akan tetapi diajurkan pada satu pilihan saja. jCheckBox beda dengan rdiobutton yang hanya memlih satu pilihan saja, checkbox boleh memilih lebih dari satu pilihan , juga bisa memlih semua pilihan yang ada.

- ✓ **jButton**

  jButton adalah komponen yang digunakan untuk menjalankan perintah yang ditulis oleh user di dalam komponen ini. Pengaturan properti alat ini cukup untuk komponen teks dan nama variabel.

- ✓ **jTable**

  Ini adalah alat yang menampilkan informasi tentang basis data aplikasi. Untuk menyetel properti jTable, Anda hanya memerlukan komponen nama variabel dan desain tajuk kolom (bidang). Namun, ini tergantung pada kebutuhan dan selera Anda. Ada beberapa cara untuk mengkonfigurasi kolom jTable, pada modul ini direpresentasikan dengan dua cara yaitu melalui properties dan coding window.

- ✓ **import**

  javax.swing.table.DefaultTableModel; dan metode .addColumn(); Metode AddColumn() dapat ditempatkan di dalam konstruktor kelas. Misalnya, kode kelas bernama Tabel dapat ditulis dalam konstruktor public Table(){ }

- ✓ **Menampilkan Data ke jTable**

  Terkadang data input perlu disajikan dalam tabel agar mudah dipahami. Menampilkan data dalam tabel berdasarkan letak data tersebut. Input data dalam tabel database dapat berbeda dengan input data jTextField. Kali ini kami menjelaskan bagaimana data ditampilkan oleh jTextField. Sebagai contoh, desain data mengacu pada tabel di atas, dimana desain antarmuka form input dapat dirancang.

  Selain itu, tampilan data yang dimasukkan dengan jTextField (Nama, NPM) dapat dilakukan dengan tombol Input Data.

- ✓ **jCalender**

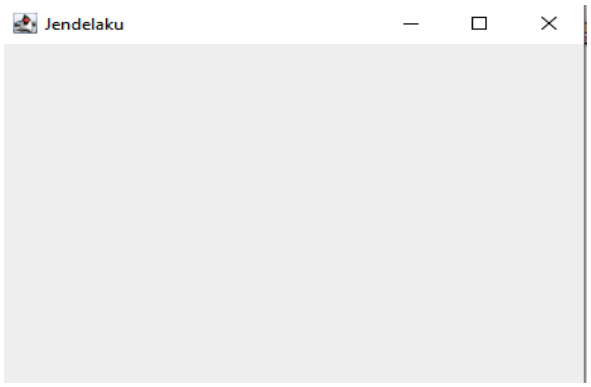
  Java juga memfasiliasi jCalender yang merupakan sebuah alat yang digunakan pada editor NetBeans tetapi memerlukan beberapa konfigurasi karena jCalender terkadang tidak jadi satu dengan editor. Untuk menggunakan perlu kita download dan mengistallnya terlbih dahulu baru bisa digunakan.

Praktikum Program dengan menggunakan GUI

1. Pembuatan Jendela dengan menggunkan Jframe

```
import javax.swing.*;
public class Jendelaku
{
    public static void main(String[] args) {
        // membuat objek jendela
        JFrame myWindow = new JFrame();

        // berikan judul pada jendela
        myWindow.setTitle("Jendelaku");

        // tentukan ukuran jendela
        myWindow.setSize(400, 300);

        // tampilkan jendela ke layar
        myWindow.setVisible(true);

    }
}
```

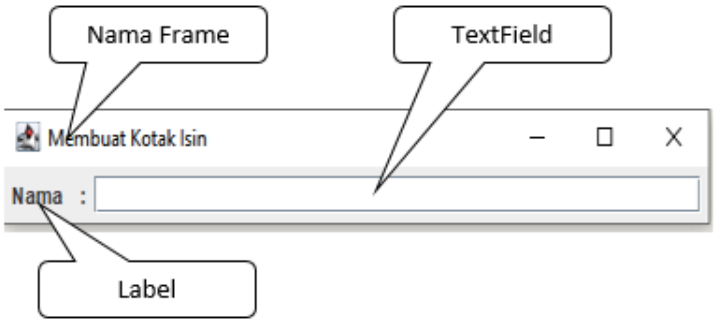
Pertama kita buat kelas dengan nama Jendelaku, selanjutnya kita tuliskan programnya jangan lupa untuk import javax swing, selanjutnya bisa membuat jendela sesuai dengan kebutuhan kita dengan tinggi dan lebarnya berapa, pada contoh diatas adalah jendela dengan ukuran 400 X 300 dengan nama jendela yaitu Jendelaku. Hasil program diatas adalah seperti berikut.



2. Pembuatan GUI untuk label dan textfield atau kotak isian

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class KomponenTeks
{
    public static void main(String[ ] args) {
        JFrame frame = new JFrame("Membuat Kotak Isin");
        JLabel label = new JLabel("Nama    :");
        JTextField textField = new JTextField(40);
        frame.getContentPane().setLayout(new FlowLayout());
        frame.getContentPane().add(label);
        frame.getContentPane().add(textField);
        frame.pack();
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Setelah selesai membuat kelas selanjutnya adalah menulikan source code programnya. Kita import javax swing, java awt, java awt event karena GUI berada dibawah naungan javax tersebut. Untuk membuat label dan kotak isian kita perlu komponen label dan textfield. Dan Hasil program tersebut adalah.



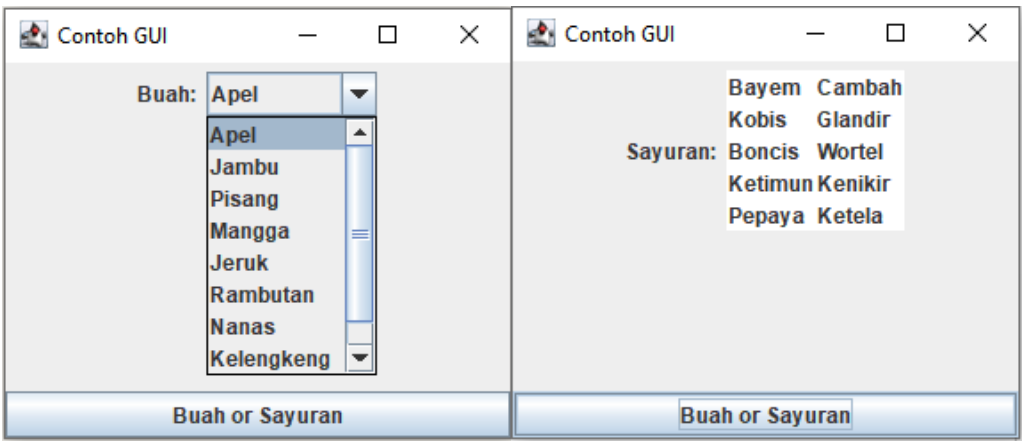
3. Pembuatan GUI untuk label dan textfield atau kotak isian

Langkah pertam silahkan Anda buat Class dengan nama Class GuiApi, jangan lupa untuk import javax.swing dan javax.awt. Disini kita akan menggunakan fasilitas combo box dan button untuk membuat tampilan pada java. Untuk lebih lengakapny sielahkan anda ketikan kode programnya sampai selesai dan sdh tidak ada kesalahan.

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JComboBox;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JList;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class GuiApi
{
  public static void main(String[] args) {
      new GuiApi();
}
    public GuiApi()
    {
      JFrame guiFrame = new JFrame();

      guiFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      guiFrame.setTitle("Contoh GUI");
      guiFrame.setSize(300,250);

      guiFrame.setLocationRelativeTo(null);
,
      String[] fruitOptions = {"Apel", "Jambu", "Pisang"
          ,"Mangga", "Jeruk", "Rambutan", "Nanas", "Kelengkeng", "Duku"};
          //Options for the JList
          String[] vegOptions = {"Bayem", "Cambah", "Kobis", "Glandir"
              , "Boncis", "Wortel", "Ketimun", "Kenikir", "Pepaya"
              , "Ketela"};

       final JPanel comboPanel = new JPanel();
       JLabel comboLbl = new JLabel("Buah:");
       JComboBox fruits = new JComboBox(fruitOptions);
       comboPanel.add(comboLbl);
       comboPanel.add(fruits);
```

```
        final JPanel listPanel = new JPanel();
        listPanel.setVisible(false);
        JLabel listLbl = new JLabel("Sayuran:");
        JList vegs = new JList(vegOptions);
        vegs.setLayoutOrientation(JList.HORIZONTAL_WRAP);
        listPanel.add(listLbl);
        listPanel.add(vegs);
        JButton vegFruitBut = new JButton( "Buah or Sayuran");
        vegFruitBut.addActionListener(new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent event)
            {
                listPanel.setVisible(!listPanel.isVisible());
                comboPanel.setVisible(!comboPanel.isVisible());
            }
});
    guiFrame.add(comboPanel, BorderLayout.NORTH);
    guiFrame.add(listPanel, BorderLayout.CENTER);
    guiFrame.add(vegFruitBut,BorderLayout.SOUTH);
    //make sure the JFrame is visible
    guiFrame.setVisible(true);
}
}
```

Pada source code programnya diatas adalah menampilkan komponen combo box atau daftar pilihan berupa dropdown list dan tombol untuk mengganti teks dalam bentul label pada tombol Buah Or Sayuran. Untuk hasil programnya dapat dilihat pada tampilan dibawah ini.



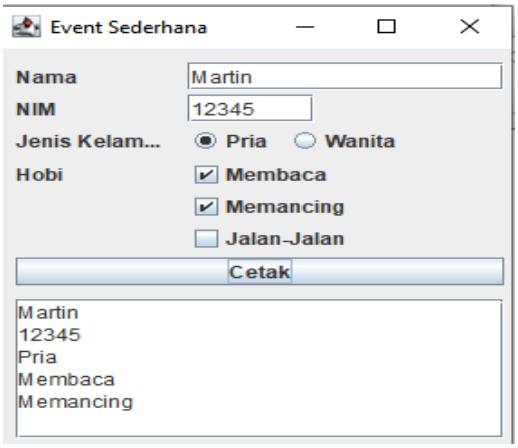
4. **Pembuatan GUI untuk beberapa komponen diantaranya adalah label, textfield , option button, check box, list box, button, dan jendela atau frame**

Langkah pertama silahkan Anda buat Class dengan nama Class EventSederhana jangan lupa untuk import javax.swing dan javax.awt , dll. Disini kita akan menggunakan fasilitas label, text field, combo box , check box dan button untuk membuat tampilan pada java. Untuk lebih lengakapnya silahkan anda ketikan kode programnya sampai selesai dan sdh tidak ada kesalahan.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class EventSederhana
{
    private JFrame frame;

    JLabel lblnama=new JLabel("Nama");
        JTextField txnama=new JTextField(20);
        JLabel lblnim=new JLabel("NIM");
        JTextField txnim=new JTextField(7);
        JLabel lblkelamin=new JLabel("Jenis Kelamin");
        JRadioButton pria=new JRadioButton("Pria");
        JRadioButton wanita=new JRadioButton("Wanita");
        ButtonGroup kelompok=new ButtonGroup();
        JLabel lblhobi=new JLabel("Hobi");
        JCheckBox baca=new JCheckBox("Membaca");
        JCheckBox mancing=new JCheckBox("Memancing");
        JCheckBox jalan=new JCheckBox("Jalan-Jalan");
        JButton cetak=new JButton("Cetak");
        JTextArea hasil=new JTextArea();
    public EventSederhana()
    {
        makeFrame();
        frame.setVisible(true);
        AksiReaksi();
    }

    public void setVisible(boolean visible)
    {
        // put your code here
        frame.setVisible(visible);
    }

    private void makeFrame()
    {
        frame = new JFrame("Event Sederhana");
        frame.setSize(300,300);
        komponenVisual();
    }

    private void komponenVisual(){
        JPanel panel = (JPanel)frame.getContentPane();
        panel.setLayout(null);
```

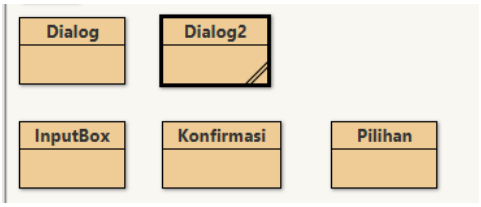
```
            panel.add(lblnama);
            lblnama.setBounds(10,10,80,20);
            panel.add(txnama);
            txnama.setBounds(105,10,175,20);
            panel.add(lblnim);
            lblnim.setBounds(10,33,80,20);
            panel.add(txnim);
            txnim.setBounds(105,33,70,20);
            panel.add(lblkelamin);
            lblkelamin.setBounds(10,56,80,20);
            kelompok.add(pria);
            kelompok.add(wanita);
            panel.add(pria);
            pria.setBounds(105,56,50,20);
            panel.add(wanita);
            wanita.setBounds(160,56,70,20);
            panel.add(lblhobi);
            lblhobi.setBounds(10,80,70,20);
            panel.add(baca);
            baca.setBounds(105,80,100,20);
            panel.add(mancing);

            mancing.setBounds(105,103,100,20);
            panel.add(jalan);
            jalan.setBounds(105,126,100,20);
            panel.add(cetak);
            cetak.setBounds(10,150,270,20);

            JScrollPane scrollPane = new JScrollPane(hasil);
            panel.add(scrollPane);
            scrollPane.setBounds(10,180,270,100);
        }
    public void AksiReaksi()
    {
        cetak.addActionListener(new ActionListener(){
                    public void actionPerformed(ActionEvent e){
                            hasil.append(txnama.getText()+"\n");
                            hasil.append(txnim.getText()+"\n");
                            if(pria.isSelected()==true){
                                    hasil.append(pria.getText()+"\n");
                            }
                            else{
                                    hasil.append(wanita.getText()+"\n");
                            }
                            if(baca.isSelected()==true){
                                    hasil.append(baca.getText()+"\n");
                            }
                            if(mancing.isSelected()==true){
                                    hasil.append(mancing.getText()+"\n");
                            }
                            if(jalan.isSelected()==true){
                                    hasil.append(jalan.getText()+"\n");
                            }
                    }
                });
public void AksiReaksi()
{
    cetak.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        hasil.append(txnama.getText()+"\n");
                        hasil.append(txnim.getText()+"\n");
                        if(pria.isSelected()==true){
                                hasil.append(pria.getText()+"\n");
                        }
                        else{
                                hasil.append(wanita.getText()+"\n");
                        }
                        if(baca.isSelected()==true){
                                hasil.append(baca.getText()+"\n");
                        }
                        if(mancing.isSelected()==true){
                                hasil.append(mancing.getText()+"\n");
                        }
                        if(jalan.isSelected()==true){
                                hasil.append(jalan.getText()+"\n");
                        }
                }
            });
}}
```

Untuk hasil program tersebut diatas dapat dilihat pada tampilan dibawah ini.



5. Pembuatan GUI untuk beberapa kotak dialog

Menampilkan beberapa pesan melalui dialogbox diataranya adalah MessageBox, DialogBox, Konfirmasi , InputBox dan Pilihan.



✓ Kotak Pesan Menampilkan pesan selamat datang di Bluej dengan joptionepane dan show message dialog

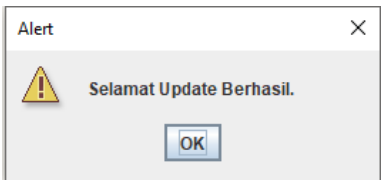
```
import javax.swing.*;

public class Dialog extends javax.swing.JFrame
{
  JFrame f;
  Dialog(){
    f=new JFrame();
    JOptionPane.showMessageDialog(f,"Selamat Datang di BlueJ.");
}
public static void main(String[] args) {
    new Dialog();
}
}
```

✓ Hasil Listing Program diatas akan terlihat seperti gambar berikut , menampilkan pesan dialog selamat update berhasil di Bluej

✓ MessageDialogbox pesan menampilkan pesan selamat update berhasil dengan joptionepane dan show message dialog, dengan model tombol OK

```
import javax.swing.*;

public class Dialog2 extends javax.swing.JFrame
{
   JFrame f;
  Dialog2(){
    f=new JFrame();
    JOptionPane.showMessageDialog(f,"Selamat Update Berhasil.",
    "Alert",JOptionPane.WARNING_MESSAGE);
}
public static void main(String[] args) {
    new Dialog2();
}  }
```

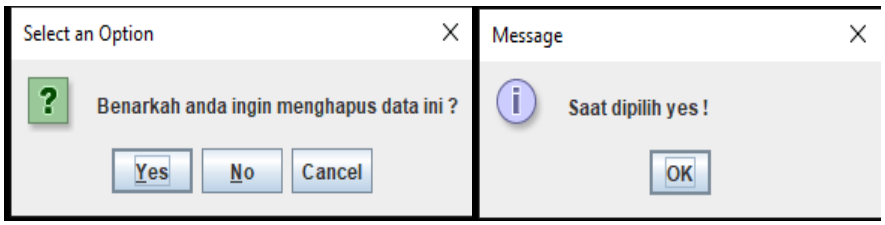
✓ Hasil Listing Program diatas jika di eksekusi akan mengahasilkan tampilan seperti gambar berikut



✓ MessageDialogbox Pesan Menampilkan pesan Seleksi atau pemilihan tombol Yes, No dan Cancel dengan menggunakan JoptionePane

```
import javax.swing.*;
import java.awt.event.*;
import javax.swing.JDesktopPane;
import javax.swing.JOptionPane;
public class Pilihan extends WindowAdapter
{
    public static void main(String[]args){
        int opsi = JOptionPane.showConfirmDialog(null, "Benarkah anda ingin menghapus data ini ?");
        switch(opsi){
            case JOptionPane.YES_OPTION:
                JOptionPane.showMessageDialog(null, "Saat dipilih yes !");
                break;
            case JOptionPane.NO_OPTION:
                JOptionPane.showMessageDialog(null, "Saat dipilih no !");
                break;
            default:
                JOptionPane.showMessageDialog(null, "Saat dipilih cancel !");
                break;
        }
    }
}
```

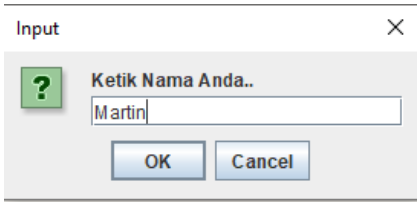
✓ Hasil Listing Program diatas akan terlihat seperti gambar berikut



✓ Messagedialog dengan Pesan Menampilkan Input Box Ketik Nama Anda dengan tomobl Ok dan Cansel

```
import javax.swing.*;
public class InputBox
{
   JFrame f;
InputBox(){
    f=new JFrame();
    String name=JOptionPane.showInputDialog(f,"Ketik Nama Anda..");
}
public static void main(String[] args) {
    new InputBox();
}
}
```

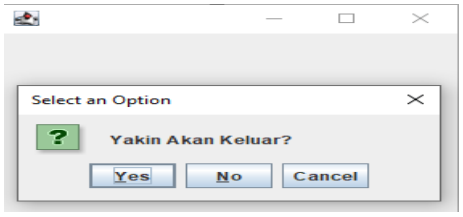
✓ Hasil Listing Program diatas akan terlihat seperti gambar berikut



✓ Kotak Pesan Menampilkan pesan konfirmasi Yakin Akan Keluar dengan tombol Yes, No dan Cancel ketika jendela ti tutup.

```
import javax.swing.*;
import java.awt.event.*;

public class Konfirmasi extends WindowAdapter
{
JFrame f;
Konfirmasi(){
    f=new JFrame();
    f.addWindowListener(this);
    f.setSize(300, 300);
    f.setLayout(null);
    f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    f.setVisible(true);
}
public void windowClosing(WindowEvent e) {
    int a=JOptionPane.showConfirmDialog(f,"Yakin Akan Keluar?");
if(a==JOptionPane.YES_OPTION){
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
public static void main(String[] args) {
    new  Konfirmasi();
}
}
```

✓ Hasil Listing Program diatas akan terlihat seperti gambar berikut



BAB XII
*UNIFIED MODELING LANGUAGE (*UML)

Unified Modeling Language

Unified Modeling Language atau bahasa pemrograman terpadu merupakan salah satu komponen yang dapat dipakai dalam membangun sebuah pemrograman berbasis objek. Bahasa pemrograman terpadu atau UML meruapakan bahasa baku yang menajdi ketetapan untuk masa yang akan datang dalam industri pengembangan aplikasi perangkat lunak berbasis objek, Hampir semua perusahaan besar menggunakannya. Pengenrtian UML menurut beberapa ahli:

- ✓ Bahasa Pemrograman Terpadu (UML) adalah sebuah alat yang diginakan untuk pengembangan perangkat lunak (sistem informasi) dengan menggunakan bentuk tampilan grafis dan UML merupakan bahasa lambang, difinitif, membangun dan mendokumentasikan(arsip).
- ✓ Bahasa Pemrograman Terpadu (UML) adalah meruapkan bahasa yang menjadi ketetapan untuk memvisualisasikan, mendefinisikan, mengembangkan dan dokumentasi alat dari sebuah sistem perangkat lunak.
- ✓ Bahasa Pemrograman Terpadu (UML) dapat diartikan sebagai bahasa baku yang digunakan dalam industri untuk visualisasi, desain, dan dokumentasi sistem perangkat lunak.

Unified Modeling Language (UML) adalah bahasa pemodelan standar yang terdiri dari kumpulan diagram yang dirancang untuk membantu perancang sistem dan perangkat lunak melakukan tugas-tugas seperti:

- Spesifikasi (data teknis)
- Visualisasi
- Desain arsitektur
- Konstruksi
- Simulasi dan testing
- Dokumentasi

Dari uraian yang telah disebutkan diatas s, dapat disimpulkan bahwa Unified Modeling Language (UML) atau Bahasa Pemrogram Terpadau adalah sebuah bahasa berbasis grafis atau gambar untuk memvisualisasikan, mendefinisikan, membangun dan mendokumentasikan pengembangan sistem perangkat lunak berbasis objek.

Bahasa Pemrograman Terpadu banyak dijumpai pada pemrograman berbasis objek. Bahasa Pemrograman Terpadu adalah meruapakan ketetapan baku untuk mempolkan visual atau keadaan yang sebenarnya pada suatu sistem yang nyata. Pemodelan Bahasa Pemrograman Terpadu (UML) awalnya dipakai dalam proses pengembangan perangkat lunak pada tahap awal yang dirujuk oleh program berbasis objek. Melalui UML ini, user dapat menentukan, menguraikan secara visual, membangun dan mendokumentasikan dari berbagai elemen sistem perangkat lunak. Dimana UML digunakan untuk:

- ✓ menggambarkan diagram domain permasalahan,
- ✓ disain perangkat lunak yang diusulkan, atau,
- ✓ juga implementasi perangkat lunak yang sudah selesai (siap pakai).

Tata letak dan spesifikasi serta implementasi merupakan keterkaitan dengan perangkat lunak dibangun, bisa dikatakan bahwa, kedua alat tersebut berhubungan dengan source code. Dengan alat spesifikasi user dapat memahami entiti atau attribut properti apa saja yang digunakan dalam suatu perangkat lunak. Begitu juga pada diagram implementasi berfungsi menterjemahkan bagaimana perangkat lunak ditulis dalam kode program-nya. Selajutnya diagram ini memiliki sudut pandang yang berbeda dan bersifat standart meskipun tidak terlalu beresiko.

Perbedaan antara diagram implementasi dan spesifikasi adalah dengan diagram konseptual tidak bersentuhan dengan kode program-nya, melainkan hanya memberikan penjelasan konsep dan abstraksi yang besinggungan dengan permasalahan user atau pengembang. Hal ini dapat menjelaskan bagaimana suatu kelas memiliki hubungan yang wajar dengan kelas lain, dan sejauah mana hubungan kelas kelas tersebut dapat berjalan dengan baik.

Ruang Lingkup UML

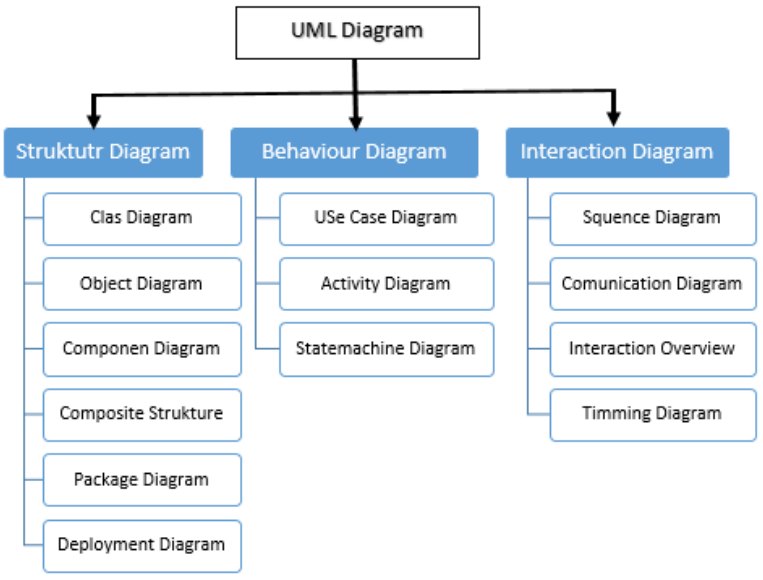
Dalam Kerangka kualifikasi Bahasa Pemrograman Baku *(UML)* tersedia pola pola yang tepat dan tidak menimbulkan salah persepsi *(ambigu)* serta lengkap. Dalam arti khusus, Bahasa Pemrograman Baku *(UML)* memformulasikan tahapan tahapan penting dalam pengambilan dalam setiap tindakan keputusan baik dalam perencanaan, analisis, perancangan serta penerapan dalam sistem yang sangat diperlukan dalam perangkat lunak *(software intensive system).*

Bahasa Pemrograman Baku *(UML) ini sebenar adalah* bukanlah bahasa pemprograman melaikan pola pola yang terbentuk dan berhubungan langsung dengan bermacam macam bahasa pemprograman, sehingga dimungkinkan dapat melakukan pemetaan *(mapping)* langsung dari pola pola yang dibuat dengan U*nified Modeling Language (UML)* dengan bahasa-bahasa pemprograman berbasis obyek, seperti pada *Java BlueJ*.

Mapping Unified Modeling Language (UML) dapat bersifat dua arah yaitu :

1. Pertama adalah Code Generation ini adalah bahasa pemprograman jenis tertentu dari U*nified Modeling Language (UML) forward engineering.*
2. Kedua adalaha Code Generation tetapi tidak sesuai dengan standart baku dan harapan oleh pengguna, pengemabang dapat melakukan langkah demi langkah yang bersifat *pencegahan* dari penerapan ke dalam U*nified Modeling Language (UML)* sampai pada sistem / peranti lunak yang sesuai dengan harapan user dan programer.

Jenis Jenis UML



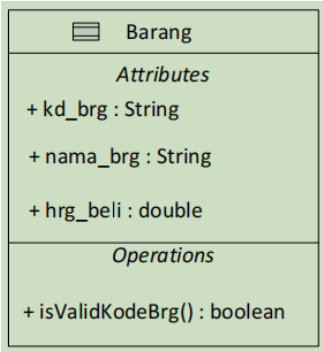
Berikut ini jenis-jenis dari UML, antara lain:

Structure Diagram

Diagram struktur adalah sekumpulan diagram yang berfungsi untuk mendifinisikan suatu struktur yang bersifat statis dari sistem yang dipolakan.

- ✓ **Class diagram**. Meruapakan salah satu dari jenis diagram kelas pada UML yang dipilih untuk menampilkan data data yang dikirimkan ataupun kelas-kelas yang ada pada sistem yang akan dipakai oleh sistem tersebut. Kesimpulannya bahwa diagram kelas ini bisa memberikan sebuah visualisasi mengenai sistem maupun hubungan hubungan yang terdapat pada sistem tersebut.

  Kelas merupakan sebuah cara suatu objek, yang memiliki atribut atau properti dan servis atau fungsional (metode/fungsi). Kelas diagram menggambarkan struktur dan deskripsi kelas, package dan objek beserta hubungan satu sama lain, seperti hal yang esenssial : Nama (dan stereotype), Atribut, dan Metode

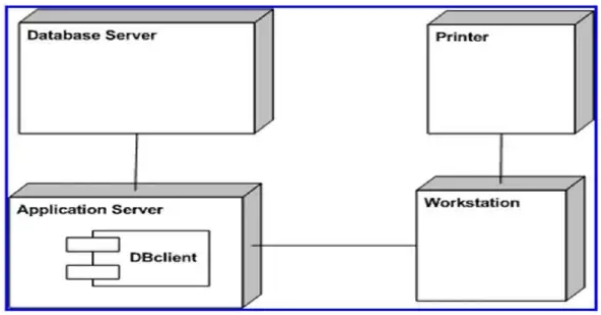


- ✓ **Component diagram** adalah merupakan salah satu jenis diagram UML yang menggambarkan bentuk perangkat lunak pada suatu sistem. Komponen diagram adalah penerapan perangkat lunak dari satu ataupun lebih kelas, dan umumnya berupa file data atau .exe, source kode, table, dokumen dsb.

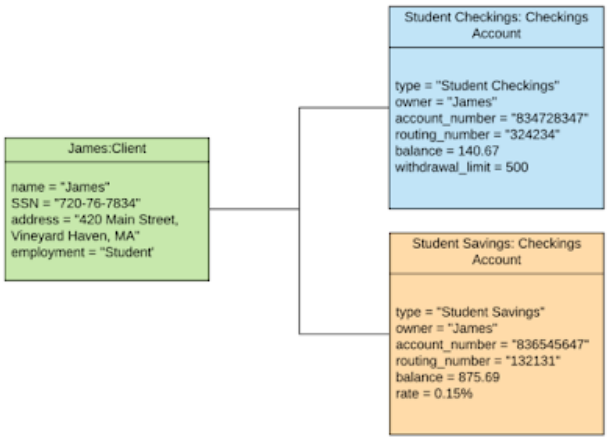
  Component Diagram adalah merupakan diagram UML yang memiliki peran yaitu menampilkan elemen pada system dan hubungan antara diagram tersebut. Pada saat melakukan relasi dengan dokumentasi sistem yang kompleks, Komponen diagram dapat membantu dalam memecahkan sistem menjadi elemen elemen yang lebih kecil.

  Tujuan utama dari komponen diagram ini adalah:

  - ~ Penggambaran elemen elemen dari suatu sistem
  - ~ Membuat procedure atau program kecil yang dapat dijalankan dengan menggunakan teknik *forward* dan *reverse engineering*
  - ~ Mendifinisikan secara komplek setiap hubungan dari komponen

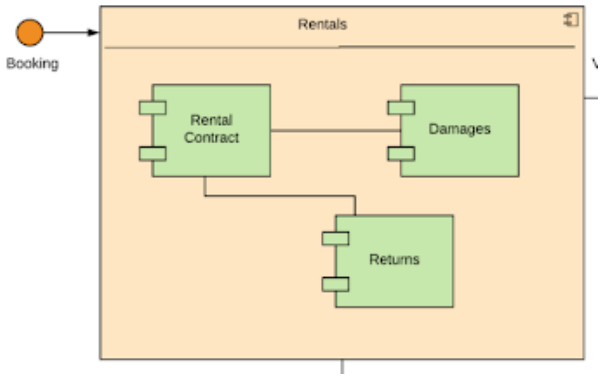


✓ **Object diagram** adalah meruapakan satu jenis diagram pada UML yang mendifinisikan objek - objek tersebut pada suatu sistem dan hubungan anatar obejek tersebut.
Pada pengembangan aplikasis sistem, Kelas pada umumnya adalah tipe data abstrak, beda dengan objek. Objek merupakan turunan dari kelas abstrak. Umpamanya, jika kita memiliki kelas "Mobil" yang merupakan tipe abstrak yang masih umum, maka turunan dari kelas "Mobil" ini bisa macam macam salah satunya adalah "Toyata". Toyata sendiri jika masih terlalu abstrak dapat di turnakan lagi menjadi “Rush”.



✓ **Deployment diagram** adalah meruapakan macam diagram pada UML yang mengatur desaian objek sebuah program. Dalam bentuk lain bisa diartikan sebagai visualisasi bagian-bagian aplikasi yang ada pada perangkat keras dan dipakai untuk menerapkan suatu sistem dan hubungan antara elemen perangkat keras.
Deployment diagram diterapkan pada visualisasi relasi antara perangkat lunak dan perangkat keras. Lebih tepatnya jika menggunakan diagram deployment diagram maka dapat dibuat pola fisik tentang bagaimana elemen perangkat lunak (tata letak perangakat lunaknya) digunakan pada elemen perangkat keras, yang dipahami sebagai node. Kesimpulannya Deployment diagram merupakan inti untuk menunjukan letak perangkat lunak yang digunakan pada perangkat keras pada sistem yang digunakan.

✓ **Composite structure diagram**. Adalah merupakan satu jenis diagram pada UML yang menjelaskan struktur dalam yang terdiri dari komponen, kelas, dan use case, termasuk juga hubungan pengklasifikasian ke elemen lain dari sebuah aplikasi sistem. Ini hampir mirip seperti class diagram akan tetapi composite structure diagram menggambarkan elemen elemen dari individu kelas saja bukan semua kelas.
Composite structure diagram Ini adalah jenis UML diagram yang jarang dipakai hal ini disebabkan karena fungsinya sangat tertentu sekalai. Composite structure diagram hanya mewakili struktur internal kelas dan hubungan antara elemen kelas yang berbeda.

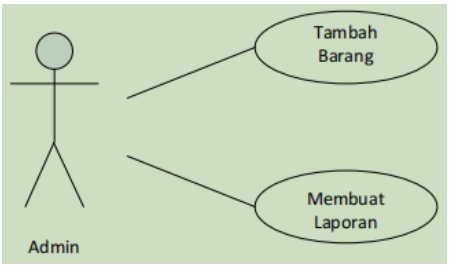


- ✓ **Package diagram**. Adlah merupakan satu jenis diagram pada UML yang memiliki fungsi sebagai pengumpul kelas dan juga memperlihatkan bagaimana elemen pola disusun serta memvisualisasikan ketergantungan antara paket – paket yang ada.

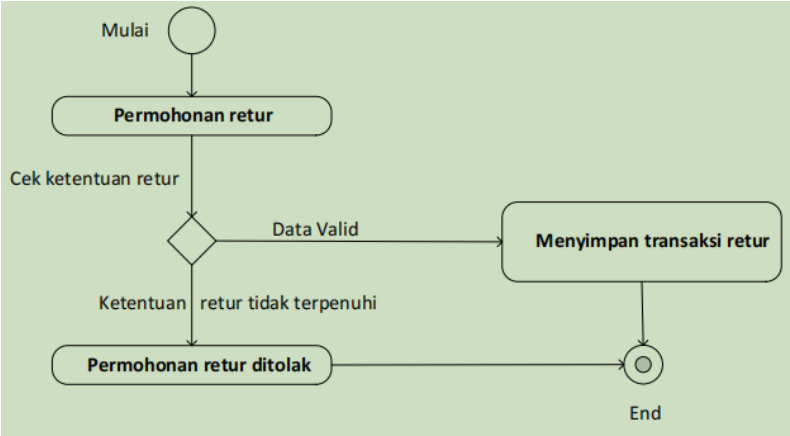
Behaviour Diagram

Diagram Behaviour adalah merupakan sebuah kumpulan diagram yang dipakai untuk meggambarkan perilaku sistem atau untaian perbaikan atau perubahan yang terlaksana pada sebuah sistem.

- ✓ **Use Case Diagram**, Use case diagram yaitu adalah salah satu pola diagram pada UML yang memvisualisasikan hubungan antara sistem dan pelaku sistem, use case diagram juga dapat memaparkan tipe interaksi antara pengguna ataua pemakai sistem dengan sistemnya. Use case dieksekusi melalui cara menggambarkan tipe interaksi antara user.
  Use case diagram mendifinisikan sebagai fungsionalitas dari sebuah sistem (apa fungsinya), yang merepresentasikan sebuah interaksi antara pelaku dan sistem



- ✓ **Activity Diagram**. Adalah bagian dari jenis diagram pada UML yang dapat mempolakan cara apa saja yang terjadi pada sebuah sistem.
  Activity Diagram **Activity Diagram** berbagai kegiatan dalam sistem yang sedang di desaian, mulai dari awal permulaan, melalui kondisi (*decision*) yang mungkin terjadi, lanjut sampai pada titik akhir. Diagram ini juga mampu memvisualisasikan seuat proses bersambung yang bisa saja terjadi pada beberapa eksekusi. Diagram ini tidak memvisualisasikan perilaku/proses internal sebuah sistem maupun interaksi antar-subsistem, tetapi lebih menggambarkan proses-proses dan jalur-jalur kegiatan secara umum.



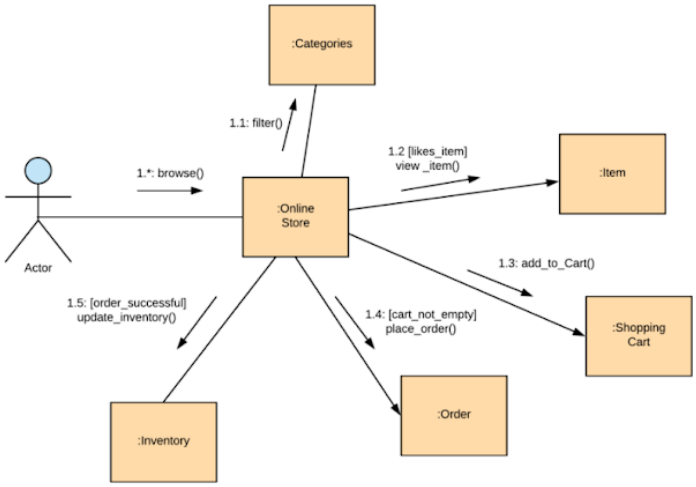
- ✓ **Statemachine diagram**. Adalah meruapak bagian dari jenis diagram pada UML yang memvisualisasikan perubahan keadaan maupun transisi suatu suatu objek pada sistem.

Interaction Diagram

Interaction diagram merupakan kumpulan diagram yang berguna untuk menggambarkan interaksi antar sistem yang satu dengan sistem kedua dan seterusnya maupun antar sistem pada sebuah sistem.

- ✓ **Sequence diagram**. Merupakan jenis diagram UML yang memvisualisasikan hubungan anatar objek berdasarkan kejadian urutan waktu. Sequence diagram dapat memvisualisasikan tahapan atau prioritas yang harus dikerjakan agar dapat menghasilkan sesuatu seperti pada halnya use case diagram.

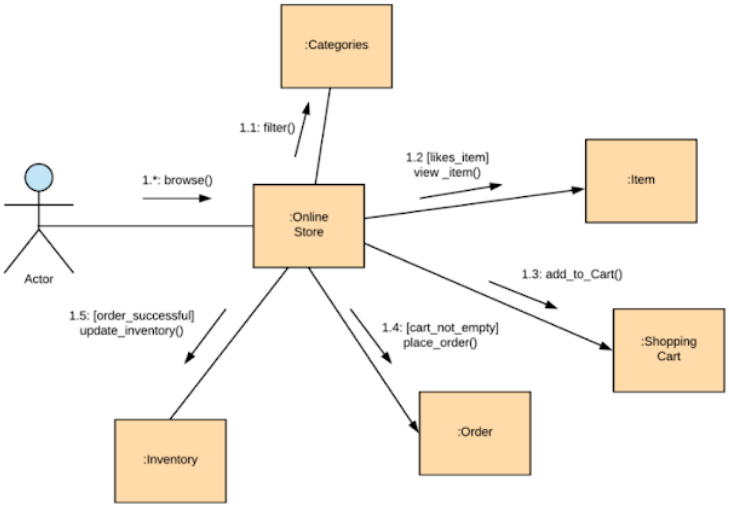
  Diagram Sequence merupakan diagram yang sangat penting dalam UML sebagai pola tingkat rancnagan untuk pengembangan bisnis aplikasi. Diagram ini memvisualisasikan urutan pesan dan kejadian yang terjadi antara pelaku dan objeknya.



- ✓ **Communication diagram** merupakan diagram yang dapat memvisualisasikan proses terjadinya suatu kegiatan dan diagram ini juga menggambarkan interaksi antara objek yang ada pada sebuah sistem. Pada prisifnya hampir sama dengan sequence diagram akan tetapi communication diagram lebih menekankan kepada peranan masing-masing objek pada sistem.

  Kesimpulannya diagram komunikasi ini lebih mudah didesain karena dapat menambahkan objek di papan visual. Setelah semua objek terhubung, maka objek

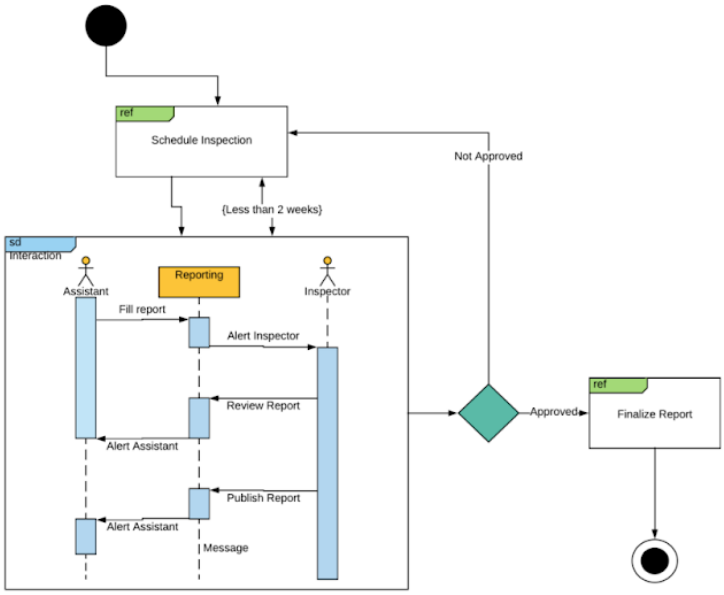
tersebut hanya perlu menjadi komponen dari nomor urutan, tidak harus secara nyata dekat satu sama lain.



✓ **Interaction Overview diagram**. Merupakan diagram UML yang berfungsi untuk menggambarkan hubungan dan kerjasama antara kegiatan diagram dengan kejadian urutan diagram.

Interaction overview diagram merupakan UML diagram yang paling kompleks jika dibandingkan dengan diagram lainnya. Masih ada lagi yaitu dalam behavioral UML diagram, ada juga yang dikenal sebagai Interaction Diagram atau diagram interaksi yang mana terdiri dari :

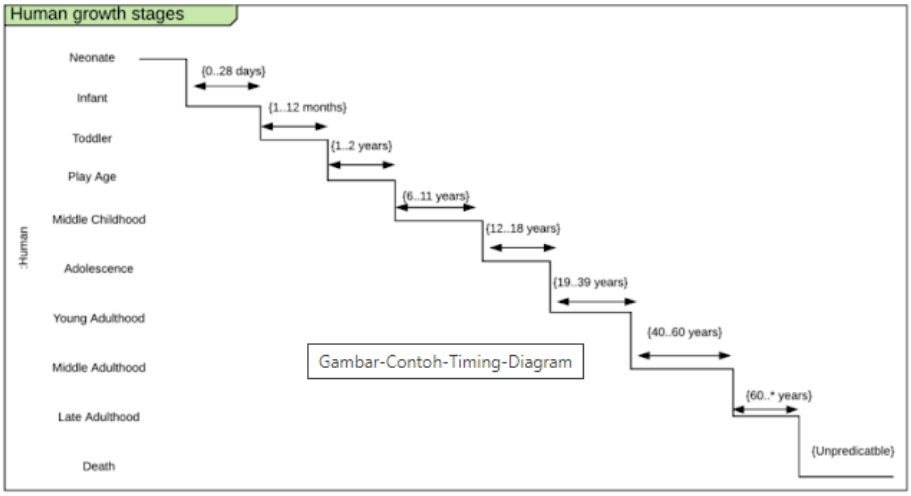
~ Interaction Overview Diagram,
~ Timing Diagram,
~ Sequence Diagram dan
~ Communication Diagram.



✓ **Timing diagram**. Adalah merupakan salah satu diagram UML berfungsi sebagai bentuk lain dari interaksi diagram, diagram ini fokus terhadap waktu. Diagram timing berguna

untuk mengambarkan variabel variabel faktor yang membatasi waktu antara perubahan status terhadap objek yang lain.

Timing diagram adalah diagram yang digunakan untuk mempresentasikan relasi antar objek saat pusat perhatian sedangkan melakukan pemberhentian sejenak. Pada dasarnya Anda tidak tertarik pada bagaimana objek berinteraksi satu sama lain, tetapi Anda ingin mempresentasikan bagaimana objek dan pelaku bertindak sepanjang waktu yang terus berjalan.



Jenis Diagram

Macam macam jenis diagram utama terdiri dari tiga(3) macam yaitu : diagram statis, dinamis, dan diagram fisik.

- ✓ Diagram Statis menjelaskan struktur logis dari komponen komponen perangkat lunak yang tetap dengan memvisualisasikan objek , kelas, struktur data, dan hubungan di antara onjek atau kelas tersebut.
- ✓ Dinamis Diagram menjelaskan entiti perangkat lunak berubah selama dijalankan dengan memvisualisasikan arah eksekusi dan cara dari entiti berubah.
- ✓ Fisik Diagram Menjelaskan struktur secara fisik entiti perangkat lunak yang tetap dengan memvisualisasikan entiti secara fisik, antara lain file sumber, library, file2 bit, file data, dan lain-lain., dan hubungan di antara file file tersebut..

Jika dilihat dari fungsinya maka ketiga diagram tersebut diatas sangat cocok dengan jenis pemodelan diagram dibawah ini :

- ✓ Diagram Struktural
- ✓ Diagram Tingkah Laku
- ✓ Diagram Arsitektur

✓ **Digram Struktural**

Structure Diagrams (Diagram strutktural) adalah salah satu pemodelan struktural diagram yang bersumber pada statis diagram dan sifat-sifat statis yang ada pada diagram tersebut. Statis Diagram ini memberi petunjuk kepada pengguna untuk menggambarkan suatu diagram ke dalam kode program. Pada umumnya, struktur digrami terdiri atas beberapa macam diagram antar laian yaitu kelas diagram, komponen diagram, penempatan diagram , objek diagram , paket diagram, profile diagram, dan struktur komposit diagram.

Struktural Diagram sebagai wakil dari frame work sistem, frame work ini merupakan tempat di mana pada semua elemen lain ada. Oleh sebab itu, kelas diagram, komponen diagram, penempatan diagram , objek diagram , paket diagram adalah bagian dari pemodelan struktural.

Diagram tersebut sebagai wakil komponen dan mekanisme untuk menyatukan. Karena sifatnya yang statis, maka kelompok struktural diagram, kelas diagram adalah diagram yang paling terakhir digunakan dalam perancangan uatu program.

- ✓ **Diagram Tingkah Laku**

  Behavior Diagram adalah jenis diagram yang bersifat dinamis. Diagram ini mengggambarkan frame work system. Diagram behavior Ini mewakili interaksi antara diagram struktural. Diagram tingkah laku menunjukkan sifat dinamis dari sistem. Yang termasuk kedalam golongan diagram ini adalah diagram aktivitas, diagram interaksi, diagram use case , diagram urutan, diagram keadaan, komunikasi diagram, ikhtisar diagram interaksi, dan, timming diagram.

- ✓ **Diagram Arsitektur**

  Arsitektur Diagram adalah merupakan tulang punggung dari keseluruhan sistem. Ini artinya, diagram struktural dan diagram tingkah laku dapat menjadi kesatuan yang tidak terpisahkan dalam sistem perangkat lunak. Arsitektur Diagram dapat menggambarkan sebagai bentuk dari cetak biru dari keseluruhan sistem. Diantara arsitektur banyak macam jenis untuk diagram ini, ada diagram paket (package diagram) juga termasuk di dalam kelompok diagram ini, walaupun diagram ini juga bisa terdapat pada diagram struktural.

Tujuan atau Fungi Penggunaan UML

Maksud tujuan dan fungsi dari penerapan UML, antara lain:

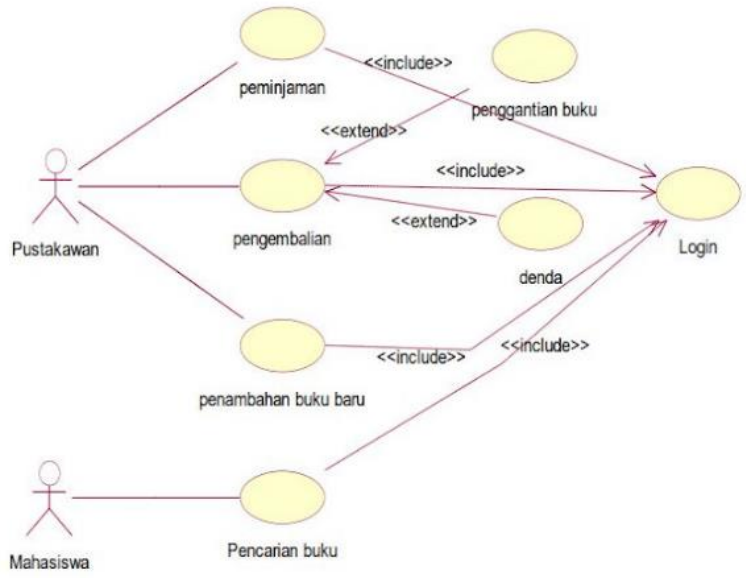
- ✓ Menjadikan bahasa standart baku untuk bahasa permodelan visual kepada pengembang.
- ✓ Menjadi suatu metode yang tepat dalam diagram pemodelan.
- ✓ Memberikan model yang siap digunakan, untuk saling berbagi model dengan cepat.
- ✓ Menjadikan cetak biru atau blue print, disebabkan kelengkapan dan detail dalam perancangan.
- ✓ Bisa mempolakan sistem yang berbasis objek, dan bukan hanya berguna untuk mempolakan perangkat lunak saja.
- ✓ Mempermudah dalam pemodelan yang akan dipergunakan oleh pengembang maupun penerjemah.

Notasi UML

Simbol	Nama	Keterangan
	Aktor	Menggambarkan seseorang, alat atau sistem yang lain yang berinteraksi dengan sistem. Dan dapat menerima dan memberi informasi pada sistem. aktor sama sekali tidak mengontrol sistem
	Use Case	Komponen selanjutnya adalah use case. Use case merupakan komponen gambaran fungsional dalam sebuah sistem. Dengan begitu, pengguna atau konsumen dapat mengetahui setiap fungsi yang dibangun dalam sistem tersebut.
	Association	Association adalah teknik yang digunakan untuk mengidentifikasi interaksi yang dilakukan oleh aktor tertentu dengan use case tertentu. Ini digambarkan dengan garis penghubung antara aktor dengan use case.

	Generalisasi	Generalisasi adalah hubungan antara dua use case atau dua aktor. Dimana salah satunya meng-*inherit* dan menambahkan atau *override* sifat dari yang lainnya.
	Note	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi
	Class	Kumpulan objek yang mempunyai attribut dan operasi
	Interface	Kumpulan dari operasi tanpa implementasi dari sebuah kelas
	Interaction	Dipakai untuk menunjukan aliran operasi dari sebuah pesan
	Relations	Hubungan element yang ada pada tanda panah akan menyatakan pernyataan elemen yang ada pada bagian tanda panah
	Devendency	Dependency adalah relasi yang terbagi menjadi dua jenis, yaitu include dan exclude. Include berfungsi untuk mengidentifikasi hubungan atau relasi antara dua use case, dimana use case yang satu akan memanggil use case lainnya. Sementara exclude merupakan jenis yang jika dilakukan pemanggilan memerlukan suatu kondisi tertentu dan akan terjadi dependensi
	Package	Sebuah Wadah yang dipakai untuk mengelompokan elemen elemen dari sebuah sytem yang di rancang atau dibangun

Contoh Use Case pada Sistem Perpustakaan



Menggunakan Diagram

Bahasa Pemrograman Baku atau UML mirip sama pemodelan diagram, dimana pada level konsep, spesifikasi, ataupun penerapan divisualisasikan ke dalam bentuk gambar. Ada tiga macam model diagram telah disampaikan pada awal, yang bisa digunakan sesuai keinginan dan bergantung kepada struktur mana yang akan dipakainya. Salah satu diantara diagram tersebut akan dikupas dibawah ini.

✓ **Diagram Kelas**

Diagram kelas atau class diagram pada dasarnya adalah representasi grafis dari tampilan statis suatu sistem dan mewakilinya sebagai bagian dari aplikasi.Selain itu, kumpulan diagram kelas mewakili keseluruhan sistem. Dan diagram kelas juga menggambarkan abstraksi yang mendefinisikan keseluruhan struktur dan perilaku suatu objek.

✓ **Struktur dan sintak diagram kelas**

Tiga bagian utama dari struktur diagram kelas antara laian nama kelas, atribut, dan operasi atau tingkahlaku. Untuk dapat memahami kelas tersebut perhatikan pada tabel dibawah ini :

Namakelas
- atribut : typedata ...
+tingkahlaku():returntype ….

Tabel di atas menunjukkan bahwa diagram kelas memiliki hubungan dengan kelas lain, terutama pada sistem di atas dengan banyak kelas yang terkait, kelas dikelompokkan bersama untuk membentuk diagram kelas. Hubungan antar kelas bisa berbeda-beda, namun biasanya ditunjukkan dengan ikon panah yang menunjuk ke kelas tertentu.

▪ **Notasi sintaks**

Saat mendesain diagram kelas, programmer harus memperhatikan notasi sintaksis, yang membahas beberapa persyaratan penting, termasuk pengubah, properti, dan jenis atribut dan metode. Pengubah adalah status atribut dan metode, apakah terlihat sebagai publik, pribadi, atau dilindungi. Properti atribut dan metode mengacu pada properti statis atau dinamis. Atribut membutuhkan tipe yang akan dibangun sesuai dengan batasan dan kriterianya, sedangkan metode memiliki tipe pengembalian tidak ada (kosong). Pengubah memiliki manfaat, lihat dengan cermat tabel di bawah ini untuk detailnya.

Modifier	Simbol	Hak Akses
Public	+	Dapat diakses oleh objek diluar kelas Private
Private	_	Hanya dapat diakses oleh objek di kelas yang sama Protected
Protected	#	Hanya dapat diakses oleh objek di kelas yang sama dan kelas-kelas turunannya
No Modifier		Pleksibel, dan umumnya dapat di deklarasi secara global

d

Ketika mendesain diagram kelas, perlu memperhatikan rambu rambu sebagai berikut :

~ nama kelas pada lajur pertama, atribut pada lajur kedua dan operasi terletak di dilajur ketiga.

~ Nama kelas biasanya kata benda berawalan huruf besar

~ Tidak menggunkan simbol simbol simbol tertentu

~ Method dan Atribut harus didifinisikan dengan baik
~ Tipe data pada atribut setelah titik dua umpama (atribut : tipedata)
~ Komponen dan relasinya harus didifiniskan sebelumnya
~ Banyaknya properti harus jelas sebab semakin banyak properti membuhkan perhatian yang serius
~ Mudah dipahami oleh pengembang program.

Selajutnya di jelaskan bahwa desain diagram kelas. Atribut dan method harus diidentifikasi terlebih dahulu sesuai dengan rambu rambu yang ada.

Umpama: *Barang memiliki keterangan - kodebarang, namabarang, hargabarang, dan jumlahbarang dan memiliki kegiatan seperti memasuka data, memproses data, mencetak data dan perbaikan data. Objek pada permsalahan ini akan diturunkan dari kelasnya. Untuk* lebih mudah memahaminya, perhatikan gambar dibawah ini adalah proses instansiasi objek yang diturunkan dari suatu kelas berikut ini :

Perhatikan dengan baik baik untuk daftar properti dan method yang merujuk pada contoh kasus diatas :

Properti	Method
Kodebar	Inputdata
Namabar	Inputdata
Harga	Editdata
Jumbar	Printdata

Nilai Properti	Method
789123456	Inputdata
Buku Tulis	Inputdata
5.000,00	Editdata
10	Printdata

Daftar properti dan metode di atas kemudian dapat ditata sesuai dengan simbol yang sesuai dalam bentuk diagram kelas. Perlu dicatat bahwa kelas mewakili konsep yang merangkum keadaan (status) dan perilaku (operasi). Setiap atribut memiliki tipe data dan setiap operasi memiliki simbol, sedangkan nama kelas adalah satu-satunya informasi yang diperlukan.
Diagram kelas di atas kemudian dapat diterjemahkan ke dalam bahasa pemrograman, seperti bahasa Java berikut:

Barang
- Kodebar : String - Namabar : String - Hargabar : Real - Jumbar : Integer
- Inputdata () : void - Editdata () : void - Cetakdata () : void

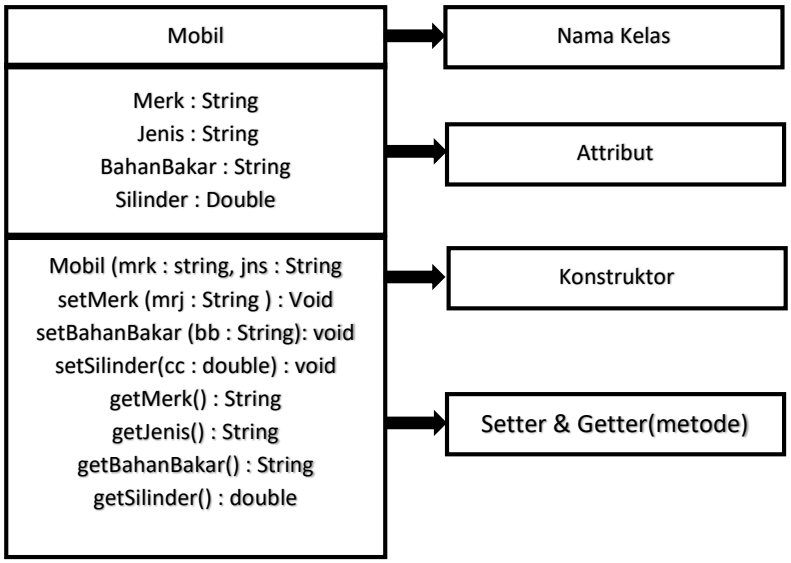
✓ **Diagram Objek**

Object Diagram atau Diagram objek dapat juga dikatakan sebagai diagram instansi, yang memvisualisasikan suatu keadaan yang sebenarnya. Objek Diagram memperlihatkan cara kerja bagaimana suatu sistem akan terlihat pada saat dijalankan (runtime*)*. Sebab itu karena objek diagram membawa data yang akan diproses untuk instansiasi, yang berlaku untuk menggambarkan hubungan yang kompleks sesama objek.

Pada kenyataannya, diagram objek mirip dengan diagram kelas, hanya saja diagram objek terdiri dari nama objek, nama kelas, dan instansinya. Lihat struktur objek penulisan pada rumus diagram objek berikut ini :

Namainstansi: namakelas
variabel = nilai

Penerapan Program UML pada JAVA Bluej



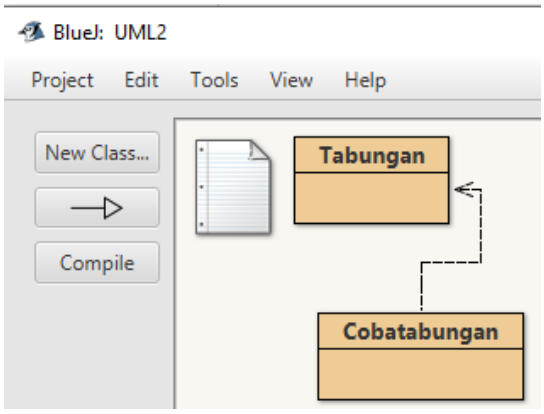
Implementasi Kelas

Untuk dapat melaksanakan penerapan kelas ikuti langkah langkah berikut yang dipakai untuk mentransformasikan kelas – kelas menjadi kode dalam pemrograman berbasis objek Langkah langkahnya adalah sebagai berikut :

- ✓ Penerapan kode program java untuk kelas – kelas yang ditemu pada diagram kelas, yaitu dengan membuat konstruktor dan setter serta getter untuk kelas – kelas tersebut.
- ✓ Menerapkan attribut – attribut yang ada dalam kelas menjadi kode – kode pada PBO, dan harus sesuai dengan konsep modulasi pada PBO, Penerapan attribut pada kelas ini pada prinsipnya adalah dengan visibitas PRIVATE(-).
- ✓ Menerapkan pola serta fungsi menjadi kode dalam PBO, untuk polanya sendiri memiliki fungsi umum pada visibilitas PUBLIK (+), dengan maksud agar pola dan fungsi dapat dipakai oleh kelas – kelas atau objek lain yang membutuhkannya.
- ✓ Menerapkan attribut – attribut serta pola-pola yang akan diwariskan menjadi hak akses protected sehingga sub kelas yang berada pada kelas induk dapat menggunkannya, tidak perlu melakukan pengetikan ulang, cara ini yang memungkinkan adanya penggunaan ulang komponen (reusable).

Praktikum Program UML pada Java

1. Program tabungan sederhana yang terdiri dari class Tabungan dan Cobatabungan



✓ **Class Tabungan**

Buatlah kelas baru dengan nama class tabungan, pada class ini terdapat public saldo dan ambil uang. Untuk dapat lebih memahami silahkan Anda lihat pada gambar berikut dan juga untuk listing programnya.

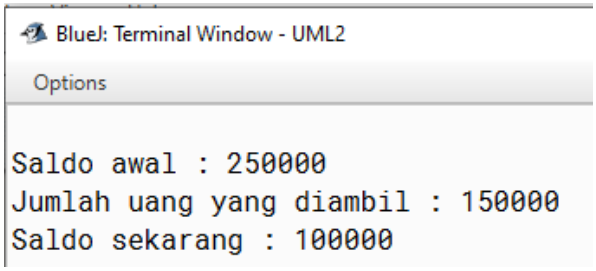
```
public class Tabungan
{
    public int saldo;
        public Tabungan(int initsaldo){
                saldo=initsaldo;
        }
        public void ambilUang(int jumlah){
                saldo-=jumlah;
        }
}
```

✓ **Class Cobatabungan**

Langkah selajutnya buatlah kelas baru juga dengan nama class CobaTabungan, class ini yang nanti akan digunakan untuk menguji programnya dan pada class ini terdapat saldo awal, jumlah uang yang diambil dan saldo sekarang. Untuk pemahaman supaya jelas dapat Anda lihat pada gambar berikut untuk listing programnya.

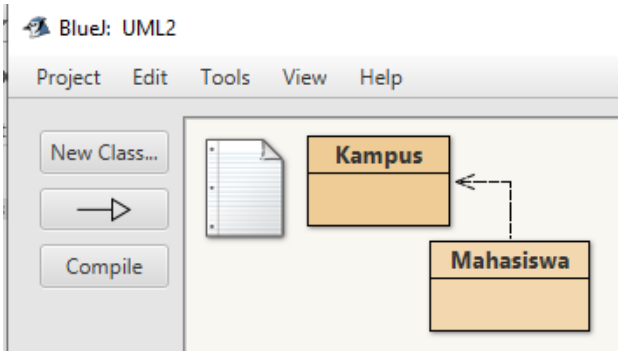
```
public class Cobatabungan
{
      public static void main(String args[]){
       System.out.println("\u000c");
       Tabungan tabungan=new Tabungan(250000);
       System.out.println("Saldo awal : " +tabungan.saldo);
       tabungan.ambilUang(150000);
       System.out.println("Jumlah uang yang diambil : 150000");
       System.out.println("Saldo sekarang : " +tabungan.saldo);
       }
}
```

✓ **Hasil Program Tersbut Adalah**



2. **Contoh Program kedua yaitu class Kampus dengan class Mahasiswa**

Pada contoh kedua ini akan menampilkan NIM dan nama Mahasiswa, yang terdapat pada kelas kampus dan class mahasiswa. Untuk strukturnya dapat dilihat pada gambar dibawah ini.



✓ **Class Kampus**

Buatlah kelas baru dengan nama class kampus sesuai dengan strukutur sebelumnya, pada class ini terdapat NPM dan Nama. Untuk selanjutnya tuliskan kode programnya seperti terlihat pada gambar yang tersedia.

```
public class Kampus
{
  public int nrp;
  public String nama;
  public Kampus(int i, String n){
        nrp=i;
        nama=n;
  }
  public int getNrp(){
        return nrp;
  }
  public String getNama(){
        return nama;
  }
}
```

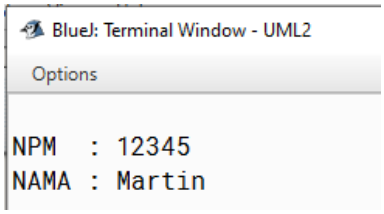
✓ **Class Mahasiswa**

Langkah selajutnya buatlah kelas yang laian dengan nama class mahasiswa sesuai dengan strukutur sebelumnya, pada class ini digunakan untuk menampilkan data data npm dan juga nama. Untuk selanjutnya tuliskan kode programnya seperti terlihat pada gambar yang tersedia.

```
public class Mahasiswa
{
     public static void main(String args[]){
         System.out.println("\u000c");
         Kampus kps=new Kampus(12345,"Martin");
         System.out.println("NPM  : "+kps.getNrp());
         System.out.println("NAMA : "+kps.getNama());
       }
}
```

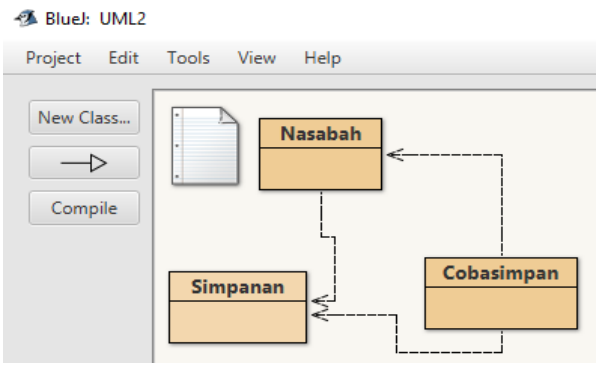
✓ **Hasil Program**

Hasil program dapat dilihat pada tampilan dibawah ini jika semua source codenya benar dan sudah tidak ada no syntax error



3. Program Class Nasabah , Class Simpanan dan Clas Cobasimpanan

Pada contoh yang lain yang ketiga ini hampir sama dengan contoh yang pertama terkait dengan tabungan, pada contoh yang ketiga ini terdapat class nasabah, class simpanan dan class cobasimpanan. Perhatikan struktur berikut ini biar jelas dan paham.



✓ **Class Nasabah**

Langkah pertam buatlah kelas baru dengan nama class nasabah, pada class ini dipergunakan untuk meyimpan nama nasabah dan saldo awal dari tabungan dari nasabah. Untuk detail programnya dapat dilihat pada tampilan dibawah.

```
public class Nasabah
{
        private String namaAwal;
        private String namaAkhir;
        private Simpanan simpanan;

        public Nasabah(String namaAwal, String namaAkhir){
                this.namaAwal=namaAwal;
                this.namaAkhir=namaAkhir;
        }
        public String getNamaAwal(){
                return namaAwal;
        }

        public String getNamaAkhir(){
                return namaAkhir;
        }

        public Simpanan getSimpanan(){
                return simpanan;
        }

        public void setSimpanan(Simpanan simpanan){
                this.simpanan=simpanan;
        }
}
```

✓ **Class Simpanan**

Langkah kedua adalah membuatlah kelas dengan nama class simpanan, pada class ini di pergunakan untuk menyimpan saldo awal dan saldo tambahan dari simpanan. Untuk dapat lebih memahami silahkan Anda lihat pada gambar berikut dan juga untuk listing programnya.

```
public class Simpanan
{
    private int saldo;

        public Simpanan(int saldo){
                this.saldo=saldo;
        }

        public int getSaldo(){
                return saldo;
        }

        public void simpanUang(int jumlah){
                saldo += jumlah;
        }

        public boolean ambilUang(int jumlah){
                if (jumlah > saldo)
                   return false;
                else
                   saldo-=jumlah;
                   return true;
        }
}
```

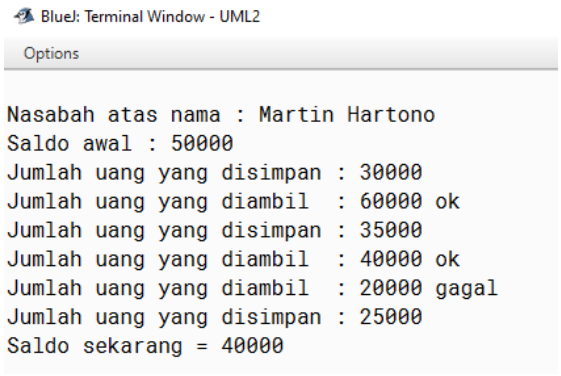
✓ **Class Cobasimpanan**

Langkah ketiga adalah dengan membuatlah kelas untuk ujicoba programnya dengan nama class cobasimpanan, pada class ini untuk menampilkan data nama nasabah, saldo awal, jumlah uang yang disimpan, ju,lah uang yang diambil dan pesan jika uang yang di ambil melebihi saldonya. Untuk dapat lebih memahami silahkan Anda lihat pada gambar berikut dan juga untuk listing programnya.

```
public class Cobasimpan
{
    public static void main(String args[]){
        System.out.println("\u000c");
        int tmp;
        boolean status;
        Nasabah nasabah=new Nasabah("Martin","Hartono");
        System.out.println("Nasabah atas nama : " +nasabah.getNamaAwal()+ " " + nasabah.getNamaAkhir());
        nasabah.setSimpanan(new Simpanan(50000));
        tmp=nasabah.getSimpanan().getSaldo();
        System.out.println("Saldo awal : "+ tmp);
        nasabah.getSimpanan().simpanUang(30000);
        System.out.println("Jumlah uang yang disimpan : 30000");
        status=nasabah.getSimpanan().ambilUang(60000);
        System.out.print("Jumlah uang yang diambil  : 60000");
           if (status)
              System.out.println(" ok");
           else
              System.out.println(" gagal");
        nasabah.getSimpanan().simpanUang(35000);
        System.out.println("Jumlah uang yang disimpan : 35000");
        status=nasabah.getSimpanan().ambilUang(40000);
        System.out.print("Jumlah uang yang diambil  : 40000");
        if (status)
           System.out.println(" ok");
        else
           System.out.println(" gagal");
        status=nasabah.getSimpanan().ambilUang(20000);
        System.out.print("Jumlah uang yang diambil  : 20000");
        if (status)
           System.out.println(" ok");
        else
           System.out.println(" gagal");
        nasabah.getSimpanan().simpanUang(25000);
        System.out.println("Jumlah uang yang disimpan : 25000");
        tmp=nasabah.getSimpanan().getSaldo();
        System.out.println("Saldo sekarang = " + tmp);
    }
```

✓ **Hasil Program Tersebut diatas**

Hasil program tersebut jika semua sudah di compile dan hasil compilenya tidak ada kesalahan sehingga akan menghasilkan tampilan program seperti contoh dibawah ini.


BlueJ: Terminal Window - UML2
Options
Nasabah atas nama : Martin Hartono
Saldo awal : 50000
Jumlah uang yang disimpan : 30000
Jumlah uang yang diambil : 60000 ok
Jumlah uang yang disimpan : 35000
Jumlah uang yang diambil : 40000 ok
Jumlah uang yang diambil : 20000 gagal
Jumlah uang yang disimpan : 25000
Saldo sekarang = 40000

BAB XIII
KONEKSI DATABASE

Aplikasi

Hal yang utama dari belajar pemrograman adalah bagaimana dapat mendesain dan membuat sebuah aplikasi yang interaktif, mudah untuk dioperasikan oleh user atau dengan istilah bahwa program tersebut adalah *user friendly*. Sebab aplikasi *user friendly* tentu mengacu pada aplikasi antarmuka pengguna yang menarik dan berbasis grafis *(GUI).* Aplikasi berbasis GUI, aplikasi basis GUI ini dari hari ke hari menjadi lebih menarik dari sisi interfacenya, tetapi juga dari feature dan teknologi yang terdapat pada aplikasi tersebut.

Grafis adalah kata kunci dalam sebuah aplikasi , grafis sendiri dapat di rujuk pada bentuknya garis, huruf dan simbol. Apliakasi berorientasi grafis saat ini sudah pasti dipakai hampir disemua perangkat baik komputer maupun perangkat bergerak (*mobile)*. Aplikasi dengan model ini memudahkan pasa user yang terlibat langsung pada aplikasi tersebut. Suatu Aplikasi itu dapat dikatakab baik atau lengakp bila di dukung dengan media penyimpanan data. Sebab segala sesua yang kita kerjakan tentu saja tidak bisa selesai saat itu. Maka perlu adanya penyimpan data tersebut.

Interface dengan model GUI ini pada proses pengurutan data ke dalam suatu database aplikasi menjadi sangat mudah untuk digunakan. Desain aplikasi harus bisa mendukung basisdata, hal ini mewajibkan bagi pengembang memahami mode desain antarmuka sekaligus desain struktur basisdata, karena pada dasarnya kedua bagian tadi adalah beda yang disatukan bersama. Selanjutnya supaya menjadi lebih baik maka perlu dicoba disatukan kedua bagian tersebut dalam suatu model sederhana. Sebagai pengingat, ada baiknya untuk sedikit membahas teori konsep basis data dengan struktur bahasa SQL.

Basisdata

Basis data terdiri dari dua kata yaitu basis dan data, “basis‟ artinya adalah dasar atau pangkalan. Data memiliki arti sesuatu yang masih mentah atau belum memiliki makna supaya memliki makna maka data tersebut perlu di oleh agar menajdi sebuah informasi. Basis data ini pada prinsipnya adalah sesuatu yang terpisah - pisah yang dikemas dalam satu wadah yang dinamakan tabel tabel dan dari tebel tabel tersebut saling berelasi satu dengan yang lainnya sehingga membentuk sutau kumpulan data terhubung dalam satu wadah yang biasa disebut dengan database.

Tools Basisdata

Desain Aplikasi basisdata adalah alat yang digunakan untuk membantu perancangan yang bersifat fisik atau struktur, suatu desain struktur tabel-tabel yang diperlukan untuk mendukung suatu hasil pengolahan data menjadi informasi. Untuk latihan di sini hanya mencoba menerapkan satu tabel saja sebagai dasar untuk mengkoneksikan saja. Untuk pengembangan kedepan silahkan bisa dicoba sendiri dengan menambahkan tabel tabel yang lain agar menjadi satu kesatuan yang kompleks.

Structure Query Language

Structure Query Language (SQL) adalah bahasa untuk mendesain struktur database. SQL adalah bahasa standar yang digunakan di hampir semua alat basis data yang menggunakan prinsip sistem manajemen basis data (DBMS). SQL secara umum dibagi menjadi tiga (3) kelompok berdasarkan

kelasnya, antara lain Data Definition Language (DDL), Data Manipulation Language (DML), dan Data Control Language (DCL)

- ✓ **Bahasa definisi data / Data Definition Language (DDL)**
  Data Definition Language (DDL) adalah kueri yang mendefinisikan basis data. Definisi yang dimaksud di sini adalah penamaan basis data, penamaan tabel, ukuran data, dll. Ada beberapa perintah query dalam DDL yaitu CREATE, DROP dan ALTER.
- ✓ **Bahasa manipulasi data / Data Manipulation Languge (DML)**
  Data Manipulation Language (DML) adalah query yang memanipulasi data. Manipulasi kata mengacu pada operasi yang terkait dengan penambahan, perubahan, dan penghapusan data dalam tabel. Untuk melakukan fungsi manipulasi datanya, DML memiliki beberapa perintah query seperti INSERT, DELETE, SELECT, UPDATE dll.
- ✓ **Bahasa manajemen informasi / Data Control Languge (DCL)**
  Ini adalah kueri yang memberi pengguna akses ke database dan tabel. Perintah khusus DCL adalah GRAND dan REVOKE

Perancangan Basisdata

Perancangan database pada prinsifnya diawali dari sebuah analisa kegunaan akan data. Analisa data tersebut dilaksanakan untuk menentukan kebutuhan database, relasi data dalam tabel, serta ukuran data disetiap tabelnya. Desain basisdata dimulai dari menjelaskan basisdata, selanjutnya struktur tabel-tersebut digunakan. Untuk mendesain basisdata dalam query sedikit berbeda dengan aplikasi lainya, berikut adalah cara untuk mendesain tabel tabel menggunakan query :

- ✓ **Mecinptakan basisdata, dengan perintah query**

```
CREATE DATABASE namaBasisdata;
```

- ✓ **Membuat tabel, dengan perintah query berikut**

```
CREATE TABLE table_name (
column1 datatype,
column2 datatype,
column3 datatype,
....
);
```

Untuk membuat primary key dalam query, dapat ditambahakan dengan menggunakan parameter primary key, dengan perintah berikut :

```
CREATE TABLE table_name (
column1 datatype,
column2 datatype,
column3 datatype, primary key(column1)
....
);
```

Memanipulasi Basisdata

Aplikasi sebuah sistem tidak lengkap tanpa adanya dukungan sebuah media penyimpanan. Data yang ada dapat inputkan atau di tambah, di edit , di proses melalui interface sebua aplikasi tersebut yang biasanya dinamakan formulir atau kotak isian. Kotak isian pada sebuah aplikasi tidak langsung dapat digunakan untuk memanipulasi data (tambah, edit, hapus), dibutuhkan beberapa langkah demi

langkah salah satunya adalah instalasi dan konfigurasi. Instalasi artinya mengistal tools yang dapat digunakan untuk mendukung aplikasi tersebut antara lain dapat menggunakan Ms. Access, MySql, dan lain sebagainya.
Selain itu, alat aplikasi harus dikonfigurasi menggunakan alat pendukung berbasis API, serta konfigurasi modul, biasanya berupa opcode yang ditambahkan ke API. Pengaturan form untuk aplikasi database dan tools harus menampilkan konfigurasi dalam bentuk perintah kode antara lain perintah untuk memanipulasi data (save, search, change dan delete). Perintah-perintah yang biasa digunakan untuk mengisi formulir pendaftaran dengan query SQL adalah sebagai berikut:

- ✓ **Add data, dengan perintah :**
  INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);

- ✓ **Search data, dengan perintah sebagai berikut :**
  Mencari data berdasarkan kolom dapat menggunakan perintah berikut :
  SELECT column1, column2, ...
  FROM table_name;

  Bisa juga dengan perintah seperti berikut jika ingin mencari data berdasarkan nama tabel *yang digunakan, yaikur :*

  *SELECT * FROM table_name;*

- ✓ **Upate data, dengan perintah query sebagai berikut :**
  UPDATE table_name
  SET column1 = value1, column2 = value2, ...
  WHERE condition;

- ✓ **Delete data**
  DELETE FROM table_name WHERE condition;

Konfigurasi Database
Form pembuatan database merupakan langkah penambahan dan penambahan class package library yang dibangun ke dalam file API. File API biasanya berupa file yang terdiri dari sekumpulan kelas yang bebas digunakan oleh pengguna, dalam aplikasi ini pengembang aplikasi. File API dikemas dalam file .jar, misalnya mysql-connector-java-5.1.24-bin.jar

Koneksi Basisdata
Pada umum dalam pembuata sebuah aplikasi tidak menjadi satu kesatuan tersediri akan tetap terpisah antaran aplikasi dan databasenya. Karena aplikasi yang terpisah tersebut sehingga perlua adanya hubungan atau koneksi basisdata dengan apliaksinya agar aplikasi tersebut menjadi lengakap dan untuh. Hubungan data adalah hubungan terpadu antara antarmuka pengguna dan aplikasi dan alat basis data yang digunakan sebagai media. Relasi atau koneksi database bertindak sebagai perantara yang memungkinkan pengguna untuk mengedit (menambah, mengubah, menghapus) informasi pada formulir pendaftaran.

Relasi atau koneksi data dapat menggunakan beberapa pustaka paket tergantung pada model koneksi yang digunakan. Metode yang umum adalah dengan menggunakan Java Database Connection Method (JDBC). Di mana jdbc adalah antarmuka API yang dipasang di driver mysql-java-connector. Model JDBC ini menggunakan beberapa class library termasuk import java.sql.Connection, import java.sql. Statement, import java.sql. Driver Manager dan class library paket lainnya yang diimpor ke dalam aplikasi saat program dikodekan. Paket ditambahkan ke baris pertama kode program.

Ada beberapa cara atau algoritme untuk menghubungkan antarmuka aplikasi ke alat basis data. Salah satu cara untuk terhubung ke database yang ada adalah dengan memperhatikan algoritma berikut:

- ✓ Konfigurasi driver mysql jdbc dengan cara klik kanan pada library di sebelah kiri lalu pilih Add library. Jika berhasil, driver jdbc mysql terdaftar pada aplikasi yang digunakan:
- ✓ Ketika driver jdbc mysql diidentifikasi, kita dapat membuat database sesuai dengan langkah-langkah yang ditunjukkan.
- ✓ Selesaikan proses dengan menekan interface test lalu pilih Next dan Finish. Selanjutnya segera buat database seperti pada langkah l

Praktikum Koneksi Databas

1. Contoh Koneksi ke Database phpMy Admin

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class koneksi {
   public static Connection con;
   public static Statement stm;
   public static void main(String args[]){
      try {
         String url ="jdbc:mysql://localhost/tutorial";
         String user="root";
         String pass="";
         Class.forName("com.mysql.jdbc.Driver");
         con =DriverManager.getConnection(url,user,pass);
         stm = con.createStatement();
         System.out.println("koneksi berhasil;");
      } catch (Exception e) {
         System.err.println("koneksi gagal" +e.getMessage());
      }   } }
```

2. Penerapan Koneksi ke dalam database model-1

```
public class Mahasiswa {
    private String npm;
    private String namamhs;
    private float ipk;
    private String progdi;

    public Mahasiswa(String npm, String namamhs, float ipk, String progdi) {
        this.npm = npm;
        this.namamhs = namamhs;
        this.ipk = ipk;
        this.progdi = progdi;
    }

    public String getNpm() {
        return npm;
    }

    public void setNpm(String npm) {
        this.npm = npm;
    }

public String getNamamhs() {
    return namamhs;
}

public void setNamamhs(String namamhs) {
    this.namamhs = namamhs;
}

public float getIpk() {
    return ipk;
}

public void setIpk(float ipk) {
    this.ipk = ipk;
}

public String getProgdi() {
    return progdi;
}
```

```
    public void setProgdi (String progdi) {
        this.progdi = progdi;
    }

    public String toString() {
        return "Mahasiswa{" + "npm=" + npm + ", namamhs=" + namamhs + ", ipk=" + ipk + ",
progdi=" + progdi + '}';
    }
}
```

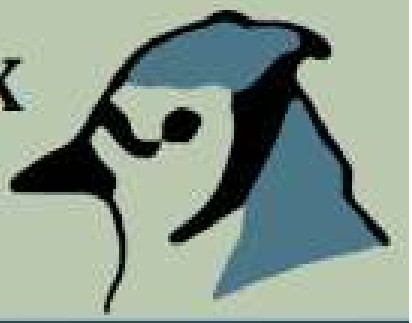
3. Penerapan Koneksi ke dalam database model-1

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class test_koneksi {
    public static Connection con;
    public static Statement stm;
    public static void main (String args[]){

        try{
            String url="";//ip database server + ("pake slash / ") nama database ex :
jdbc:mysql://localhost/db_test_koneksi
            String user="";//isi sesuai dengan user yang di pakai ex :root
            String pass="";//isi sesuai dengan passoword usernya ex : qwerty123 atau , jika kosong
passwordnya ; tidak usah di isi
            Class.forName("com.mysql.jdbc.Driver");
            con =DriverManager.getConnection(url,user,pass);
            stm = con.createStatement();
            System.out.println("koneksi berhasil : ");
        }
        catch (Exception e){

            System.err.println("koneksi gagal : "+e.getMessage()); }
    }
}
```

DAFTAR PUSTAKA

1. Java dengan BlueJ,Ron McFadyen dan Jeanette Bautista, Department of Applied Computer Science, Universitas Winnipeg, Kanada, 2019
2. An Introduction to Object-Oriented Programming with Java™, fifth edition, C Thomas Wu, McGraw-Hill, New York America, 2006
3. Java Untuk Pemula, Jubilee Enterprise, Elek Media Komputindo, Jakarta, 2014
4. ICSE Java Complete Reference, Mr, Swapnil Karmalkar, Bluej Group, University Of Kent, 2012
5. Teori dan Implementasi JAVA, Rismon Hasiholan Sianipar, Informatika , Bandung, 2013
6. Learning Object Oriented Programing Using Java, Vikas Publishing, New Delhi, S Chand and Company Limeted, 2010
7. Pemrograman Java dari Nol, Tim EMS, PT. Elexmedia Komputindo Jakarta, 2015
8. Tutorial Pemrograman Java Untuk Pemula, Vivian S & RH Sianipar, Balige Publishing, Sumatra Utara

PEMROGRAMAN BERORIENTASI OBJEK dengan JAVA BlueJ



Oleh : Budi Hartono. M.Kom

Budi Hartono, S.Kom, M.Kom Lahir di Sleman pada Tanggal 19 September 1968, dan sekarang menetap di Semarang. Menyelesaikan Pendidikan Dasar hingga SLTA di Sleman, Yogyakarta. Menempuh pendidikan Sarjana dari Program Studi S1-Sistem Komputer Universitas Sains dan Teknologi Komputer (Universitas STEKOM) Lulus Tahun 2004, Program Magister diselesaikan pada Tahun 2006 pada Program Magister Teknik Informatika Universitas Dian Nuswantoro.

Saat ini penulis tercatat aktif sebagai Dosen Tetap di Universitas Sains dan Teknologi Komputer (Universitas STEKOM) pada Program Studi S1 Teknik Informatika. Untuk matakuliah yang di ampu antara laian adalah Logika dan Algoritma Pemrograman, Pemrograman Visual, Pemrograman Berorientasi Objek, Pengantar Bahasa Query, Sistem Pendukung Keputusan, Sistem Basis Data, Struktur Data, Analisa dan Perancangan Sistem Informasi.

Buku Pemrograman Berorientasi Objek ini adalah hasil karyanya pada tahun 2022. Adapun buku buku karya penulis yang sudah terbit antara lain Pemrograman WAP, Pemrograman Visual dengan Delphi, Sistem Pendukung Keputusan dan Cara Mudah dan Cepat Belajar Pengembangan Sistem Informasi.

ISBN 978-623-8120-21-5 (PDF)

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