

Modul Praktikum

Pemrograman Berorientasi Objek (Object Oriented Programming) dengan JAVA



OKKITA RIZAN



Sekolah Tinggi Manajemen Informatika dan Komputer

ATMA LUHUR

Tahun Ajaran 2015/2016
Semester Genap

Bahasa pemrograman telah banyak disediakan untuk membuat sebuah aplikasi, baik yang berskala kecil atau berskala besar guna menjawab berbagai kebutuhan pengguna. Dalam membangun aplikasi, dikenal ada dua teknik atau metode dalam pembuatan program, yaitu metode pemrograman terstruktur dan metode pemrograman berorientasi objek. Kedua teknik ini secara tidak langsung menunjukkan ciri khas bahasa pemrograman yang digunakan. Pada modul praktikum ini, secara khusus akan membahas bahasa pemrograman yang bersifat berorientasi objek. Ada banyak bahasa pemrograman yang berciri khas berorientasi objek. Bahasa Pemrograman yang akan diperkenalkan adalah Bahasa Pemrograman Java dan kemudian dalam modul praktikum ini disebut dengan Bahasa Java.

Modul Praktikum ini disusun untuk membantu dan memudahkan mahasiswa STMIK Atma Luhur Pangkalpinang khususnya program studi Sistem Informasi dan Teknik Informatika dalam mengikuti mata kuliah Pemrograman Berorientasi Objek, tetapi tidak menutup kemungkinan bagi pembaca yang lain untuk menjadikan modul praktikum ini sebagai salah satu referensi didalam mempelajari program yang berorientasi objek.

Modul Praktikum ini lebih ditujukan kepada para programmer tingkat pemula. Modul Praktikum ini berisi pengenalan dan dasar pemrograman berorientasi objek diantaranya konsep object oriented programming sampai dengan proses membuat program berorientasi objek berbasis GUI.

Dikarenakan pemrograman tidak terlepas dari algoritma, diharapkan para pembaca sudah terlebih dahulu menguasai konsep algoritma dan bagaimana implementasinya didalam program. Dilingkungan STMIK Atma Luhur, algoritma sudah diajarkan pada mata kuliah Algoritma dan Struktur Data.

Tak lupa penulis menyampaikan banyak terima kasih kepada seluruh blogger di dunia maya yang telah bersedia membagikan ilmunya untuk penulis bagikan kembali keseluruhan para pembaca melalui modul praktikum ini. Tak lupa juga penulis menyampaikan terima kasih kepada para pengembang software JCreator, Netbeans dan JDK yang telah bersedia membagikan software ini secara gratis (free software).

Ditengah kemampuan modul praktikum ini yang sangat terbatas, semoga sedikit uraian ilmu yang terkandung didalam modul praktikum ini dapat bermanfaat bagi para pembacanya, untuk mempelajari dan mempraktekkan program dengan konsep berorientasi objek.

Pangkalpinang, Maret 2016

Okkita Rizan

**PRAKATA****DAFTAR ISI****MODUL 1 – PENDAHULUAN**

1. Evolusi Teknik Pemrograman	1
2. Fundamental OOP	4
3. Bahasa Pemrograman Java	5
4. Aplikasi dan Library Bahasa Pemrograman Java	6

MODUL 2 – PENGENALAN APLIKASI DAN STRUKTUR JAVA

1. Aplikasi JCreator	7
2. Class Pada Java	8
3. Membuat Class dengan main() method	9

MODUL 3 – ALGORITMA DENGAN JAVA

1. Tipe Data	13
2. Variabel	14
3. Operator	16
4. Memahami Class String	22
5. Memahami Class Untuk Input Data pada Java	30
6. Struktur Pencabangan	33
7. Struktur Perulangan	40
8. Array	43

MODUL 4 – FUNDAMENTAL OOP - KONSEP CLASS & DESAIN CLASS

1. Konsep Class dan Object	45
2. Class Diagram	45
3. Class Diagram Menjadi Class Pada Java	47

MODUL 5 – IMPLEMENTASI OBJEK, ATRIBUT & METHOD PADA JAVA

1. Instansiasi Objek	50
2. Penggunaan Method Melalui Sebuah Objek	51
3. Static Method	53
4. Penggunaan Atribut Melalui Sebuah Objek	54

MODUL 6 – RELATIONSHIP PADA OBJECT ORIENTED

1. Memahami Asosiasi	58
2. Penerapan Asosiasi Pada Java	60
3. Memahami Agregasi	62
4. Penerapan Agregasi Pada Java	62

MODUL 7 – FUNDAMENTAL OOP - ENKAPSULASI

- 1. Memahami Enkapsulasi 64
- 2. Penerapan Enkapsulasi Pada Java 65

MODUL 8 – KONSTRUKTOR & DESTRUKTOR

- 1. Memahami Konstruktor 67
- 2. Penerapan Konstruktor Pada Java 68
- 3. Memahami Destruktor 72

MODUL 9 – FUNDAMENTAL OOP - INHERITANCE

- 1. Memahami Inheritance 75
- 2. Penerapan Inheritance Pada Java 77
- 3. Macam – Macam Turunan 78

MODUL 10 – FUNDAMENTAL OOP - POLYMORPHISM

- 1. Memahami Polymorphism 84
- 2. Memahami Overloading 85
- 3. Penerapan Overloading Pada Java 85
- 4. Memahami Overriding 87
- 5. Penerapan Overriding 87

MODUL 11 – INTERFACE

- 1. Memahami Interface 89
- 2. Penerapan Interface Pada Java 90

MODUL 12 – RAGAM CLASS

- 1. Super Class 91
- 2. Sub Class 91
- 3. Abstract Class 91
- 4. Nested Class 94
- 5. Final Class 102

MODUL 13 – CLASS JFrame

- 1. Memahami Class JFrame 103
- 2. Penerapan Method Class JFrame Dalam Pemrograman 104
- 3. Penerapan Konsep Turunan Pada Class JFrame 110

MODUL 14 – CLASS GUI

- 1. Pengenalan Package AWT Dan SWING 112
- 2. Class GUI Pada Package SWING 113
- 3. Penerapan Class GUI Pada Pemrograman 116

MODUL 15 – CLASS LAYOUT MANAGER

1. Memahami Class Layout Manager 118
2. Penerapan Class FlowLayout Pada Pemrograman 119
3. Penerapan Class GridLayout Pada Pemrograman 121
4. Penerapan Class BorderLayout Pada Pemrograman 124

MODUL 16 – CLASS JPANEL

1. Memahami Class JPanel 127
2. Penerapan Class JPanel Pada Pemrograman 128

MODUL 17 – INTERFACE LISTENER

1. Memahami Interface Listener 130
2. Penerapan Interface ActionListener Pada Pemrograman 132
3. Penerapan Interface MouseListener Pada Pemrograman 135
4. Penerapan Interface MouseMotionListener Pada Pemrograman 138
5. Penerapan Interface KeyListener Pada Pemrograman 140

DAFTAR PUSTAKA

• PENDAHULUAN

TUJUAN :

- Dapat memahami perkembangan / evolusi model pemrograman
- Dapat memahami prinsip-prinsip pemrograman berorientasi objek
- Dapat memahami perkembangan bahasa pemrograman java
- Dapat memahami tools membuat program menggunakan java

1. EVOLUSI TEKNIK PEMROGRAMAN

Programmer sudah menulis bahasa pemrograman modern sejak tahun 1940. Bahasa pemrograman terdahulu mengharuskan programmer untuk bekerja dengan alamat memory dan membuat kode program yang berhubungan dengan bahasa mesin. Bahasa pemrograman yang baru lebih mudah digunakan dengan beberapa alasan diantaranya :

- Baris perintah menggunakan bahasa yang lebih mudah dimengerti. Sebagai contoh : perintah untuk menampilkan keluaran, menggunakan kata kerja yang umum diantaranya seperti output, write atau print.
- Memungkinkan programmer untuk menggunakan nama dari lokasi memory daripada alamat memory aslinya. Sebagai contoh, sebuah data yang akan disimpan pada memory komputer, programmer dapat menunjukkan lokasi memory komputer menggunakan notasi heksadesimal daripada menggunakan alamat sesungguhnya yang membutuhkan digit yang panjang.
- Memungkinkan untuk membuat modul tersendiri dan dapat disatukan dengan modul yang lain dengan berbagai cara. Bahasa pemrograman terdahulu mengharuskan program dibuat dalam satu bagian saja, dari mulai sampai dengan akhir. Menulis baris perintah dalam beberapa modul kecil lebih mudah harus menulis satu bagian baris perintah yang banyak.

Pada saat sekarang ini, programmer menggunakan salah satu dari dua teknik utama dalam mengembangkan/membuat program/aplikasi yaitu :

a. Procedural Programming / Pemrograman Terstruktur

Pemrograman menggunakan teknik ini berfokus pada bagaimana programmer memanipulasi data. Teknik pemrograman ini mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dalam bentuk program. Selain itu teknik pemrograman ini dapat juga diartikan sebagai suatu aktifitas pemrograman dengan memperhatikan urutan langkah-langkah perintah secara sistematis, logis dan tersusun berdasarkan algoritma yang sederhana dan mudah dipahami. Prinsip dari teknik pemrograman ini adalah jika suatu proses telah sampai pada suatu titik/langkah tertentu, maka proses selanjutnya tidak boleh mengeksekusi langkah sebelumnya / kembali lagi ke baris sebelumnya, kecuali pada langkah-langkah untuk proses perulangan (looping).

Sifat-sifat dari teknik pemrograman terstruktur diantaranya :

- Memuat teknik pemecahan masalah yang logis dan sistematis
- Membuat algoritma yang efisien, efektif dan sederhana
- Program disusun dengan logika yang mudah dipahami
- Tidak menggunakan perintah GOTO
- Biaya pengujian program relatif rendah
- Memiliki dokumentasi yang baik
- Biaya perawatan dan dokumentasi yang dibutuhkan relatif rendah.

Adapun beberapa bahasa pemrograman yang mendukung teknik pemrograman terstruktur diantaranya Cobol, Turbo Prolog, C, Pascal, Delphi, Borland Delphi.

b. Object Oriented Programming / Pemrograman Berorientasi Objek

Pemrograman menggunakan teknik ini berfokus pada objek atau berorientasikan kepada objek. Semua data dan fungsi pada teknik ini dibungkus dalam kelas-kelas atau objek-objek. Bahasa pemrograman yang mendukung Object Oriented ini diantaranya Visual Foxpro, Java, C++, Pascal, Visual Basic.NET, SIMULA, Smalltalk, Ruby, Python, PHP, C#, Delphi, Eiffel, Perl dan Adobe Flash AS 3.0.

Perbedaan mendasar antara teknik pemrograman tersebut dengan pemrograman berorientasi objek yaitu pada Teknik pemrograman berorientasi objek semua data dan fungsi dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data dan mengirim pesan ke objek lainnya. Sedangkan untuk teknik pemrograman terstruktur adalah sebuah cara pemrosesan data yang terstruktur dalam analisa, cara dan penulisan pemrograman, dikarenakan harus terstruktur sehingga dalam pembuatannya antara satu line pemrograman dengan yang lainnya berhubungan.

Konsep utama dari teknik pemrograman beorientasi objek terletak pada kondisi kode / baris pemrogramannya dimana merupakan sebuah kesatuan modular. Untuk program yang simpel atau sederhana, biasanya menggunakan pemrograman terstruktur karena masih mudah dan tidak banyak dilakukan perubahan yang berarti, sedangkan untuk line lebih dari 100 atau bisa dikatakan rumit, maka digunakan pemrograman berorientasi objek. Pemrograman terstruktur terdiri dari pemecahan masalah yang besar menjadi masalah yang lebih kecil dan seterusnya. Sedangkan pemrograman breorientasi objek terdiri dari pengelompokan kode dengan data yang mana setiap objek berfungsi secara independen sehingga untuk setiap perubahan kode tidak tergantung pada kode yang lainnya, atau lebih dikenal dengan modular. Perbedaan lain antara teknik terstruktur dan objek oriented adalah pada kelas dan objek. Pada pemrograman terstruktur tidak terdapat kelas dan objek.

Sehingga dapat disimpulkan bahwa teknik pemrograman terstruktur lebih unggul dalam melakukan pemrograman sederhana karena lebih efisien dan lebih mudah

perawatannya tetapi pemodelan ini lebih susah untuk dipahami oleh orang-orang selain pembuat program itu sendiri, misalnya pada saat tracing program.

Berikut adalah rangkuman beberapa perbedaan mendasar antara Teknik Pemrograman Terstruktur dan Teknik Pemrograman Berorientasi Objek.

- Teknik pemrograman berorientasi objek, bentuk pemodelan programnya diorientasikan dalam bentuk objek-objek, sedangkan teknik pemrograman terstruktur pemodelan programnya diuraikan dan diorganisasikan secara lebih detail.
- Konsep dasar pemrograman berorientasi objek dikelompokkan kedalam kelas, objek, abstraksi, enkapsulasi dan polimorfisme melalui pengiriman pesan, sedangkan konsep dasar pemrograman terstruktur harus mengandung teknik pemecahan masalah yang tepat dan benar, memiliki algoritma pemecahan masalah yang sederhana, standar dan efektif, penulisan program memiliki struktur logika yang benar dan mudah dipahami, program hanya memiliki 3 struktur dasar, yaitu : Struktur berurutan, struktur seleksi, dan struktur perulangan. Menghindari penggunaan pernyataan GOTO. Memiliki dokumentasi yang baik.
- UML (Unified Modelling Language) adalah salah satu contoh model perancangan sistem yang berkonsep Pemrograman Berorientasi Objek. Sedangkan DFD (Data Flow Diagram) dan ERD (Entity Relationship Diagram) adalah contoh model perancangan sistem yang berkonsep Pemrograman Terstruktur.
- Dengan menggunakan teknik pemrograman berorientasi objek, dalam melakukan pemecahan suatu masalah tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut, sedangkan untuk teknik pemrograman terstruktur, menggunakan prosedur/tata cara yang teratur untuk mengoperasikan data struktur.
- Untuk tata nama, keduanya memiliki tatanan yang sama walaupun memiliki pengertian tersendiri. Object Oriented menggunakan "methode" sedangkan terstruktur menggunakan "function". Pada Object Oriented sering didengar mengenai "objects" sedangkan di terstruktur dikenal dengan "modules". Pada Object Oriented dikenal dengan "message" sedangkan pada terstruktur dikenal "argument". Pada teknik pemrograman berorientasi objek dikenal dengan "atribute" sedangkan pada teknik pemrograman terstruktur dikenal dengan istilah "variabel".

2. FUNDAMENTAL BAHASA PEMROGRAMAN BERORIENTASI OBJEK (OOP)

Object Oriented Programming (OOP) adalah sebuah bahasa pemrograman yang memandang segala sesuatu menjadi sebuah objek. Paradigma dari OOP adalah menyelesaikan masalah dengan merepresentasikan masalah ke model objek.

Karakteristik dari Object Oriented Programming adalah :

- Class dan Objek

Class dapat dikatakan sebagai cetak biru (blue print) dari sebuah objek ataupun beberapa objek. Class diartikan juga menggambarkan ciri-ciri objek secara umum. Sebagai contoh nokia, samsung, blackberry dan motorola merupakan objek dari class handphone. Nokia dan objek yang lainnya mempunyai atribut (apa yang dimiliki oleh objek / what they have) yang sama diantaranya memiliki tipe, tahun rilis, dimensi, berat, fitur dan lain sebagainya. Selain atribut, objek-objek tersebut memiliki metode (apa yang bisa dikerjakan objek / what they do) yang sama diantaranya kunci tombol, menampilkan teks dilayar, kirim sms dan lain sebagainya.

- Encapsulation / Enkapsulasi

Disebut juga dengan pembungkusan, yaitu melindungi program dan data yang sedang diolah agar tidak diakses secara sembarangan oleh program lain. Dalam java, dasar enkapsulasi adalah class. Variabel atau metode pada sebuah class tidak dapat diakses oleh class lain dengan menjadikan class tersebut bersifat **private**, atau menjadikan class tersebut bersifat **protected**, yang hanya bisa diakses oleh turunannya (inheritance) atau menjadikan class tersebut bersifat **public**, sehingga bisa diakses oleh sembarang class.

- Inheritance / Inheritansi

Disebut juga dengan turunan. Prinsipnya adalah sebuah class dapat diturunkan dari class yang lain. Class yang menurunkan ke class lain disebut dengan superclass, parent class atau base class atau kelas induk, sedangkan class yang merupakan turunan disebut sebagai subclass, child class atau derived class atau class turunan. Class turunan secara otomatis memiliki sifat (variable) dan kelakuan (behavior, method) yang dimiliki oleh super class-nya. Class turunan dapat menambahkan fitur atau behavior dengan mendefinisikan suatu metode di dalam class turunan tersebut.

- Polimorfisme / Polimorfisme

Polimorfisme secara bahasa dapat diartikan dengan memiliki banyak bentuk. Kegunaan dari polimorfisme adalah agar dapat mendefinisikan beberapa konstruktor atau metode dengan karakteristik yang berbeda-beda agar nantinya dapat digunakan untuk kasus-kasus yang berbeda. Method atau perilaku yang sama tapi implementasinya/caranya yang berbeda-beda.

3. BAHASA PEMROGRAMAN JAVA

Seperti yang sudah disebutkan diatas Bahasa Pemrograman Java merupakan salah satu dari bahasa pemrograman yang berorientasi objek. Mudahnya, berbicara mengenai bahasa pemrograman adalah sebuah bahasa yang berisi baris-baris perintah dan menghasilkan sebuah program. Ada banyak bahasa pemrograman, mulai dari bahasa pemrograman tingkat rendah sampai dengan bahasa pemrograman` tingkat tinggi. Java adalah bahasa pemrograman yang tergolong ke dalam bahasa pemrograman tingkat tinggi karena lebih mudah di operasikan.

Sama seperti bahasa yang lain, java juga memiliki sintaks, tata bahasa dan aturan penulisan sendiri.

Java lahir pada tahun 1991, diciptakan oleh sebuah tim yang bernama green dan berjalan selama 18 bulan. Green dimotori oleh James Gosling, Mike Sheridan, Bill Joy dan Patrick Naughton beserta 9 programmer lainnya. Mereka adalah insinyur dari Sun Microsystem. Awalnya bahasa Java di beri dengan nama "oak", yaitu nama dari sebuah pohon yang tumbuh di depan jendela ruang kerja James Gosling. Tetapi, nama tersebut tidak digunakan karena sebelumnya sudah ada perangkat lunak yang di rilis dengan nama "oak". Akhirnya nama "oak" diganti dengan nama "java", nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Pada tahun 1996, java versi pertama dari java di keluarkan dengan nama rilis java 1.02.

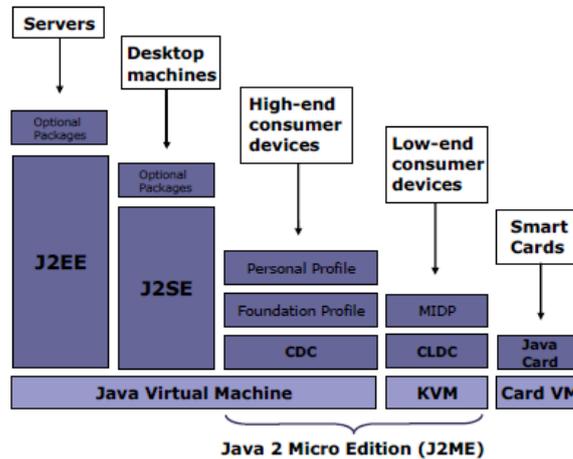
Beberapa fitur yang dimiliki java diantaranya adalah :

- a. Java Virtual Machine / Cross Platform
- b. Garbage Collection
- c. Code Security

Dengan keluarnya versi 1.2, platform Java telah dipilah-pilah menjadi beberapa edisi yaitu The Standard Edition(J2SE), Enterprise Edition(J2EE), Mobile Edition(J2ME), dan JavaCard API. Masing-masing edisi berisi Java 2 Software Development Kit (J2SDK) untuk mengembangkan aplikasi dan Java 2 Runtime Environment (J2RE) untuk menjalankan aplikasi. Adapun kegunaan dari masing-masing edisi ini ditunjukkan pada tabel berikut.

J2SE – Java 2 Platform, Standard Edition	Aplikasi Desktop
J2EE – Java 2 Platform, Enterprise Edition	Aplikasi enterprise dengan arsitektur pada pengembangan sisi webserver, termasuk servlet, JSP, EJB, dan XML
J2ME – Java 2 Platform, Micro Edition	Perangkat Mobile
JavaCard	Smart Cards

Adapun platformnya yang digambarkan pada JENI (Java Education Network Indonesia) adalah sebagai berikut :



4. APLIKASI DAN LIBRARY BAHASA PEMROGRAMAN JAVA

Ada beberapa aplikasi (IDE – Integrated Development Environment) yang dapat digunakan untuk mengimplementasikan Bahasa Java, diantaranya adalah sebagai berikut :

- a. Notepad
- b. Textplus
- c. Editplus
- d. Jcreator
- e. Netbeans.
- f. Crimson Editor
- g. Eclipse
- h. BlueJ
- i. Dr.Java
- j. JDeveloper

Dalam modul praktikum ini, untuk membuat program Java, aplikasi yang akan digunakan untuk mengimplementasikannya adalah Jcreator Pro 4.5 yang akan diuraikan lebih lanjut pada pembahasan berikutnya.

Selain aplikasi diatas, dalam membuat program java, juga diperlukan aplikasi pendukung yaitu JDK dan JRE.

JDK (Java Development Kit) digunakan untuk mengakses pustaka/library atau data yang dimiliki oleh java, selain itu JDK juga diperlukan untuk mengubah kode yang ditulis dengan bahasa Java (bahasa yang dimengerti manusia) menjadi byte code (bahasa yang dimengerti variable), proses ini dikenal dengan istilah compile atau kompilasi. Setelah kode di compile, program java siap untuk dijalankan di ariable (execute/eksekusi).

JRE (Java Runtime Environment) adalah aplikasi yang digunakan untuk menjalankan program yang dibuat dengan java, dalam hal ini program yang sudah di kompilasi.

Perbedaan dari kedua aplikasi tambahan ini adalah JDK digunakan untuk membuat program, sedangkan JRE digunakan untuk menjalankan program. Jadi kalau hanya ingin menjalankan program, maka hanya perlu menggunakan aplikasi JRE. Tetapi, pada saat penginstalan JDK di komputer, secara otomatis JRE pun akan ikut terinstall di komputer.

Pengenalan Aplikasi Dan Struktur Java

TUJUAN :

- Dapat memahami dan menggunakan aplikasi java
- Dapat memahami struktur java
- Dapat membuat program menggunakan aplikasi java

1. APLIKASI JCREATOR

a. Tentang Jcreator

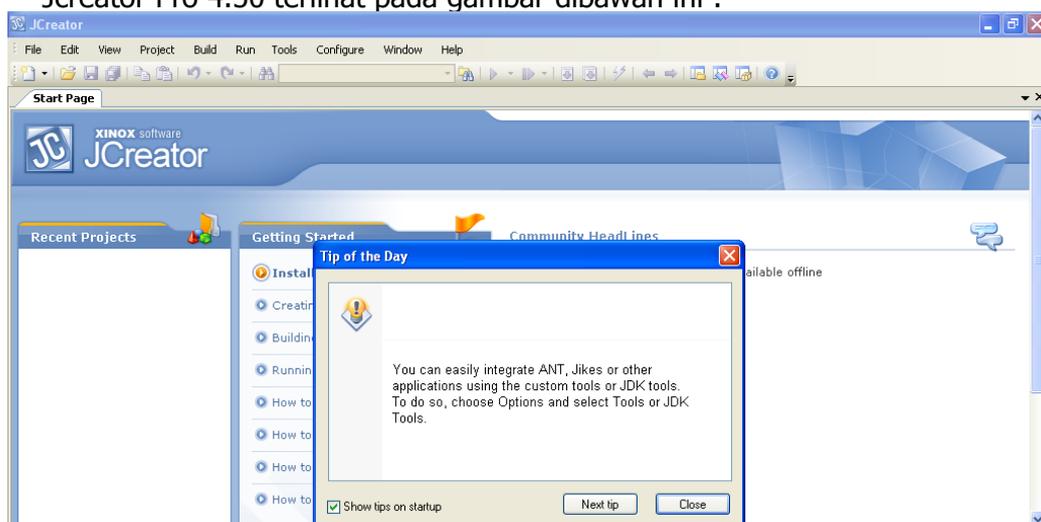
Aplikasi Jcreator adalah aplikasi untuk membuat program Java yang dibuat oleh Xinox Software. Jcreator ditulis dalam bahasa C/C++ sehingga lebih cepat (dan menggunakan memori lebih sedikit) dari kebanyakan IDE yang lain. Beberapa versi dari Jcreator adalah Jcreator 2.50, Jcreator 3.0, Jcreator 3.50, Jcreator 4.00 dan Jcreator 4.50. Modul Praktikum ini digunakan Jcreator versi 4.50, lebih tepatnya Jcreator 4.50.010 yang dirilis pada 10 Januari 2008.

b. Menjalankan Aplikasi Jcreator

Untuk menjalankan aplikasi Jcreator, dapat dilakukan paling tidak dengan dua cara yaitu :

- Klik start (), kemudian pilih All Programs (), dilanjutkan dengan memilih Jcreator Pro (), langkah terakhir mengklik Jcreator Pro 4.50 ().
- Klik ganda shortcut Jcreator Pro 4.50 yang berada di desktop ().

Setelah salah satu dari dua langkah diatas dilaksanakan, maka tampilan awal dari Jcreator Pro 4.50 terlihat pada gambar dibawah ini :



Klik Tombol **Close** kotak dialog **Tip of the Day**.

2. CLASS PADA JAVA

a. Mengetahui class dengan main () method

Pada bahasa pemrograman yang betul-betul berparadigma object oriented, seperti java, setiap baris perintah / statement yang dibuat harus menjadi bagian dari class. Pada beberapa bahasa pemrograman yang lain, seperti C++, baris perintah program dapat ditulis mengandung object dari class walaupun tanpa didalam class itu sendiri.

Dalam sebuah class, seluruh aksi ataupun baris perintah harus ditempatkan didalam sebuah method. Sebuah method merupakan kumpulan dari baris perintah yang mengerjakan beberapa atau kelompok dari tugas-tugas. Kebanyakan class mempunyai method walaupun tidak diharuskan setiap class mempunyai method. Class aplikasi paling tidak harus memiliki satu method yang disebut dengan **main method**. Pada beberapa bahasa pemrograman nama dari main method nya adalah **main()**, yang menunjukkan method tersebut adalah main method, seperti java dan c++. Apabila sebuah class mengandung main(), maka class tersebut adalah class aplikasi. Apabila dalam sebuah aplikasi terdapat banyak method, maka class yang mempunyai method main() akan dijalankan/dieksekusi pertama sekali.

Gambaran dari main method dari sebuah class adalah sebagai berikut :

```
class nama_kelas
    main ( )
    statements
end class
```

b. Struktur main () method pada java

Setiap bahasa pemrograman memiliki struktur bahasa sendiri yang menjadi ciri khasnya. Setiap bahasa pemrograman juga memiliki aturan sendiri yang bisa saja berbeda ataupun sama dengan aturan bahasa pemrograman yang lain.

Salah satu ciri dan konsep pokok dari pemrograman berorientasi objek yang juga dimiliki oleh bahasa pemrograman java adalah kelas (class). Dalam pembuatan program dengan java, secara tidak langsung juga pendefinisian atau pembentukan kelas. Kelas merupakan unit pembentuk program. Seluruh pembentuk program diwadahi dan ditampung di suatu kelas. Java dapat terdiri dari banyak kelas, dan sebuah kelas dapat terdiri dari banyak method. Namun biasanya sebuah file hanya terdiri dari satu kelas yang disimpan dengan nama kelas itu sendiri. Aturan untuk nama kelas di java sangat umum yaitu nama harus dimulai huruf, setelah itu boleh kombinasi huruf dan angka.

Struktur utama Bahasa Java adalah sebagai berikut :

```
public class namakelas
{
    public static void main ( String [ ] args )
    {
        statement ;
        statement ;
        statement ;
    }
}
```

Fungsi main() harus ditetapkan sebagai berikut :

- public berarti metode dapat dipanggil dari manapun di dalam dan di luar kelas
- static berarti adalah sama untuk seluruh instant dari kelas
- void berarti metode tidak mengirim apapun setelah selesainya.
- main berarti metode awal yang dijalankan

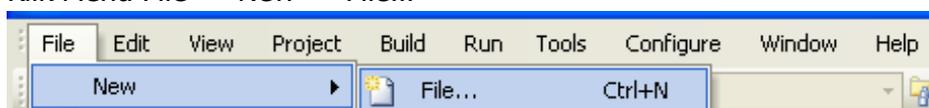
Java juga merupakan bahasa pemrograman yang bersifat case sensitive, yang berarti penulisan menggunakan huruf kapital ataupun huruf kecil pada kode program dapat berarti lain. Pasangan kurung kurawal menandakan awal dan akhir kumpulan dari instruksi-instruksi yang ada di dalam metode. Setiap instruksi atau baris-baris perintah harus diakhiri dengan titik koma untuk menandakan akhir dari satu baris perintah. Walaupun baris perintah ditulis lebih dari satu baris, jika tidak diakhiri dengan titik koma, maka Java akan menganggap baris-baris perintah tersebut merupakan satu baris perintah saja.

3. MEMBUAT CLASS DENGAN MAIN () METHOD

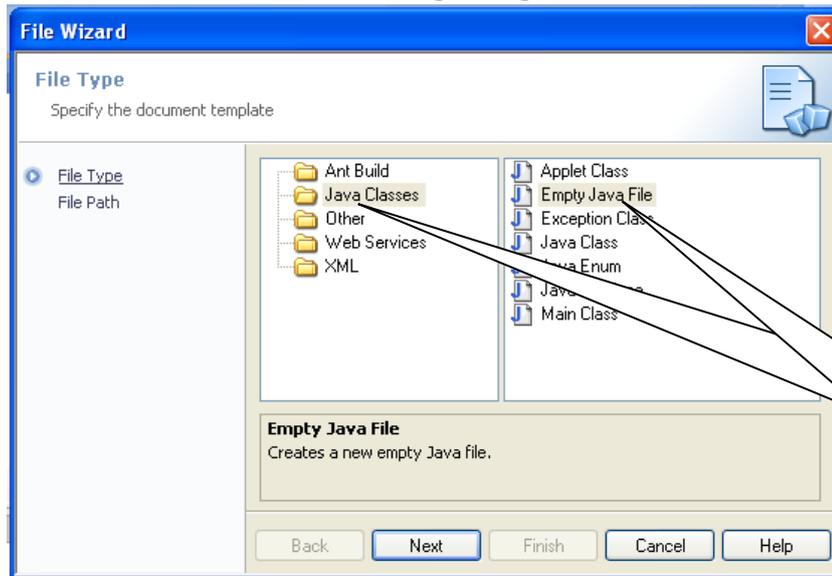
Dari banyak buku yang membahas mengenai bahasa pemrograman, selalu saja dimulai dengan membuat sebuah program yang sederhana sebagai pengantar untuk membahas area yang lebih spesifik. Program sederhana yang dibuat diantaranya menampilkan output berupa teks **"Hello World"** atau **"Hello Dunia"**. Perlu diperhatikan, untuk membuat program dengan Java, **NAMA FILE** dari program harus sama dengan **NAMA CLASS** yang mempunyai **MAIN() METHOD**. Apabila dalam satu program terdapat banyak class, maka **NAMA FILE** harus sama dengan **NAMA CLASS** yang mempunyai **MAIN() METHOD**.

Untuk memulai membuat sebuah program langkah-langkahnya sebagai berikut :

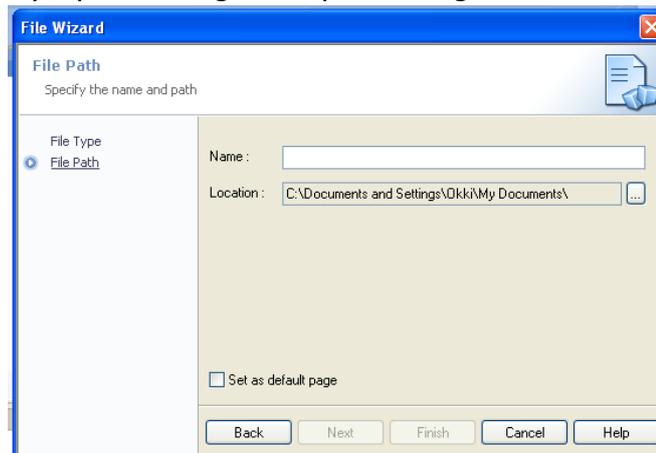
- Klik Menu File -> New -> File...



- Setelah diklik muncul kotak dialog sebagai berikut :



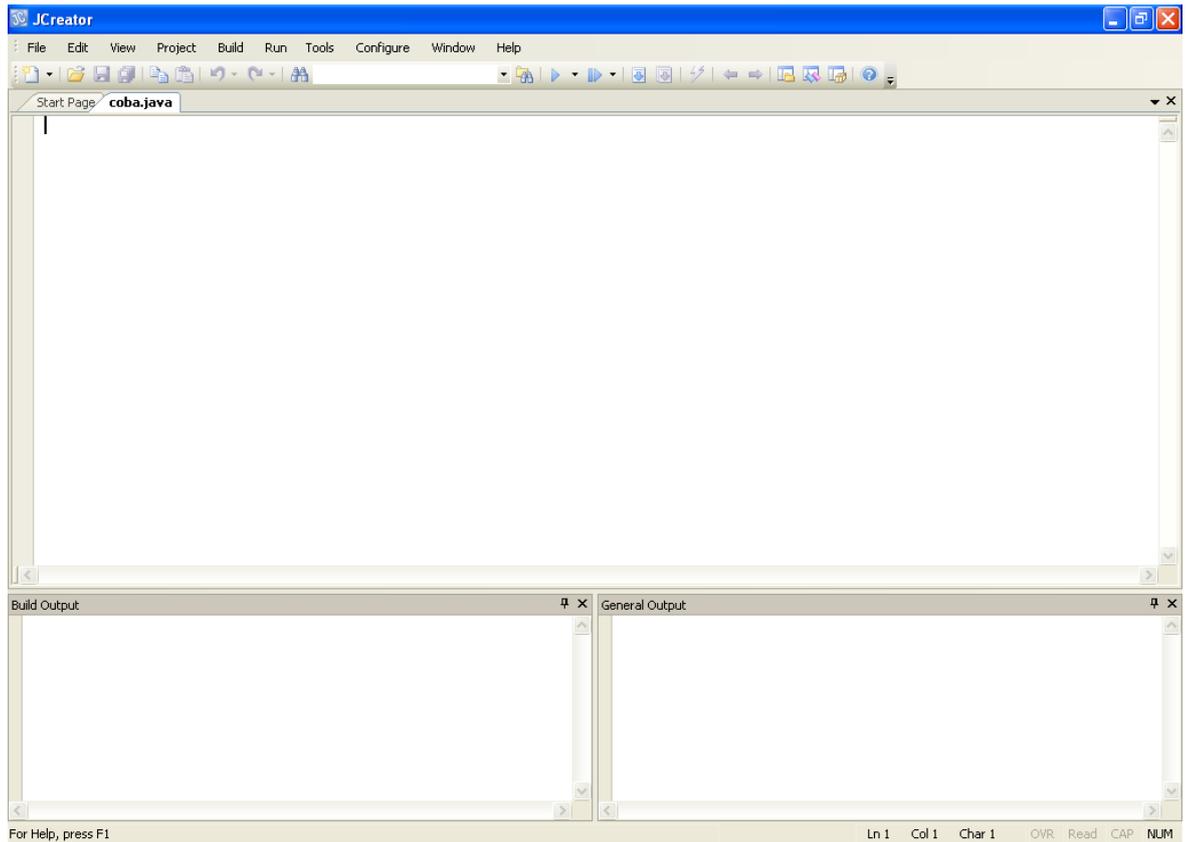
- Pastikan **Java Classes** dan **Empty Java File** dalam keadaan terpilih, dan klik tombol **Next**.
- Setelah tombol **Next** diklik, tampil kotak dialog untuk menentukan nama file dan lokasi tempat penyimpanan dengan tampilan sebagai berikut :



Isi Text Box Name dengan **coba** dan location ditentukan dengan mengklik tombol  yang berada di sebelah kanan. Contoh tampilan akhir setelah di edit kotak dialog tersebut sebagai berikut :



- Setelah yakin benar, klik tombol finish () , dan tampilan Jcreator sebagai berikut :



- Ketik baris perintah sebagai berikut :

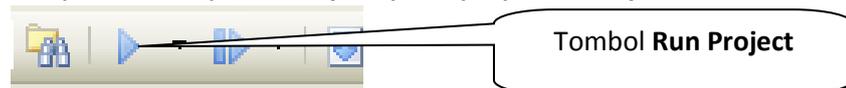
```

public class coba
{
    public static void main(String args[])
    {
        System.out.println("Hallo Dunia");
    }
}

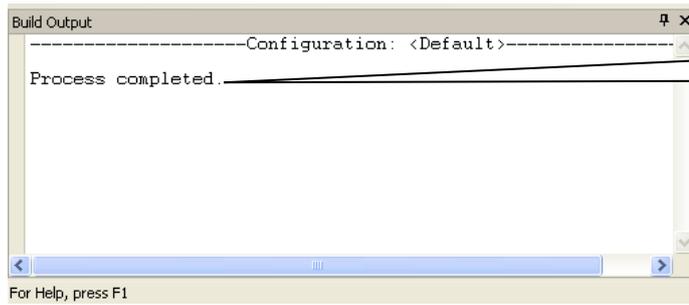
```

Sekali lagi diingatkan, bahwa bahasa Java adalah bahasa pemrograman yang bersifat case sensitive.

- Setelah selesai klik tombol **Run Project** pada toolbar untuk mengcompile/kompilasi baris perintah (compile apa yaaa ???).



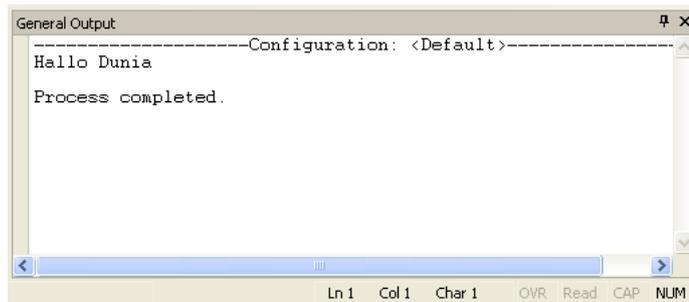
- Setelah diklik, sekarang perhatikan kotak dialog **Build Output** yang ada di bawah kiri dari aplikasi Jcreator. Jika tampilannya bertuliskan sebagai berikut :



Tulisan **Process Completed**

Maka proses compile sukses. Tidak ada kesalahan penulisan sintaks program untuk diterjemahkan. Jika bertuliskan bukan seperti diatas, perbaiki terlebih dahulu sintaks yang ditulis dan di lakukan lagi proses compile sampai berhasil.

- Setelah proses compile selesai, perhatikan kotak dialog **General Output** yang ada di bawah kanan dari aplikasi Jcreator. Kotak dialog ini berfungsi untuk menampilkan output/hasil dari sintaks program yang telah dikompilasi. Tampilannya sebagai berikut :



- Secara keseluruhan pembuatan program telah selesai.

Dalam pembuatan program-program berikutnya, langkah-langkahnya juga sama dengan langkah-langkah yang telah diuraikan dalam pembuatan program sederhana diatas. Langkah dasar dari pembuatan program menggunakan Jcreator.

Latihan :

Buat sebuah program sederhana, program diberi nama dengan **biodata.java**. Program tersebut berisi data pribadi (NIM, Nama, Tempat/Tanggal Lahir, Alamat dan Nomor Telepon) masing-masing !!!.

ALGORITMA DENGAN JAVA

TUJUAN :

- Dapat memahami dasar-dasar algoritma menggunakan java
- Dapat memahami struktur-struktur algoritma menggunakan java
- Dapat membuat aplikasi algoritma sederhana menggunakan java

1. TIPE DATA

Tipe data adalah pengelompokan data berdasarkan isi dan sifatnya. Bahasa Java bersifat **strongly typed**, sehingga setiap variabel yang digunakan harus ditentukan tipe datanya. Tipe data dalam bahasa java terdiri dari dua macam, yaitu Tipe Data Primitif dan Tipe Data Referensi.

Tipe data variabel merupakan tipe data dasar. Tipe data ini tidak diturunkan dari tipe lain. Terdapat delapan tipe variabel dalam bahasa Java, yaitu :

- Empat tipe adalah untuk bilangan bulat (Integer) : byte, short, int, long
- Dua untuk tipe angka titik mengambang (floating point) : float dan double
- Satu untuk tipe karakter yaitu char untuk karakter dengan pengkodean Unicode : char
- Satu lagi adalah Boolean untuk nilai-nilai logika : Boolean

Berikut Tabel Tipe Data Primitif yang digunakan dalam Bahasa Java.

Sebutan Tipe Data	Bentuk Penulisan Dalam Bahasa Java	Jumlah Byte Yang Diperlukan	Jangkauan Nilai Numerik
Character	char	1	0 s.d 65535
Integer	byte	1	- 128 s.d 127
	short	2	- 32768 s.d 32767
	int	4	- 2147483648 s.d 2147483647
	long	8	- 9223372036854775808 s.d 9223372036854775807
Floating point single precision	Float	4	3.4E-38 s.d 3.4E38 Positif atau negatif
Floating point double precision	double	8	1.7E-308 s.d 1.7E308 Positif atau negatif
Boolean	Boolean	1	Nilai yang dinyatakan dengan true atau false

2. VARIABEL

Variabel merupakan tempat yang digunakan untuk menyimpan suatu nilai dan/atau mengambil nilai tersebut pada sebuah program dengan tipe data tertentu. Nilai sebuah variable dapat diubah-ubah. Variabel didefinisikan/dideklarasikan menggunakan kombinasi antara identifier, tipe dan dapat juga sekaligus diinisialisasikan (pemberian nilai awal). Nama dari sebuah variable ditentukan sendiri oleh si pembuat program.

Identifier merupakan kumpulan karakter yang dapat digunakan untuk menamai variable, method, class, interface dan package. Beberapa aturan dalam menentukan sebuah identifier pada bahasa Java adalah :

- Tidak boleh sama dengan nama atau kata yang sudah disiapkan oleh komputer (reserved word) seperti keyword dan functions.
- Maksimum 32 karakter, bila lebih dari 32 karakter, maka karakter selebihnya tidak diperhatikan oleh komputer.
- Huruf besar (kapital) dan huruf kecil berbeda (case sensitive)
- Karakter pertama harus huruf atau karakter garis bawah (under score), dan karakter berikutnya boleh huruf atau angka, atau karakter garis bawah.
- Tidak boleh mengandung spasi atau blank

Bentuk Umum Pendeklarasian Variabel adalah :

```
tipe_data nama_variabel ; atau
tipe_data nama_variabel = nilai_variabel ;
```

Pada contoh program dibawah ini, tipe variabel pada teknik pemrograman berorientasi objek disebut dengan **local variable** , dikarenakan variabel tersebut dibentuk didalam sebuah method. Pada contoh ini, variabel tersebut dibentuk didalam main() method. Sehingga disebut sebagai **LOCAL VARIABLE**.

Pada modul berikutnya akan diperkenalkan tipe variabel yang lain yang disebut sebagai **INSTANCE VARIABLE** atau **ATRIBUT**.

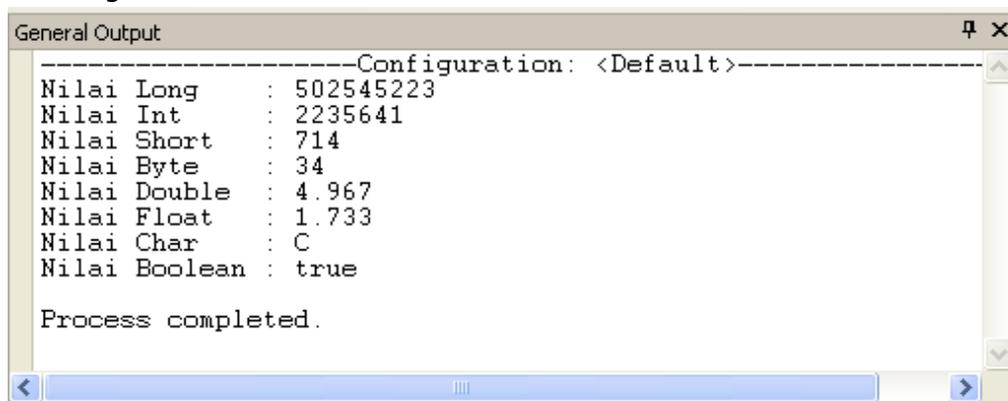
Contoh Program Mendefinisikan Variabel :

Buka Aplikasi Jcreator, buka file baru, beri nama filenya dengan nama **variable.java** dan tentukan lokasi tempat penyimpanan. Kemudian ketik sintaks program sebagai berikut :

```
public class variabel
{
    public static void main(String args[])
    {
        //Deklarasi Variabel
        long data1 = 502545223;
        int data2 = 2235641;
        short data3 = 714;
        byte data4 = 34;
        float data6 = (float) 1.733; // tipe data pecahan
        double data5 = 4.967; // tipe data pecahan
        char data7 = 'C';
        boolean data8 = true;

        //Mencetak Variabel
        System.out.println("Nilai Long      : "+ data1);
        System.out.println("Nilai Int       : "+ data2);
        System.out.println("Nilai Short    : "+ data3);
        System.out.println("Nilai Byte     : "+ data4);
        System.out.println("Nilai Double   : "+ data5);
        System.out.println("Nilai Float    : "+ data6);
        System.out.println("Nilai Char     : "+ data7);
        System.out.println("Nilai Boolean  : "+ data8);
    }
}
```

Lakukan proses Kompilasi dan Eksekusi program. Output dari program tersebut adalah sebagai berikut :



```
General Output
-----Configuration: <Default>-----
Nilai Long      : 502545223
Nilai Int       : 2235641
Nilai Short    : 714
Nilai Byte     : 34
Nilai Double   : 4.967
Nilai Float    : 1.733
Nilai Char     : C
Nilai Boolean  : true

Process completed.
```

3. OPERATOR

Perhatikan contoh di bawah ini :

$$T = A + B * C$$

Tanda tambah (+) dan asterisk (*) disebut sebagai operator, sedangkan A, B dan C disebut dengan operand (yang dioperasikan).

Operator memiliki beberapa jenis yaitu :

a. Unary

Yaitu operator yang hanya melibatkan satu operand, contoh : -10

Adalah operator yang digunakan untuk melakukan operasi matematika yang hanya melibatkan satu buah operand.

Yang termasuk ke dalam operator unary adalah sebagai berikut :

Operator	Operasi	Ekspresi
+	Membuat nilai positif	+X
-	Membuat nilai negatif	-X
++	Increment (menambah nilai dengan 1)	x++
--	Decrement (mengurangi nilai dengan 1)	x--

b. Binary

Yaitu operator yang melibatkan dua buah operand, contoh : 5 + 8

c. Ternary

Yaitu operator yang melibatkan tiga buah atau lebih operand, contoh : 10 + 5 * 2

Operator yang dapat digunakan dalam pemrograman antara lain :

a. Operator Increment

Pada bahasa java, simbol yang operator yang digunakan adalah ++. Operator Increment terbagi kedalam dua macam yaitu Pre-Increment dan Post-Increment.

1) Pre-Increment

Pre-Increment berarti menaikkan nilai yang terdapat pada sebuah variabel sebelum nilai dari variabel tersebut diproses didalam program. Operator++ akan dianggap sebagai pre-increment apabila dituliskan didepan nama variabel atau nilai yang akan dinaikkan.

Contoh Program :

Nama Program : preIncrement.java

Sintaks Program :

```
class preIncrement
{
    public static void main (String[] args)
    {
        int data;
        int x=7;
        data = ++x; //pre-increment
        System.out.println("Isi Variabel data : "+data);
    }
}
```

Output Program :

```
-----Configuration: <Default>-----
Isi Variabel data : 8
```

2) Post-Increment

Post-increment berarti menaikkan nilai yang terdapat pada sebuah variabel setelah nilai dari variabel tersebut diproses di dalam program. Pada post-increment operator ++ setelah variabel atau nilai yang akan dinaikkan. Coba program berikut ini untuk membuktikan operator post-increment.

Contoh Program :

Nama Program : postIncrement.java

Sintaks Program :

```
class postIncrement
{
    public static void main (String[] args)
    {
        int data;
        int x=7;
        data = x++; //post-increment
        System.out.println("Isi Variabel data : "+data);
    }
}
```

Output Program :

```
General Output
-----Configuration: <Default>-----
Isi Variabel data : 8
```

b. Operator Decrement

Decrement merupakan kebalikan dari increment, yang merupakan proses penurunan nilai dengan 1. Decrement juga dibagi menjadi dua macam, yaitu pre-decrement dan post-decrement.

Contoh Program :

Nama Program : cthDecrement.java

Sintaks Program :

```
class cthDecrement
{
    public static void main (String[] args)
    {
        int data1,data2;
        int x=7;
        data1 = --x; //pre-decrement
        data2 = x--; //post-decrement
        System.out.println("Isi Variabel data1 : "+data1);
        System.out.println("Isi Variabel data2 : "+data2);
    }
}
```

Output Program :

```
-----Configuration: <Default>-----
Isi Variabel data1 : 6
Isi Variabel data2 : 6
```

c. Operator Aritmatika

Yaitu operator yang biasa digunakan dalam matematika. Berikut beberapa operator aritmatika yang dapat digunakan dalam operasi-operasi dalam pemrograman, diantaranya :

- Operator Perkalian
Simbol operator yang digunakan adalah " * ", operator ini digunakan untuk mengalikan dua atau lebih operand. Contoh : $4 * 2$
- Operator Pembagian
Simbol operator yang digunakan adalah " / ", operator ini digunakan untuk pembagian dua atau lebih operand. Contoh : $4 / 2$
- Operator Penjumlahan
Simbol operator yang digunakan adalah " + ", operator ini digunakan untuk penjumlahan dua atau lebih operand. Contoh : $4 + 2$
- Operator Pengurangan
Simbol operator yang digunakan adalah " - ", operator ini digunakan untuk pengurangan dua atau lebih operand. Contoh : $4 - 2$
- Operator Sisa Pembagian
Simbol operator yang digunakan adalah " % ", operator ini digunakan untuk mendapatkan sisa pembagian. Contoh : $4 \% 2$

Contoh Program :

Nama Program : **aritmatika.java**

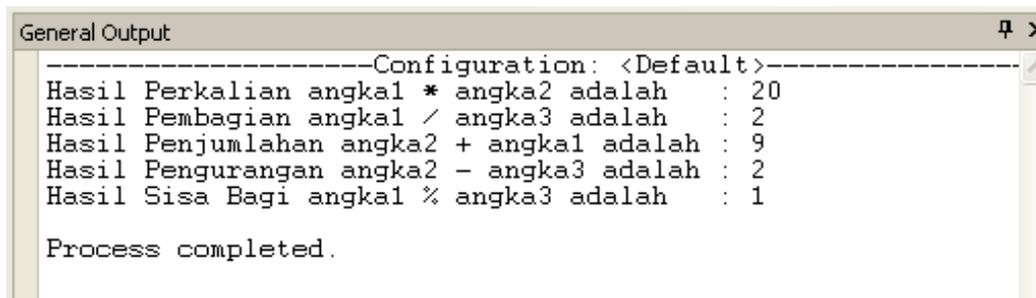
Sintaks Program :

```
public class aritmatika
{
    public static void main(String args[])
    {
        int angka1, angka2, angka3, kali, bagi, jumlah, kurang, modulus ;

        angka1 = 5;
        angka2 = 4;
        angka3 = 2;
        kali = angka1 * angka2 ;
        bagi = angka1 / angka3 ;
        jumlah = angka2 + angka1 ;
        kurang = angka2 - angka3 ;
        modulus = angka1 % angka3 ;

        System.out.println("Hasil Perkalian angka1 * angka2 adalah : "+kali);
        System.out.println("Hasil Pembagian angka1 / angka3 adalah : "+bagi);
        System.out.println("Hasil Penjumlahan angka2 + angka1 adalah : "+jumlah);
        System.out.println("Hasil Pengurangan angka2 - angka3 adalah : "+kurang);
        System.out.println("Hasil Sisa Bagi angka1 % angka3 adalah : "+modulus);
    }
}
```

Output program :



```
-----Configuration: <Default>-----
Hasil Perkalian angka1 * angka2 adalah : 20
Hasil Pembagian angka1 / angka3 adalah : 2
Hasil Penjumlahan angka2 + angka1 adalah : 9
Hasil Pengurangan angka2 - angka3 adalah : 2
Hasil Sisa Bagi angka1 % angka3 adalah : 1

Process completed.
```

d. Operator Relasi

Yaitu operator yang biasa digunakan untuk membandingkan dua buah nilai. Hasil dari operator ini adalah Benar (TRUE) atau Salah (FALSE). Biasanya operator ini digunakan pada struktur percabangan atau perulangan pada sebuah program. Operator-operator relasi yang dapat digunakan dalam pemrograman sebagai berikut :

- Operator sama dengan
Simbol operator yang digunakan adalah " == ", operator ini menyatakan bahwa nilai yang dibandingkan antara dua operan adalah sama. Contoh : a == b
- Operator tidak sama dengan
Simbol operator yang digunakan adalah " != ", operator ini menyatakan bahwa nilai yang dibandingkan antara dua operan tidak sama. Contoh : a != b
- Operator lebih dari
Simbol operator yang digunakan adalah " > ", operator ini menyatakan bahwa nilai operan pertama lebih besar dari nilai operan yang kedua. Contoh : a > b
- Operator kurang dari
Simbol operator yang digunakan adalah " < ", operator ini menyatakan bahwa nilai operan pertama lebih kecil dari nilai operan yang kedua. Contoh : a < b
- Operator lebih dari sama dengan
Simbol operator yang digunakan adalah " >= ", operator ini menyatakan bahwa nilai operan pertama lebih besar dari atau sama dengan nilai operan yang kedua. Contoh : a >= b
- Operator kurang dari sama dengan
Simbol operator yang digunakan adalah " <= ", operator ini menyatakan bahwa nilai operan pertama lebih kecil dari atau sama dengan nilai operan yang kedua. Contoh : a <= b

Contoh Program :

Nama Program : relasi.java

Sintaks Program :

```

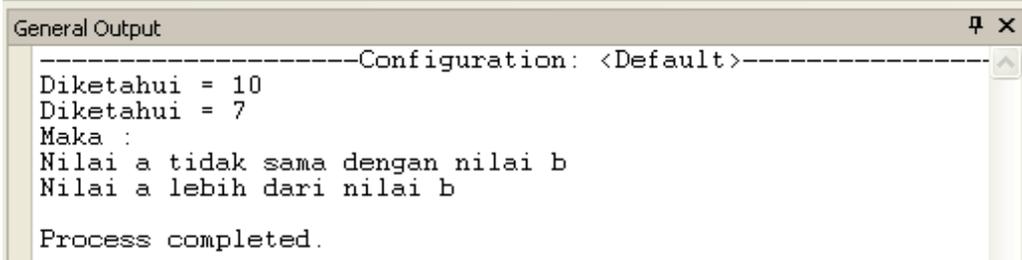
public class relasi
{
    public static void main (String args[])
    {
        int a = 10, b = 7;
        System.out.println("Diketahui = "+a);
        System.out.println("Diketahui = "+b);
        System.out.println("Maka : ");

        if (a!=b)
        {
            System.out.println("Nilai a tidak sama dengan nilai b");
        }
        else
        {
            System.out.println("Nilai a sama dengan nilai b");
        }

        if (a < b)
        {
            System.out.println("Nilai a kurang dari nilai b");
        }
        else
        {
            System.out.println("Nilai a lebih dari nilai b");
        }
    }
}

```

Output Program :



```

General Output
-----Configuration: <Default>-----
Diketahui = 10
Diketahui = 7
Maka :
Nilai a tidak sama dengan nilai b
Nilai a lebih dari nilai b
Process completed.

```

e. Operator Logika Boolean

Yaitu operator yang biasa digunakan untuk mengaitkan dua buah ungkapan kondisi menjadi sebuah kondisi. Operator-operator logika Boolean yang biasa digunakan dalam pemrograman adalah **AND (&&)**, **OR (||)** dan **NOT (!)**.

Perbedaan dari ketiga operator tersebut adalah :

- Operator AND : Hasil operator akan BENAR jika kedua ungkapan kondisi, kedua-duanya Benar.
- Operator OR : Hasil operator akan BENAR jika kedua atau salah satu ungkapan kondisi tersebut bernilai Benar.
- Operator NOT : Berfungsi pembalik nilai logika TRUE menjadi FALSE atau nilai logika FALSE menjadi TRUE

Contoh Program :

Nama Program : oprboolean.java

Sintaks Program :

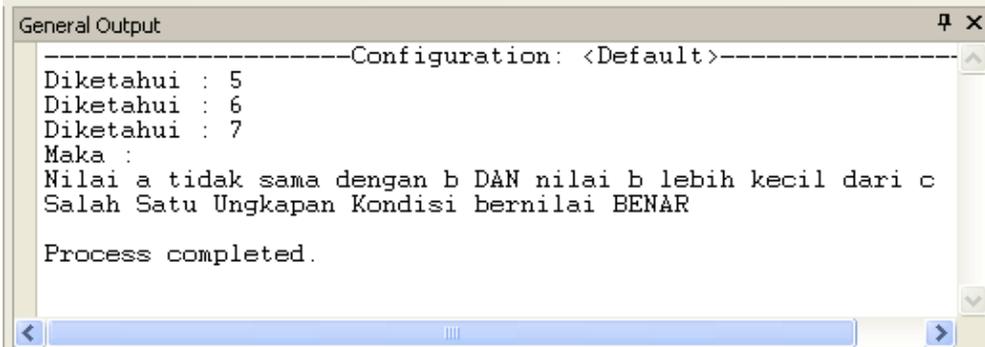
```
public class oprboolean
{
    public static void main (String args[])
    {
        int a = 5, b = 6, c = 7;

        System.out.println("Diketahui : "+a);
        System.out.println("Diketahui : "+b);
        System.out.println("Diketahui : "+c);
        System.out.println("Maka : ");

        if ((a != b) && (b < c))
        {
            System.out.println("Nilai a tidak sama dengan b DAN nilai b lebih kecil dari c");
        }
        else
        {
            System.out.println("Nilai a sama dengan b DAN nilai b lebih besar dari c");
        }

        if ((a == b) || (b < c))
        {
            System.out.println("Salah Satu Ungkapan Kondisi bernilai BENAR");
        }
        else
        {
            System.out.println("Kedua Ungkapan Kondisi bernilai SALAH");
        }
    }
}
```

Output Program :



```
General Output
-----Configuration: <Default>-----
Diketahui : 5
Diketahui : 6
Diketahui : 7
Maka :
Nilai a tidak sama dengan b DAN nilai b lebih kecil dari c
Salah Satu Ungkapan Kondisi bernilai BENAR

Process completed.
```

4. MEMAHAMI CLASS STRING

Dalam pemrograman Java string merupakan aspek penting, karena dapat mempelajari mengenai class dan objek melalui penggunaan string. String sebenarnya merupakan class yang terdapat dalam library Java. Java String merupakan salah satu class/kelas dasar yang disediakan oleh Java untuk memanipulasi karakter.

a. Membuat Objek String

Java mendefinisikan class String dalam package java.lang.String, sehingga tidak perlu melakukan impor secara eksplisit. Java String digunakan untuk mendefinisikan string yang konstant (tidak bisa berubah). Untuk membuat string, dapat melakukannya dengan beberapa cara, dan yang sering digunakan adalah contoh sebagai berikut.

Perhatikan kode berikut !
String varString = "abcd";

Kode diatas adalah bentuk singkat dari :
Char[] datanya = {'a','b','c','d'};
String varString = new String(datanya);

Jadi dapat disimpulkan bahwa String terdiri dari data array yang bertipe char, dan kita juga dapat membuat objek String dengan menggunakan keyword new yang biasa digunakan untuk membuat objek dari class.

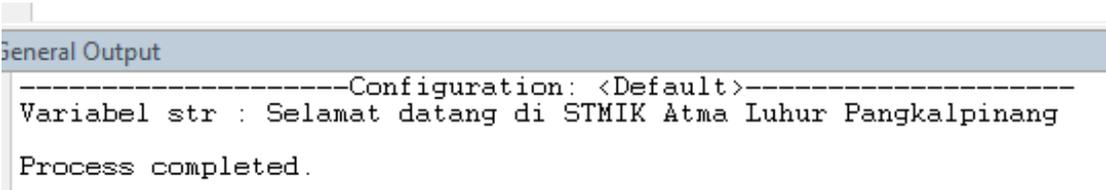
Contoh Program :

Nama Program : string1.java

Sintaks Program :

```
class string1
{
    public static void main (String[] args)
    {
        String str="Selamat datang di STMIK Atma Luhur Pangkalpinang";
        System.out.println("Variabel str : "+str);
    }
}
```

Output Program :



```
General Output
-----Configuration: <Default>-----
Variabel str : Selamat datang di STMIK Atma Luhur Pangkalpinang
Process completed.
```

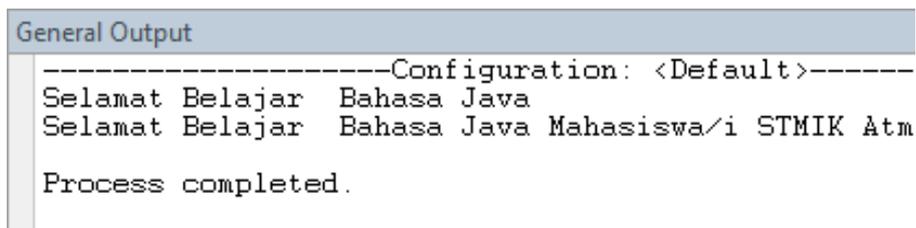
b. Menggabungkan String

Seringkali dalam pemrograman perlu menggabungkan String untuk mendapatkan String baru. Pada bahasa java, dapat menggunakan operator (+) untuk menggabungkan beberapa String.

Contoh Program :
Nama Program : gabungString.java
Sintaks Program :

```
class gabungString
{
    public static void main(String[] args)
    {
        String str;
        str="Selamat Belajar "+" Bahasa Java ";
        System.out.println(str);
        str += "Mahasiswa/i STMIK Atma Luhur";
        System.out.println(str);
    }
}
```

Output program :



```
General Output
-----Configuration: <Default>-----
Selamat Belajar Bahasa Java
Selamat Belajar Bahasa Java Mahasiswa/i STMIK Atm
Process completed.
```

c. Membandingkan Dua String

Untuk membandingkan dua objek String, dapat menggunakan fungsi sebagai berikut :

- equals(String s)
Dengan fungsi ini, maka objek string yang bersangkutan akan dibandingkan dengan objek string s, pada parameter fungsi ini, dengan membedakan antara huruf besar dengan huruf kecil.
- equalsIgnoreCase(String s)
Dengan fungsi ini, maka objek string yang bersangkutan akan dibandingkan dengan objek string s, pada parameter fungsi ini, dengan tanpa memperdulikan perbedaan antara huruf besar dengan huruf kecil.

Kedua fungsi diatas akan menghasilkan nilai boolean *true* apabila benar dan *false* apabila salah

Contoh Program :
 Nama Program : bandingString.java
 Sintaks Program :

```
class bandingString
{
    public static void main(String[] args)
    {
        String str1 = "Ilmu";
        String str2 = "Komputer";
        String str3 = "IlmuKomputer";
        System.out.println("String 1:"+str1);
        System.out.println("String 2:"+str2);
        System.out.println("String 3:"+str3);
        System.out.println("String 1 = String 3 ==> "+str1.equals(str3));
        System.out.println("String 2 = String 3 ==> "+str2.equals(str3));
        System.out.println("String 1 + String 2 = String 3 ==> "+(str1+str2).equals(str3));
        String str4 = "STMIK ATMA LUHUR";
        String str5 = "stmik atma luhur";
        System.out.println("String 4:"+str4);
        System.out.println("String 5:"+str5);
        System.out.println("Perintah Pembandingan dibawah mengabaikan antara huruf besar dengan huruf kecil");
        System.out.println("String 4 = String 5 ==> " +str4.equalsIgnoreCase(str5));
        System.out.println("String 4 + String 5 = String 5 + String 4 ==> " +(str4+str5).equalsIgnoreCase(str5+str4));
        String str6 = "SISTEMINFORMASI";
        String str7 = new String(str6);
        String str8 = "sisteminformasi";
        String str9 = "SISTEMINFORMASI";
        System.out.println("String 6:"+str6);
        System.out.println("String 7:"+str7);
        System.out.println("String 8:"+str8);
        System.out.println("String 9:"+str9);
        System.out.println("Perintah Pembandingan dengan ==");
        System.out.println("String 6 = String 7 ==> "+ (str6==str7));
        System.out.println("String 6 = String 8 ==> "+ (str6==str8));
        System.out.println("String 6 = String 9 ==> "+ (str6==str9));
    }
}
```

Output program :

```
String 1: Ilmu
String 2: Komputer
String 3: IlmuKomputer
String 1 = String 3 ==> false
String 2 = String 3 ==> false
String 1 + String 2 = String 3 ==> true
String 4: STMIK ATMA LUHUR
String 5: stmik atma luhur
Perintah Pembandingan dibawah mengabaikan antara huruf besar dengan huruf kecil
String 4 = String 5 ==> true
String 4 + String 5 = String 5 + String 4 ==> true
String 6: SISTEMINFORMASI
String 7: SISTEMINFORMASI
String 8: sisteminformasi
String 9: SISTEMINFORMASI
Perintah Pembandingan dengan ==
String 6 = String 7 ==> false
String 6 = String 8 ==> false
String 6 = String 9 ==> true
```

d. Menentukan Awal dan Akhir String

Untuk menentukan awal dan akhir String, dapat menggunakan dua fungsi utama, yaitu :

- `startsWith(String s)`
 Dengan fungsi ini, maka objek String yang bersangkutan akan diperiksa, apakah diawali oleh objek String *s*, pada parameter fungsi ini.
- `endsWith(String s)`
 Dengan fungsi ini, maka objek string yang bersangkutan akan diperiksa, apakah diakhiri oleh objek string *s*, pada parameter fungsi ini.

Fungsi diatas akan menghasilkan nilai boolean *true* bila benar dan *false* bila salah.

Contoh Program :

Nama Program : awalakhirString.java

Sintaks Program :

```
class awalAkhirString
{
    public static void main(String[] args)
    {
        String str1 = "Sekolah Tinggi Manajemen Informatika dan Komputer";
        System.out.println(str1.startsWith("Sekolah"));
        System.out.println(str1.endsWith("Sekolah"));
        System.out.println(str1.startsWith("ekolah",1));
        System.out.println(str1.startsWith("ekolah",2));
        System.out.println(str1.startsWith("ekolah",3));
        System.out.println(str1.startsWith("Tinggi",10));
        System.out.println(str1.startsWith("Tinggi",9));
        System.out.println(str1.startsWith("Tinggi",8));
        System.out.println(str1.endsWith("Komputer"));
        System.out.println(str1.startsWith("Komputer"));
    }
}
```

Output Program :

```
true
false
true
false
false
false
false
true
true
false
Process completed.
```

e. Mengurutkan String

Dapat juga melakukan pengurutan string dengan method compareTo(). Method ini membandingkan karakter-karakter pada String secara berurutan dari awal String. Misalnya string pertama bernilai "a" dan string kedua bernilai "b", maka apabila Stringpertama.compareTo(Stringkedua) akan menghasilkan nilai negatif (<0) dan apabila dilakukan sebaliknya akan menghasilkan nilai positif (>0). Nilai 0 akan dihasilkan apabila string pertama dan kedua sama.

Contoh Program :

Nama Program : urutString.java

Sintaks Program :

```

class urutString
{
    public static void main(String[] args)
    {
        String [] nama={"Mangga","Anggur","Apel","Jeruk","Nanas","Pepaya"};
        String temp;
        System.out.println("Nama - Nama Buah Sebelum Diurutkan");
        for (int i=0; i<nama.length; i++)
        {
            System.out.println(i+1+" "+nama[i]+" ");
        }
        //Mengurutkan nama buah
        System.out.println("=====");
        System.out.println("Nama - Nama Buah Setelah Diurutkan");
        for (int i=0; i<(nama.length-1); i++)
        {
            for (int j=0; j<(nama.length-1); j++)
            {
                if (nama[j].compareTo(nama[j+1])>0)
                {
                    temp=nama[j+1];
                    nama[j+1]=nama[j];
                    nama[j]=temp;
                }
            }
        }
        for (int i=0; i<nama.length; i++)
        {
            System.out.println(i+1+" "+nama[i]);
        }
    }
}

```

Output Program :

```

General Output
Nama - Nama Buah Sebelum Diurutkan
1 Mangga
2 Anggur
3 Apel
4 Jeruk
5 Nanas
6 Pepaya
=====
Nama - Nama Buah Setelah Diurutkan
1 Anggur
2 Apel
3 Jeruk
4 Mangga
5 Nanas
6 Pepaya

```

f. Menghitung Digit Karakter

Method yang digunakan untuk menghitung digit dari karakter yang diinginkan adalah length().

Contoh Program :

Nama Program : digitString.java

Sintaks Program :

```

class digitString
{
    public static void main(String[] args)
    {
        String str="STMIK Atma Luhur Pangkalpinang";
        int panjang;
        panjang = "Selamat Belajar Java".length();
        System.out.println("Variabel Str : " + str);
        System.out.println("Panjang Variabel Str : " + str.length());
        System.out.println("Panjang Variabel Panjang : " + panjang);
    }
}

```

Output Program :

```

Variabel Str : STMIK Atma Luhur Pangkalpinang
Panjang Variabel Str : 30
Panjang Variabel Panjang : 20

```

g. Membuat Array Karakter dari String

Array yang bertipe char dapat dibuat dari variabel string dengan menggunakan method `toCharArray()` dari class string. Karena method ini mengembalikan array bertipe char, maka kita perlu mendeklarasikan variabel bertipe array char untuk menyimpan hasil char array dari string. Selain method `toCharArray()`, juga ada method `getChars()`. Untuk menggunakan method `getChars()` diperlukan empat argumen, yaitu :

- Awal posisi pada string dalam integer
- Akhir posisi pada string dalam integer
- Nama variabel array char yang digunakan untuk menyimpan
- Posisi indeks pertama untuk menyimpan karakter pertama dalam integer.

Contoh Program :

Nama Program : `arrayString.java`

Sintaks Program :

```

class arrayString
{
    public static void main(String[] args)
    {
        String str="STMIK ATMA LUHUR";
        char[] arraystr=str.toCharArray();
        System.out.println("String : " +str);
        System.out.println("String Baru [toCharArray]: ");
        for (int i=0; i<arraystr.length ; i++)
        {
            System.out.println(arraystr[i]);
        }
        System.out.println("String Baru [getChars]: ");
        char[] getstr=new char[10];
        str.getChars(5,10,getstr,0);
        for (int i=0; i<getstr.length ; i++)
        {
            System.out.println(getstr[i]);
        }
    }
}

```

Output Program :

```
String : STMIK ATMA LUHUR
String Baru [toCharArray]:
S
T
M
I
K

A
T
M
A

L
U
H
U
R
String Baru [getChars]:

A
T
M
A
```

h. Mendapatkan String dari Array Karakter

Selain mengubah string menjadi array char, class String juga menyediakan method untuk mendapatkan objek String dari array bertipa char[]. Method tersebut adalah : `copyValueOf(char[] arraychar)`.

Selain itu, juga mendapatkan nilai integer dari string dengan cara menggunakan method `parseInt` dari class integer. Misalnya :

```
String kode = "345";
```

```
Int bil = Integer.parseInt(kode);
```

Contoh Program :

Nama Program : `arrayString2.java`

Sintaks Program :

```
class arrayString2
{
    public static void main(String[] args)
    {
        char[] arraystr={'A','T','M','A',' ','L','U','H','U','R'};
        System.out.println("Array Char ");
        for (int i=0; i<arraystr.length ; i++)
        {
            System.out.println(arraystr[i]);
        }
        System.out.println("\nString Baru : ");
        String str=String.copyValueOf(arraystr);
        System.out.println(str);
    }
}
```

Output Program :

```
Array Char
```

```
A  
T  
M  
A
```

```
L  
U  
H  
U  
R
```

```
String Baru :  
ATMA LUHUR
```

i. StringBuffer

StringBuffer adalah pasangan class String yang menyediakan banyak fungsi string yang umum. StringBuffer merepresentasikan urutan karakter yang dapat dikembangkan dan ditulis ulang. StringBuffer dapat disisipi karakter dan subString di tengahnya, atau ditambah di belakangnya.

Contoh Program :

Nama Program : bufferString.java

Sintaks Program :

```
class bufferString  
{  
    public static void main(String[] args)  
    {  
        StringBuffer sb = new StringBuffer();  
        String kata="ATMA";  
        sb.append("STMIK ").append(kata).append(" LUHUR");  
        System.out.println(sb.toString());  
        System.out.println(sb);  
    }  
}
```

Output Program :

```
STMIK ATMA LUHUR  
STMIK ATMA LUHUR
```

5. MEMAHAMI CLASS UNTUK INPUT DATA PADA JAVA

Dalam pemrograman dengan menggunakan bahasa Java, jika ingin menginput data melalui keyboard, maka harus dideklarasikan dahulu variable dengan menggunakan fungsi atau *class* yang ada dalam bahasa Java untuk menginput data. Proses ini disebut dengan **instansiasi**. Proses ini juga termasuk kedalam metode *object oriented programming*. Variabel yang dideklarasikan tersebut disebut sebagai **object**, unit terkecil dalam metode *object oriented programming*.

Ada tiga macam kelas (*class*) yang dapat digunakan untuk menginstansiasi object untuk menginput data didalam bahasa Java, yaitu :

- a. Buffered Reader
- b. DataInputStream
- c. JoptionPane

Class Buffered Reader dan Class DataInputStream, merupakan class yang digunakan untuk menginstansiasi object yang berbasis text atau console, sedangkan class JoptionPane digunakan untuk menginstansiasi object dengan basis GUI (Graphical User Interface) / Windows. Class Buffered Reader dan Class DataInputStream merupakan bagian dari *library functions* dalam java, sehingga *library functions* yang membawahi kedua class tersebut harus dicantumkan atau diikuti sertakan (import) kedalam program. *Library functions* tersebut adalah **java.io.***.

Contoh Program :

Nama Program : input.java

Sintaks Program :

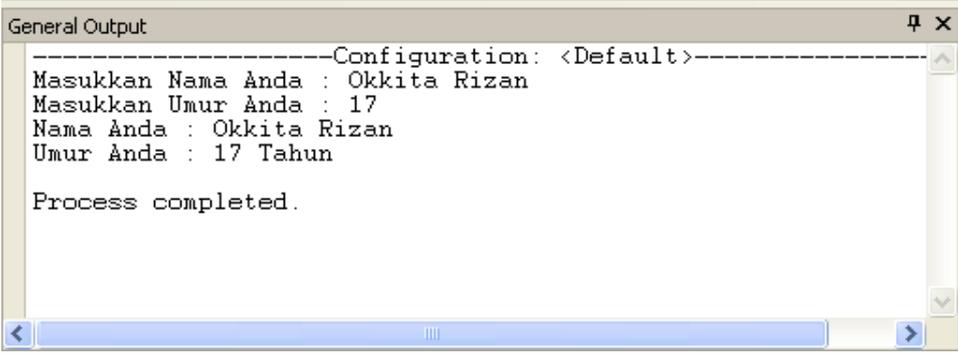
```
import java.io.*;

public class input
{
    public static void main (String args[]) throws Exception
    {
        try
        {
            String nama ;
            int umur ;
            //proses instansiasi
            BufferedReader tulis = new BufferedReader (new InputStreamReader(System.in));

            System.out.print("Masukkan Nama Anda : ");
            nama = tulis.readLine();
            System.out.print("Masukkan Umur Anda : ");
            umur = Integer.parseInt(tulis.readLine());

            System.out.println("Nama Anda : "+nama);
            System.out.println("Umur Anda : "+umur+" Tahun");
        }
        catch (Exception ie)
        {
            System.out.println("Data yang diinput salah");
        }
    }
}
```

Output Program :



```
-----Configuration: <Default>-----
Masukkan Nama Anda : Okkita Rizan
Masukkan Umur Anda : 17
Nama Anda : Okkita Rizan
Umur Anda : 17 Tahun

Process completed.
```

Penjelasan singkat mengenai program :

- **tuilis** merupakan objek yang dibuat untuk class **BufferedReader**. Proses membuat sebuah objek dari sebuah class disebut sebagai **instansiasi objek**. Dari objek inilah sebuah data dapat diinput dengan memanfaatkan method yang sudah ada pada class **BufferedReader** yaitu method **readLine()**. Dengan kata lain objek **tuilis** menggunakan method **readLine()** untuk menginput data. Setelah diinput data akan dimasukkan ke masing-masing variabel lokal yaitu variabel nama dan umur.
- Setiap data yang diinput pada java akan dianggap sebagai **String** (tipe data karakter). Untuk menampung data ke variabel yang bertipe selain **String**, maka data tersebut harus dikonversikan (diubah) tipe datanya sesuai dengan tipe data variabel penampungnya. Contoh diatas variabel umur bertipe data integer, sehingga baris perintahnya menjadi : **Integer.parseInt (input.readLine())**. **Integer.parseInt()** merupakan perintah untuk mengubah tipe data dari **String** menjadi **Integer**.
- Penggunaan **try** and **catch** berfungsi untuk memvalidasi program. Apabila ada kesalahan dalam program, maka akan dialihkan ke bagian **catch**. Sebagai contoh apabila menginput data karakter kedalam variabel umur, program akan menunjukkan ke bagian **catch**.

Berikut adalah contoh program input data dengan memanfaatkan **CLASS JOPTIONPANE**. Sama seperti contoh program diatas, perlu untuk mengimport kedalam program package / library function dari class tersebut. Untuk class **JOPTIONPANE** terdapat dalam package **javax.swing.***.

Contoh Program :

Nama Program : inputJOption.java

Sintaks Program :

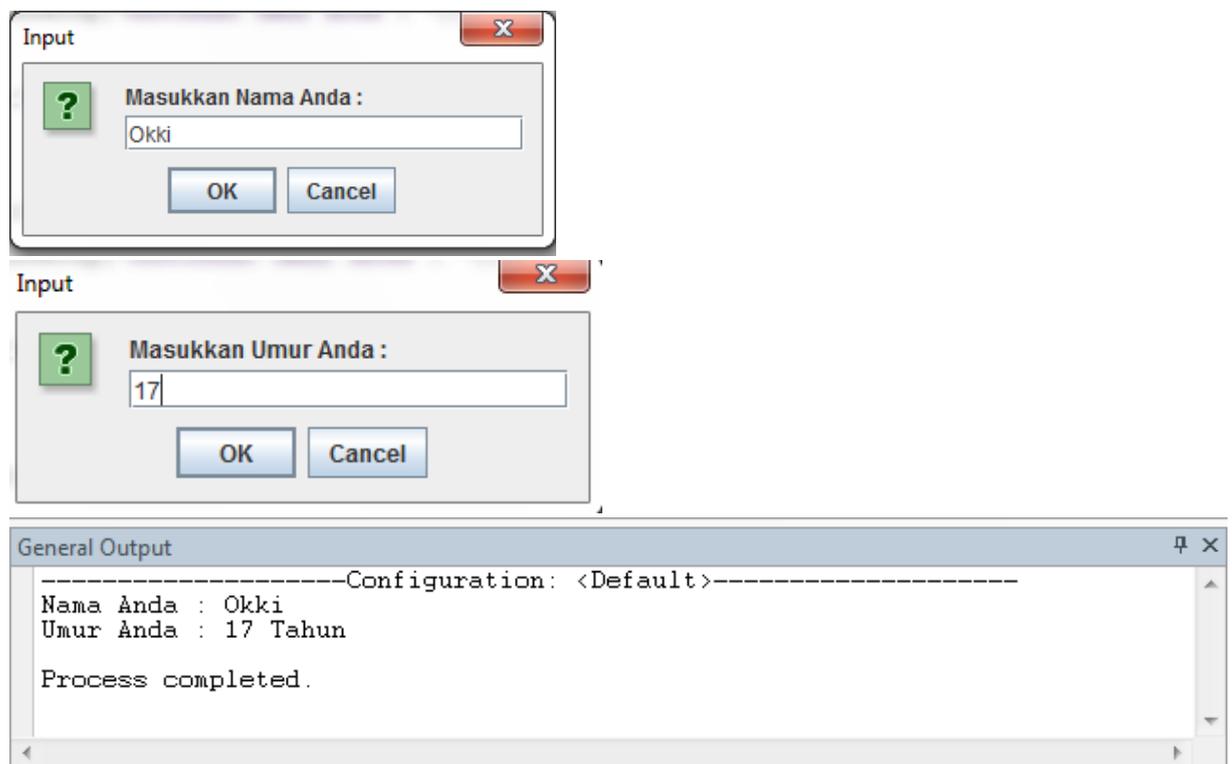
```
import javax.swing.*;

public class inputJOption
{
    public static void main (String args[]) throws Exception
    {
        try
        {
            String nama ;
            int umur ;

            nama = JOptionPane.showInputDialog("Masukkan Nama Anda : ");
            umur = Integer.parseInt(JOptionPane.showInputDialog("Masukkan Umur Anda : "));

            System.out.println("Nama Anda : "+nama);
            System.out.println("Umur Anda : "+umur+" Tahun");
        }
        catch (Exception ie)
        {
            System.out.println("Data yang diinput salah");
        }
    }
}
```

Output Program :



6. STRUKTUR PENCABANGAN

Struktur ini disebut juga dengan **SELECTION STRUCTURE** atau **DECISION STRUCTURE**. Struktur pencabangan ini menyediakan satu atau beberapa pilihan. Pilihan yang mana yang akan dikerjakan tergantung dengan kondisinya.

Sebagai contoh, setelah selesai melaksanakan ujian, maka seorang mahasiswa akan dinyatakan "**LULUS**" atau "**TIDAK LULUS**" tergantung dengan hasil ujiannya. LULUS dan TIDAK LULUS merupakan pilihan yang disediakan untuk mahasiswa sedangkan nilai ujian mahasiswa merupakan **KONDISI** yang menentukan hasil akhir yang akan diperoleh mahasiswa tersebut.

Dalam pemrograman, pilihan artinya memilih baris perintah yang mana yang akan dikerjakan. Baris-baris perintah ini akan dimasukkan kedalam "**BLOK**" atau "**KELOMPOK**" khusus. Satu "BLOK" menandakan satu pilihan. BLOK ini lah yang nantinya akan dikerjakan atau tidak tergantung dengan KONDISI.

Kondisi dalam pencabangan merupakan suatu pernyataan atau ungkapan atau expression yang mengandung nilai BENAR atau SALAH, dalam bahasa pemrograman familiar dengan istilah **TRUE** atau **FALSE**. Untuk menghasilkan TRUE atau FALSE, kondisi menggunakan **operator relasi** dan **logika boolean**.

Pencabangan terdiri dari dua macam yaitu Simple Conditional Branch dan Multiway Conditional Branch. Macam dari struktur ini tergantung dengan PILIHAN yang disediakan

- Simple Conditional Branch

Anggota dari pencabangan ini adalah IF-THEN, IF-THEN-ELSE, NESTED IF

- Contoh Program IF-THEN :

Nama Program : ifthen.java

```

import java.io.*;

public class ifthen
{
    public static void main(String args[]) throws Exception
    {
        try
        {
            int angka ;

            BufferedReader input = new BufferedReader (new InputStreamReader(System.in));

            System.out.print("Input sebuah angka : ");
            angka = Integer.parseInt(input.readLine());

            if (angka > 5)
            {
                System.out.println("Angka yang diinput > 5 ");
            }

            System.out.println("SELESAI !!!");
        }
        catch(Exception ie)
        {
            System.out.println("Data yang diinput salah !!");
        }
    }
}

```

Output Program :

```

General Output
Input sebuah angka : 7
Angka yang diinput > 5
SELESAI !!!

Process completed.

```

Atau :

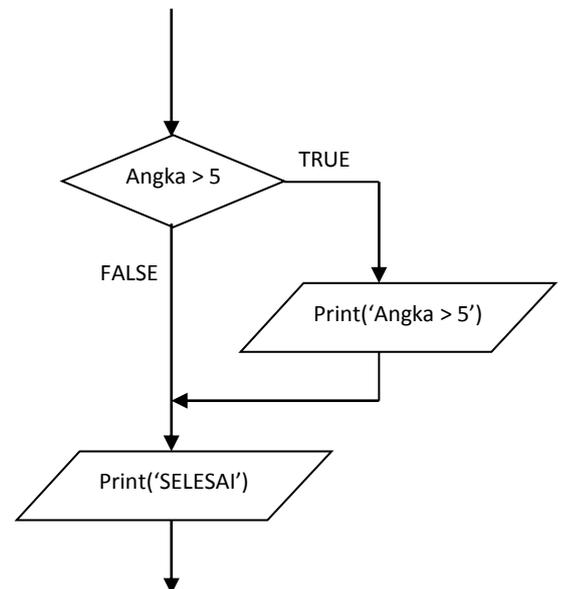
```

General Output
-----Conf i
Input sebuah angka : 3
SELESAI !!!

Process completed.

```

Flowchart :



- Contoh Program IF-THEN-ELSE :
 Nama Program : ifthenelse.java
 Sintaks Program :

```

import java.io.*;

public class ifthenelse
{
    public static void main(String args[]) throws Exception
    {
        try
        {
            int angka ;

            BufferedReader input = new BufferedReader (new InputStreamReader(System.in));

            System.out.print("Input sebuah angka : ");
            angka = Integer.parseInt(input.readLine());

            if (angka > 5)
            {
                System.out.println("Angka yang diinput > 5 ");
            }
            else
            {
                System.out.println("Angka diinput < 5" );
            }

            System.out.println("SELESAI !!!");
        }
        catch(Exception ie)
        {
            System.out.println("Data yang diinput Salah !!!");
        }
    }
}

```

Output program :

```

General Output
-----Conf
Input sebuah angka : 7
Angka yang diinput > 5
SELESAI !!!

Process completed.

```

Atau :

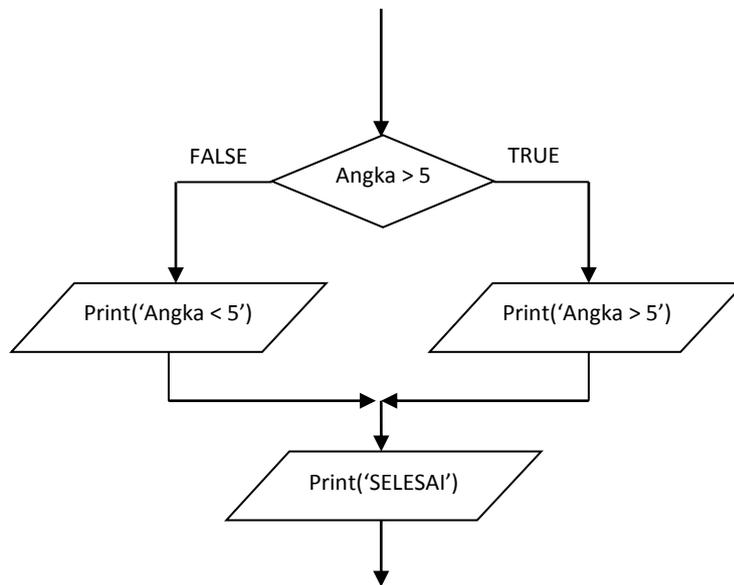
```

General Output
-----Configu
Input sebuah angka : 3
Angka diinput < 5
SELESAI !!!

Process completed.

```

Flowchart :



- Contoh Program NESTED IF :
Nama Program : nestedif.java
Sintaks Program :

```

import java.io.*;

public class nestedif
{
    public static void main(String args[]) throws Exception
    {
        try
        {
            int angka ;

            BufferedReader input = new BufferedReader (new InputStreamReader(System.in));

            System.out.print("Input sebuah angka : ");
            angka = Integer.parseInt(input.readLine());

            if ((angka > 10) && (angka <=20))
            {
                System.out.println("Angka yang diinput antara 11 s/d 20 ");
            }
            else if ((angka >= 0) && (angka <=10))
            {
                System.out.println("Angka yang diinput antara 0 s/d 10" );
            }
            else
            {
                System.out.println("Salah Input, Tidak Masuk Kriteria");
            }

            System.out.println("SELESAI !!!");
        }
        catch(Exception ie)
        {
            System.out.println("Data yang diinput SALAH !!!");
        }
    }
}

```

Output Program :

```

General Output
-----Configuration: <Default>-----
Input sebuah angka : 15
Angka yang diinput antara 11 s/d 20
SELESAI !!!
Process completed.

```

Atau :

```

General Output
-----Configuration: <Default>-----
Input sebuah angka : 8
Angka yang diinput antara 0 s/d 10
SELESAI !!!
Process completed.

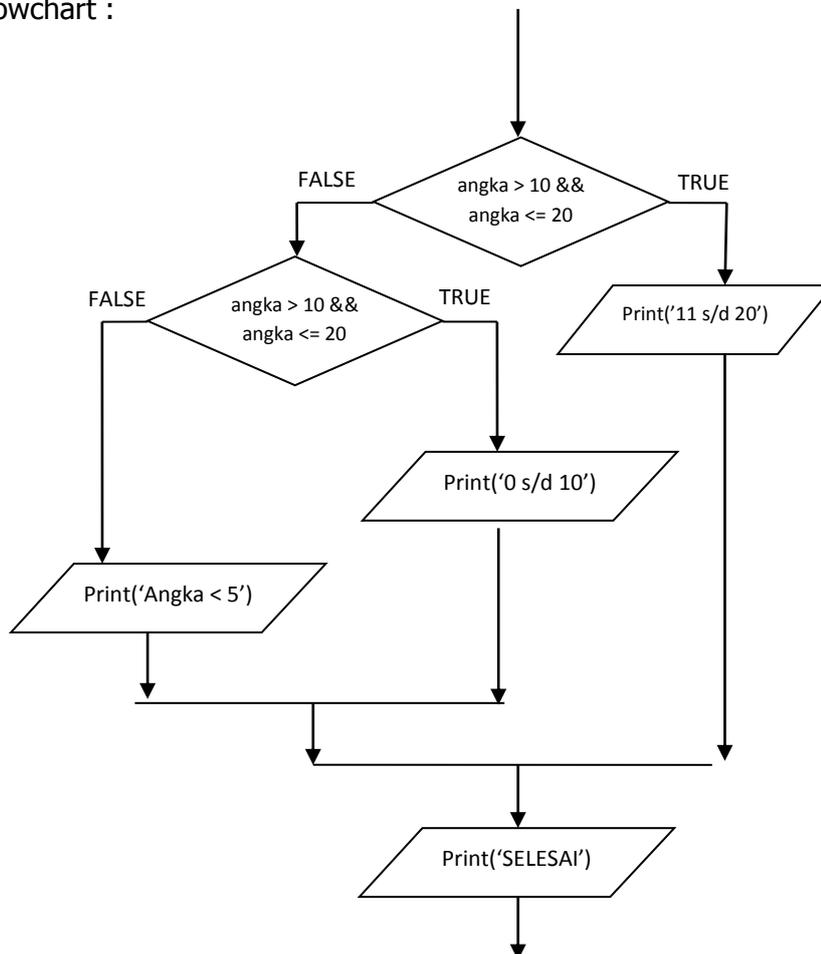
```

Atau :

```
General Output
-----Configuration: <Default>-----
Input sebuah angka : 25
Salah Input, Tidak Masuk Kriteria
SELESAI !!!

Process completed.
```

Flowchart :



- Multiway Conditional Branch
Anggota dari struktur ini adalah Switch Case.
Contoh Program :
Nama Program : switchcase.java
Sintaks Program :

```

import java.io.*;

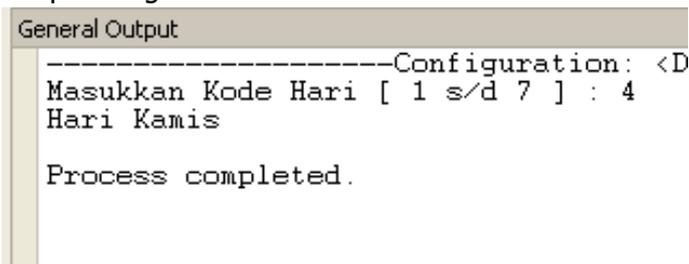
public class switchcase
{
    public static void main (String args[]) throws Exception
    {
        try
        {
            int kodehari ;
            BufferedReader input = new BufferedReader (new InputStreamReader(System.in));

            System.out.print("Input Kode Hari [ 1 s/d 7 ] : ");
            kodehari = Integer.parseInt(input.readLine());

            switch(kodehari)
            {
                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("KODE HARI SALAH !!!");
                    break ;
            }
        }
        catch(Exception ie)
        {
            System.out.println("Data yang diinput SALAH !!!!");
        }
    }
}

```

Output Program :



```

General Output
-----Configuration: <D
Masukkan Kode Hari [ 1 s/d 7 ] : 4
Hari Kamis

Process completed.

```

7. STRUKTUR PERULANGAN

Struktur ini bermanfaat dalam hal efisiensi memori yang digunakan, karena sintaks program yang sama dapat diringkas tanpa harus menuliskan berulang-ulang selama memenuhi kondisi yang telah ditentukan. Dalam bahasa pemrograman struktur alur algoritma ini biasa disebut dengan looping.

Pada dasarnya, ada 3 macam looping dasar yang digunakan dan kombinasi dari ketiganya. 3 Macam looping tersebut adalah For Next, While..Do dan Do..While. Kombinasi dari ketiganya disebut dengan Nested Loop. Berikut contoh-contohnya :

- Contoh Program For Next

Nama Program : fornex.java

Sintaks Program :

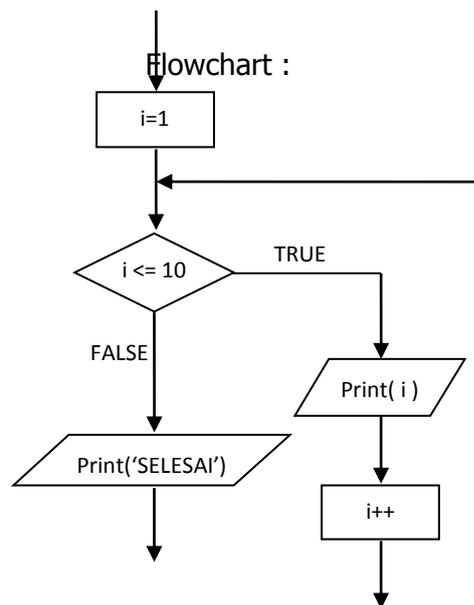
```
public class fornex
{
    public static void main(String args[])
    {
        int i ;

        for (i=1; i<=10 ;i++)
        {
            System.out.println(i);
        }
        System.out.println("SELESAI");
    }
}
```

Output Program :

```
General Output
1
2
3
4
5
6
7
8
9
10
SELESAI
Process completed.
```

Flowchart :



- Contoh Program While..Do
Nama Program : whiledo.java
Sintaks Program :

```

public class whiledo
{
    public static void main(String args[])
    {
        int i ;
        i = 1;
        while (i<=10)
        {
            System.out.println(i);
            i++;
        }
        System.out.println("SELESAI");
    }
}

```

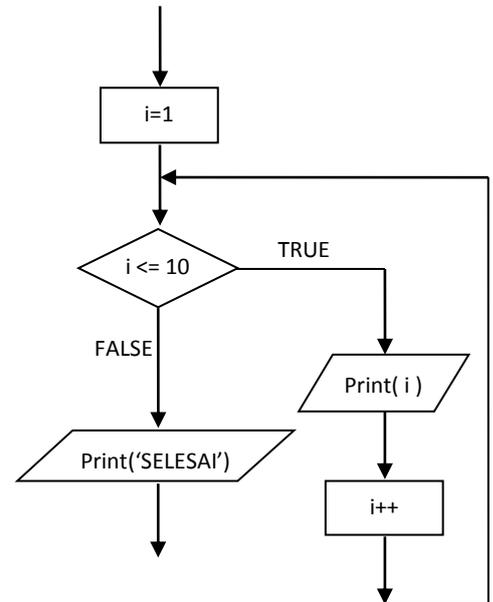
Output Program :

```

General Output
-----C
1
2
3
4
5
6
7
8
9
10
SELESAI
Process completed.

```

Flowchart :



- Contoh Program Do..While
 Nama Program : dowhile.java
 Sintaks Program :

```

public class dowhile
{
    public static void main (String args[])
    {
        int i;
        i = 1;
        do
        {
            System.out.println(i);
            i++;
        }
        while (i<=10);

        System.out.println("SELESAI");
    }
}

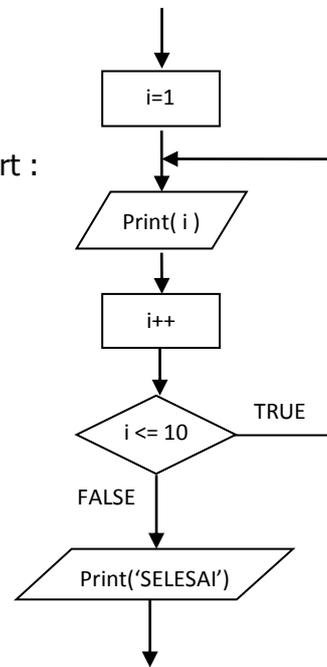
```

Output Program :

```

General Output
-----
1
2
3
4
5
6
7
8
9
10
SELESAI
Process completed.
  
```

Flowchart :



- Contoh Program Nested..Loop
 Nama Program : nestedloop.java
 Sintaks Program :

```

public class nestedloop
{
    public static void main (String args[])
    {
        int i, j;

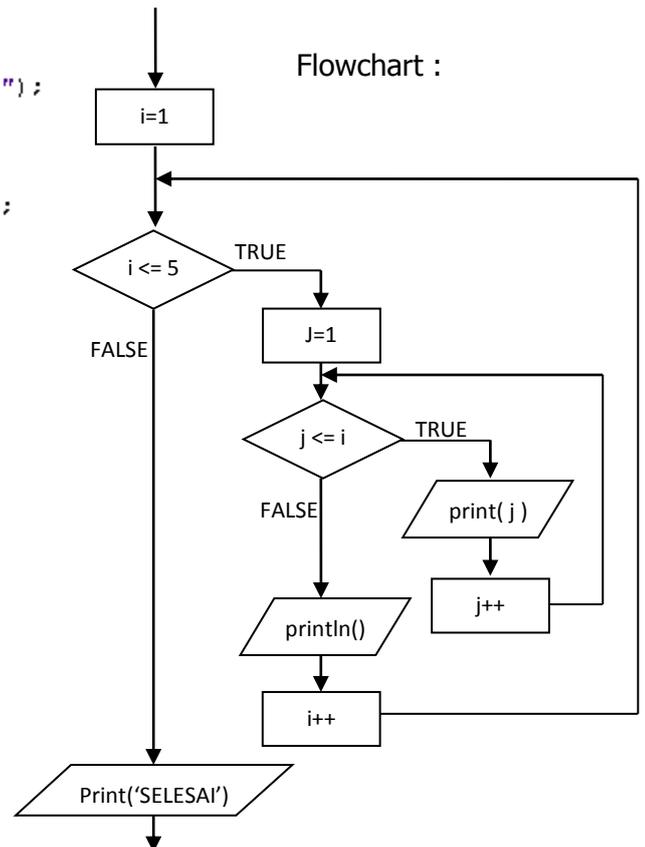
        for (i=1;i<=5;i++)
        {
            for (j=1;j<=i;j++)
            {
                System.out.print(j+" ");
            }
            System.out.println();
        }
        System.out.println("SELESAI");
    }
}
  
```

Output Program :

```

General Output
-----C
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
SELESAI
Process completed.
  
```

Flowchart :



8. ARRAY

Array adalah variabel sejenis yang berderet-deret sedemikian rupa sehingga alamatnya saling bersambung atau bersebelahan/berdampingan (contiguous). Array merupakan variabel yang dapat menampung banyak nilai / data dengan tipe data yang sejenis. Berbeda dengan variabel yang telah dibahas pada bab terdahulu, yang hanya bisa digunakan untuk menyimpan satu nilai saja, yang disebut dengan variabel tunggal. Nilai pada array diakses berdasarkan indeks yang tersusun secara berurutan. Array dapat berupa satu dimensi, dua dimensi, tiga dimensi ataupun banyak dimensi

a. Array 1 Dimensi

Array satu dimensi adalah array yang memiliki satu kolom dengan banyak baris, atau satu baris dengan banyak kolom, tergantung bagaimana cara mengilustrasikannya dalam pikiran. Array satu dimensi disebut juga dengan vector karena hanya mempunyai satu arah saja.

Bentuk Umum Deklarasi Array Satu Dimensi :

```
tipe_data [] nama_array = new tipe_data [jumlah_indeks] ;
```

Contoh programnya sebagai berikut :

Nama Program : array1d.java

Sintaks Program :

```
public class array1d
{
    public static void main(String args[])
    {
        int [] array = new int [5];
        int i;

        array[0] = 1;
        array[1] = 2;
        array[2] = 3;
        array[3] = 4;
        array[4] = 5;

        System.out.println("Isi Array 1 Dimensi : ");
        for (i=0;i<=4;i++)
        {
            System.out.println("Array Indeks ke-"+i+" : "+array[i]);
        }
    }
}
```

b. Array 2 Dimensi

Array dua dimensi adalah array yang memiliki dua atau lebih kolom dengan banyak baris, atau dua atau lebih baris dengan banyak kolom, bergantung bagaimana mengilustrasikannya didalam pikiran. Array dua dimensi juga dapat dipandang sebagai gabungan dari dua atau beberapa array satu dimensi. Array dua dimensi disebut juga dengan matriks.

Bentuk Umum Deklarasi Array Dua Dimensi :

```
tipe_data [] [] nama_array = new tipe_data [jml_indeks_baris] [jml_indeks_kolom] ;
```

Contoh programnya sebagai berikut :

```
public class array2d
{
    public static void main(String args[])
    {
        int [][] array = new int[2][2];
        int i,j;

        array[0][0]= 1;
        array[0][1]= 2;
        array[1][0]= 3;
        array[1][1]= 4;

        System.out.println("Isi Array 2 Dimensi ");
        for (i=0;i<=1;i++)
        {
            for (j=0;j<=1;j++)
            {
                System.out.print(" "+array[i][j]);
            }
            System.out.println();
        }
    }
}
```

KONSEP CLASS & DESAIN CLASS

TUJUAN :

- Dapat memahami salah satu fundamental OOP yaitu class dan object
- Dapat memahami bagian-bagian pada class
- Dapat memahami dan membuat class dengan java

1. KONSEP CLASS DAN OBJECT

Class adalah ciri dari Object Oriented Programming. Class merupakan kumpulan dari objek yang sejenis. Sedangkan objek merupakan benda, baik secara fisik atau konseptual. Ciri dari objek adalah memiliki atribut/property/data (data member) dan method/behavior/function (member function).

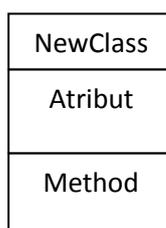
Atribut adalah variabel-variabel yang menyatakan karakteristik/ciri suatu objek (what they have). Methode adalah fungsi-fungsi yang bertugas memanipulasi nilai pada data member (what they do). Fungsi yang paling sering ada pada sebuah objek adalah fungsi untuk mengubah dan menginformasikan nilai dari data member objek. Methode juga digunakan untuk mengkomunikasikan data dalam class dengan lingkungan luar class. Pengaksesan data objek secara langsung tidak diperbolehkan.

Perbedaan antara class dengan objek adalah Class merupakan desain dan objek merupakan perwujudan suatu Class. Class bersifat abstrak dan objek bersifat kongkrit.

2. CLASS DIAGRAM

Salah satu tools yang biasa digunakan untuk memodelkan/ menggambarkan/ merancang aplikasi berparadigma Object Oriented adalah UML atau Unified Modelling Language. Pada UML terdapat beberapa diagram yang mempunyai peran tersendiri. Salah satu diagram yang terdapat didalam UML dikenal dengan **CLASS DIAGRAM**. Diagram ini secara khusus menggambarkan sebuah class dan hubungan dengan class yang lain dalam sebuah sistem. Dilingkungan STMIK Atma Luhur, CLASS DIAGRAM dan diagram-diagram yang lainnya diperkenalkan pada mata kuliah PEMODELAN SISTEM INFORMASI, sehingga dalam modul praktikum ini, tidak secara detail menggambarkan dan bagaimana merancang sebuah class diagram, hanya sebagai diagram konseptual untuk membantu membuat class yang sesungguhnya didalam program.

Bentuk umum dari class diagram digambarkan sebagai berikut :



Class diagram mempunyai tiga bagian. Bagian atas atau pertama digunakan sebagai nama class/kelas. Aturan membuat nama class sesuai disesuaikan dengan aturan bahasa pemrograman yang digunakan. Biasanya nama dari sebuah class sesuai dengan kumpulan object yang ditampung oleh class tersebut.

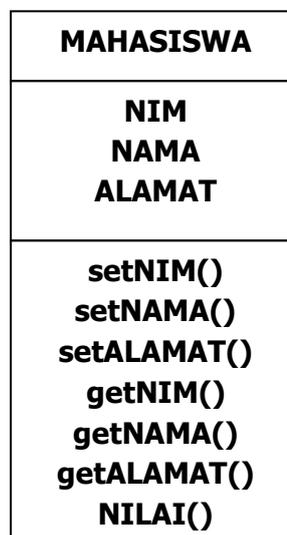
Bagian tengah atau kedua digunakan untuk menampung atribut dari class. Pada basis data atribut disebut juga sebagai **FIELD**. Pada bahasa java atribut disebut juga dengan **INSTANCE VARIABLE**.

Pada bagian yang bawah atau ketiga digunakan untuk menampung method dari class. Biasanya method berfungsi untuk memanipulasi data dari data member/atribut, tetapi tidak menutup kemungkinan sebuah method tidak berfungsi untuk memanipulasi data dari data member, tergantung dengan sifat method itu sendiri. Untuk method yang berfungsi untuk menampung atribut, diberi nama dengan awalan **set** dan **get** disertai dengan **INSTANCE VARIABLE** nya.

Contoh :

Desain sebuah class yang berfungsi sebagai kumpulan data dari mahasiswa. Class tersebut memiliki atribut diantaranya NIM, NAMA dan ALAMAT. Class tersebut juga terdapat method untuk menentukan dan mengambil data NIM, NAMA dan ALAMAT. Selain itu, juga terdapat method untuk melihat nilai yang telah diperoleh mahasiswa.

Dari uraian tersebut, class diagram untuk mahasiswa dapat didesain sebagai berikut :



3. CLASS DIAGRAM MENJADI CLASS PADA JAVA

Dari contoh pada point 2 diatas, apabila desain class mahasiswa tersebut diimplementasikan kedalam bahasa java, maka class mahasiswa tersebut menjadi sebagai berikut :

a. Class

Pada bagian atas atau pertama akan menjadi nama class. Aturan pemberian nama class pada java sama seperti memberi nama variable pada java. Class mahasiswa dapat digambarkan sebagai berikut :

```
class mahasiswa
{
    //statements
}
```

Tanda kurung kurawal penanda awal dan akhir dari sebuah class. Atribut dan method dari sebuah class akan ditulis diantara tanda kurung kurawal tersebut.

b. Atribut

Bagian kedua dari class diagram adalah atribut. Nama lain dari atribut adalah **INSTANCE VARIABLE**. Artinya, atribut pada konsep class diagram akan menjadi **INSTANCE VARIABLE** dalam pemrograman. Pada sebagian besar bahasa pemrograman, saat pendeklarasian variabel harus menyertakan apa tipe data dari variabel tersebut, artinya variabel tersebut digunakan untuk menampung data jenis apa.

Dari contoh program modul sebelumnya, LOCAL VARIABLE harus menyertakan tipe data yang digunakan. Sama seperti LOCAL VARIABLE, pembuatan INSTANCE VARIABLE juga harus menyertakan tipe datanya.

Pada contoh class mahasiswa diatas, mempunyai tiga atribut yaitu NIM, NAMA dan ALAMAT. Tipe data yang sesuai untuk tiga atribut tersebut adalah Tipe Data String karena menampung jenis data karakter.

Penulisan atribut dalam class mahasiswa digambarkan sebagai berikut :

```
class mahasiswa
{
    String NIM;
    String NAMA;
    String ALAMAT;
}
```

c. Method

Bagian ketiga dari class diagram adalah method. Untuk penulisan method dalam class ada yang diawali dengan **void** dan ada pula yang diawali dengan tipe data dari method tersebut. Method yang diawali dengan **void**, menandakan bahwa method ini tidak mengembalikan nilai (**return value**), sedangkan method yang diawali dengan tipe data, menandakan ada nilai yang akan dikembalikan (**return value**). Method yang mempunyai return value, akan terdapat perintah return() didalam method tersebut. Biasanya method yang diawali dengan void adalah method yang bernama **set..** dan method yang diawali dengan tipe data adalah method yang bernama **get..** Kesimpulannya, apakah dengan awalan void atau tipe data, tergantung dengan fungsi method itu sendiri.

Penulisan method dalam class mahasiswa sebagai berikut :

```
class mahasiswa
{
    void setNIM()
    { //statements
    }
    void setNAMA()
    { //statements
    }
    void setALAMAT()
    { //statements
    }
    String getNIM()
    { //statements
      return(..);
    }
    String getNAMA()
    { //statements
      return(..);
    }
    String getALAMAT()
    { //statements
      return(..);
    }
    int NILAI()
    { //statements
      return(..);
    }
}
```

- d. Class lengkap dengan atribut dan method.
Bentuk lengkap class mahasiswa yang mempunyai atribut dan method digambarkan sebagai berikut :

```
class mahasiswa
{
  String NIM;
  String NAMA;
  String ALAMAT;
  void setNIM()
  { //statements
  }
  void setNAMA()
  { //statements
  }
  void setALAMAT()
  { //statements
  }
  String getNIM()
  { //statements
    return(..);
  }
  String getNAMA()
  { //statements
    return(..);
  }
  String getALAMAT()
  { //statements
    return(..);
  }
  int NILAI()
  { //statements
    return(..);
  }
}
```

Catatan :

Bentuk class diatas baru menggambarkan apabila sebuah class diagram yang diimplementasikan menjadi class pada java, tetapi bentuk class tersebut belum dapat di compile dan di eksekusi. Class tersebut merupakan implementasi dari class diagram.

IMPLEMENTASI OBJEK, ATRIBUT DAN METHOD PADA JAVA

TUJUAN :

- Dapat memahami penggunaan object pada java
- Dapat memahami penggunaan atribut pada java
- Dapat memahami penggunaan method pada java

1. INSTANSIASI OBJEK

Instansiasi objek adalah proses membuat/menciptakan sebuah objek dari sebuah class. Pembuatan objek adalah dengan cara membuat sebuah variabel yang akan menunjuk ke objek tersebut. Variabel seperti ini disebut dengan **variabel objek**. Bentuk umum dari instansiasi object pada java adalah sebagai berikut :

```
namaclass namavariabel = new namaclass();
```

Sebuah objek diciptakan di luar class objek itu sendiri. Pada java, biasanya objek dari sebuah class diciptakan pada class yang mempunyai main() method (disebut saja sebagai class main()). Melalui class main() ini objek akan menggunakan seluruh atribut dan method dari classnya.

Contoh : berikut adalah contoh pembuatan sebuah objek pada class main() .

```
class testMahasiswa
{
    public static void main (String args[])
    {
        mahasiswa mhs1 = new mahasiswa();
    }
}
```

Proses instansiasi tersebut dapat diartikan membuat sebuah objek bernama **mhs1** dari **class mahasiswa**, objek dibuat dari **class main()** bernama **testMahasiswa**.

2. PENGGUNAAN METHOD MELALUI SEBUAH OBJEK

Method yang digunakan sebuah objek dipanggil dengan cara menyebutkan objeknya diikuti dengan method yang akan digunakan.

Pada contoh berikut akan diperkenalkan method tanpa parameter dan method dengan parameter. Pada contoh berikut, satu file java terdapat 2 buah class, yaitu class yang akan dibuat objeknya dan class main(). Apabila terdapat kondisi seperti ini, nama file java dinamai sama dengan nama class main()-nya.

a. Method tanpa parameter

Pada contoh berikut, pada class mahasiswa akan dibuat sebuah method bernama **hello()**. Method ini tidak mempunyai **return value**, sehingga awalnya menggunakan **void**. Method ini akan dipanggil oleh sebuah objek dari class main(). Objek yang dibuat bernama **mhs1**.

Nama file : **testMethodNon.Java**

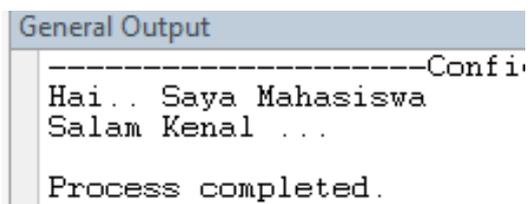
Sintaks program :

```
class mahasiswa
{
    void hello()
    {
        System.out.println("Hai.. Saya Mahasiswa");
        System.out.println("Salam Kenal ...");
    }
}

class testMethodNon
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        mhs1.hello();
    }
}
```

Output program :



```
General Output
-----Confir
Hai.. Saya Mahasiswa
Salam Kenal ...

Process completed.
```

Penjelasan :

- mahasiswa mhs1 = new mahasiswa(), adalah proses instansiasi objek dengan nama mhs1.
- mhs1.hello(), adalah perintah yang menunjukkan objek mhs1 memanggil/ menggunakan method hello().

b. Method menggunakan parameter

Parameter pada method berfungsi untuk menampung data yang akan dikirimkan ke dalam method ataupun mengirimkan data dari method ke bagian pemanggilnya. Parameter sebenarnya adalah variabel, sehingga perlu didefinisikan juga tipe datanya. Tidak ada batasan sebuah method dapat memiliki beberapa parameter. Berikut adalah contoh sebuah class dengan method yang menggunakan satu buah parameter.

Nama file : **testMethodParameter.java**

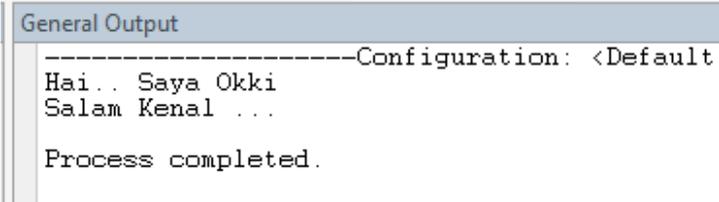
Sintaks program :

```
class mahasiswa
{
    void hello(String Nm)
    {
        System.out.println("Hai.. Saya "+Nm);
        System.out.println("Salam Kenal ...");
    }
}

class testMethodParameter
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        mhs1.hello("Okki");
    }
}
```

Output program :



```
General Output
-----Configuration: <Default:
Hai.. Saya Okki
Salam Kenal ...

Process completed.
```

Penjelasan :

- void hello(String Nm), adalah sebuah method yang memiliki sebuah parameter bernama Nm dengan tipe data String.
- mhs1.hello("Okki") adalah objek mhs1 menggunakan method hello dengan mengirimkan kata "Okki" ke parameter **Nm**.

3. STATIC METHOD

STATIC METHOD adalah method yang dimiliki oleh class, bukan milik objeknya. Berbeda dengan contoh penggunaan method di atas, penggunaan static method dalam pemrograman, dapat dipanggil langsung tanpa menggunakan objek. Tanpa harus membuat objek untuk memanggil method tersebut. Selain method, atribut juga dapat diberlakukan menjadi milik class, menjadi static variabel. Berikut adalah contoh penerapan static method dalam pemrograman.

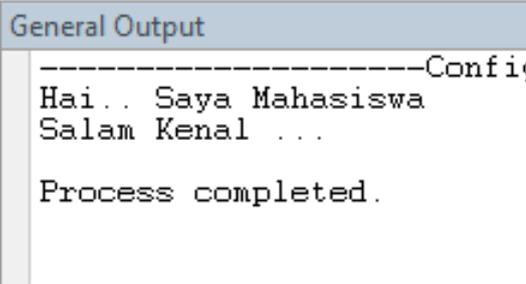
Nama file : testStaticMethod.java

Sintaks program :

```
class mahasiswa
{
    static void tampil()
    {
        System.out.println("Hai.. Saya Mahasiswa");
        System.out.println("Salam Kenal ...");
    }
}

class testStaticMethod
{
    public static void main (String[] args)
    {
        mahasiswa.tampil();
    }
}
```

Output Program ;



```
General Output
-----Config
Hai.. Saya Mahasiswa
Salam Kenal ...

Process completed.
```

Penjelasan Program :

- Pada class mahasiswa terdapat sebuah method berjenis static dengan nama tampil(). Ciri method tersebut bertipe static adalah terdapat keyword static pada saat pendeklarasian methodnya.
- Pada class testStaticMethod, tidak dibuat sebuah objek dari class mahasiswa. method tampil() langsung dipanggil dengan menyebutkan nama classnya.

4. PENGGUNAAN ATRIBUT MELALUI SEBUAH OBJEK

Atribut adalah ciri sebuah objek. Pada pemrograman atribut berfungsi untuk menampung data dari objek. Pada pemrograman, atribut adalah **INSTANCE VARIABLE** (silahkan baca modul sebelumnya)

Sebuah atribut dapat digunakan sebuah objek dengan cara diakses melalui method. Ada beberapa method yang berfungsi untuk memanipulasi data dari atributnya. Ciri method tersebut biasanya diawali dengan **set** atau **get** kemudian diikuti dengan nama atributnya. Method **set** dan **get** biasanya berpasangan untuk memanipulasi sebuah atribut. Perbedaan Method **set** dan **get** adalah :

- Method yang menggunakan **set** digunakan untuk menerima data yang dikirimkan oleh objek ke atribut.
- Method yang menggunakan **get** digunakan untuk mengirimkan data dari atribut ke objek.
- Method **set** menggunakan parameter dan tidak mempunyai return value, sehingga awalan method menggunakan **void**.
- Method **get** tidak menggunakan parameter dan mempunyai return value, sehingga awalan method menggunakan tipe data methodnya.

Berikut contoh program dengan sebuah class yang mempunyai sebuah atribut dan method untuk digunakan objeknya.

a. Atribut dengan method **set**

Nama file : testMethodSet.java

Sintak program :

```
class mahasiswa
{
    String Nama;
    void setName(String Nm)
    {
        Nama = Nm;
        System.out.println("Hai.. Saya "+Nama);
        System.out.println("Salam Kenal ...");
    }
}

class testMethodSet
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        mhs1.setName("Okki");
    }
}
```

Penjelasan program :

- Parameter yang dikirim oleh objek mhs1 berupa kata "Okki". Kata ini ditampung ke parameter **Nm** pada method **setNama**, dari parameter Nm kata tersebut dimasukkan ke atribut **Nama**. Melalui atribut **Nama** kata yang telah dikirim akan ditampilkan.
- Method **setNama** menggunakan awalan void, seluruh statements langsung dikerjakan pada method tersebut tanpa dikembalikan ke class main().

b. Mengenal keyword **this**

Salah satu kelebihan dari java adalah dapat membuat variabel parameter sebuah method dengan nama yang sama dengan atributnya (instance variable). Cara ini bisa dilakukan dikarenakan java menyediakan sebuah keyword yang dapat membedakan antara variabel parameter dengan atribut. Keyword tersebut dinamai dengan "**this**". Variabel yang menggunakan keyword **this**, menunjukkan bahwa variabel tersebut adalah atribut/instance variabel dari class nya.

Berikut contoh program menggunakan keyword **this**.

Nama file : testMethodThis.java

Sintak Program :

```
class mahasiswa
{
    String Nama;
    void setNama(String Nama)
    {
        this.Nama = Nama;
        System.out.println("Hai.. Saya "+Nama);
        System.out.println("Salam Kenal ...");
    }
}

class testMethodThis
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        mhs1.setNama("Okki");
    }
}
```

Penjelasan program :

- Nama variabel parameter dibuat sama dengan nama atribut classnya yaitu Nama. Keyword **this** menunjukkan perbedaan keduanya.
- **this.nama** menunjukkan variabel tersebut adalah atribut / instance variabel dari class mahasiswa.

c. Atribut dengan Method **get**

Berikut adalah contoh method get yang akan mengembalikan nilai (return value) pada objek.

Nama File : testMethodGet.java

Sintaks program :

```
class mahasiswa
{
    String Nama = "Okki";
    String getNama()
    {
        return (Nama);
    }
}

class testMethodGet
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        System.out.println("Hai.. Saya "+mhs1.getNama());
        System.out.println("Salam Kenal ...");
    }
}
```

Penjelasan program :

- Pada contoh program ini, kata "**Okki**" tidak dikirim melalui parameter, tetapi langsung didefinisikan pada atributnya.
- Tipe data yang digunakan pada method getHello adalah String, dikarenakan nilai/data yang dikembalikan (return) bertipe karakter, mengikuti tipe data dari atributnya.
- Return menunjukkan data yang dikembalikan pada objek yang meminta, pada contoh ini, data diambil melalui method **getHello()**.

d. Atribut dengan method **set** dan **get**.

Berikut adalah contoh program dengan class yang mempunyai dua buah method yaitu method **set** dan method **get**. Program pada umumnya menggunakan method **set** dan **get** secara berpasangan apabila class tersebut membutuhkan method untuk memanipulasi atribut.

Nama File : testMethodSetGet.java

Sintaks Program :

```
class mahasiswa
{
    String Nama;

    void setName(String Nama)
    {
        this.Nama = Nama;
    }
    String getName()
    {
        return (Nama);
    }
}

class testMethodSetGet
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        mhs1.setName("Okki");

        System.out.println("Hai.. Saya "+mhs1.getName());
        System.out.println("Salam Kenal ...");
    }
}
```

Penjelasan Program :

- Kata "Okki" dimasukkan ke atribut Nama pada class mahasiswa melalui method **setName()**.
- Setelah dimasukkan ke atribut Nama, kata "Okki" dikirimkan kembali ke objek menggunakan method **getName()**.

ASOSIASI PADA OOP

TUJUAN :

- Dapat memahami bentuk asosiasi pada object oriented programming
- Dapat memahami bentuk asosiasi pada java
- Dapat memahami bentuk agregasi pada java

1. MEMAHAMI ASOSIASI

Membuat program menggunakan teknik object oriented pada intinya adalah membuat class yang akan digunakan oleh objek. Adakalanya pada saat membuat program tidak cukup menggunakan satu class saja atau dengan maksud lain adalah memisahkan satu class menjadi beberapa class yang lain. Pemisahan class diperlukan karena antar objek dari sebuah class tidaklah sama, baik dari segi atribut ataupun dari methodenya.

Beberapa class pada satu program tidak berdiri sendiri. Melainkan saling berhubungan antara satu class dengan class yang lain. Tentunya, penghubung antar class adalah menggunakan objek dari masing-masing class. Hubungan antar class ini lah yang disebut dengan **relasi/relationship**. Relasi ini akan menunjukkan tingkatan relasi antara satu class dengan class yang lain. Relasi pada pemrograman berorientasi objek relasi antar class di sebut dengan **ASOSIASI**.

Antara pemrograman terstruktur dan pemrograman berorientasi objek mempunyai istilah yang berbeda untuk menyatakan tingkatan relasi antar class/entitas .

Jika pada pemrograman terstruktur tingkatan relasi disebut dengan **CARDINALITY**. Pada pemrograman berorientasi objek tingkatan relasi disebut dengan **MULTIPLICITY**.

CARDINALITY pada pemrograman terstruktur digunakan untuk menyatakan tingkatan relasi antar entitas pada **ENTITY RELATIONSHIP DIAGRAM (ERD) / DIAGRAM ER**. Umumnya ERD mempunyai tiga tingkatan relasi yaitu :

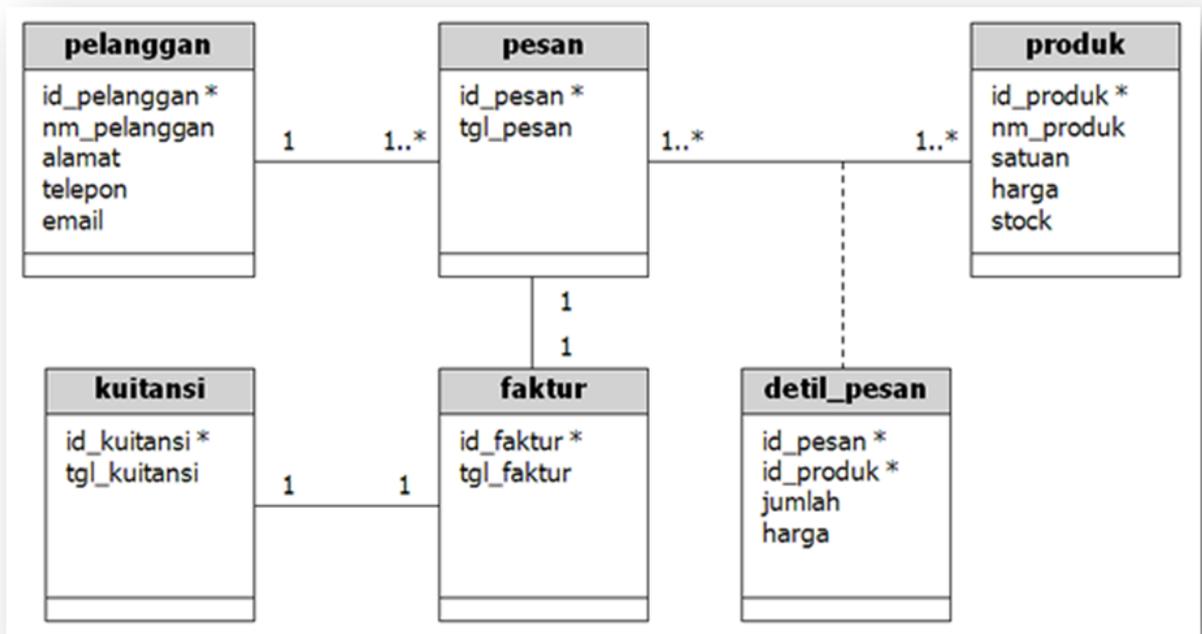
- 1 .. 1,** Satu Entitas hanya memiliki satu relasi pada entitas lain.
- 1 .. M,** Satu Entitas memiliki satu atau banyak relasi pada entitas yang lain dengan entitas yang sama.
- M .. N,** Satu Entitas memiliki satu atau banyak relasi pada entitas yang lain dengan entitas yang sama dan begitu juga sebaliknya.

Gambaran Cardinality pada Entity Relationship Diagram diperlihatkan pada contoh sebagai berikut :

MULTIPLICITY pada pemrograman berorientasi objek digunakan untuk menyatakan tingkatan asosiasi antar class. Biasanya tingkatan asosiasi ditunjukkan pada **CLASS DIAGRAM**. Tingkatan asosiasi mempunyai beberapa macam dan ditunjukkan pada tabel berikut :

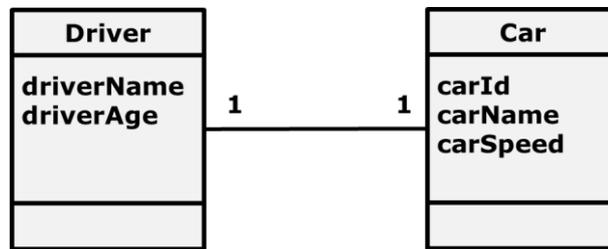
Indicator	Meaning	Example
0..1	Zero or one	
0..*	Zero or more	
0..n	Zero to n (where n>1)	0..3
1	One only	
1..*	One or more	
1..n	One to n (where n>1)	1..5
*	Many	
n	Only n (where n>1)	9
n..*	n or more, where n >1	7..*
n..m	Where n& m both >1	3..10

Adapun contoh MULTIPLICITY pada Class Diagram ditunjukkan sebagai berikut :



2. PENERAPAN ASOSIASI PADA JAVA

Berikut adalah contoh penerapan asosiasi pada java dengan tingkatan MULTIPLICITY adalah **1 .. 1** berikut dengan contoh class diagramnya.



Nama File : TransportCompany.java

Sintaks program :

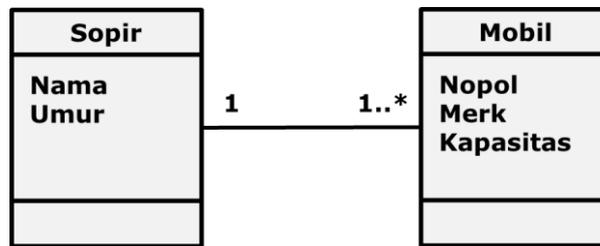
```
]class CarClass{
    int carId;
    String carName;
    double carSpeed;

] CarClass(String name, double speed, int Id)
{
    this.carName=name;
    this.carSpeed=speed;
    this.carId=Id;
- }
-}
]class Driver{
    String driverName;
    int driverAge;
] Driver(String name, int age){
    this.driverName=name;
    this.driverAge=age;
- }
-}
]class TransportCompany{
] public static void main(String args[])
{
    CarClass obj= new CarClass("Ford", 180.15, 9988);
    Driver obj2 = new Driver("Andy", 45);
    System.out.println(obj2.driverName+" is a driver of "+obj.carName+" with car Id: "+obj.carId);
- }
-}
```

Penjelasan Program :

- Pada class utama terdapat dua buah objek, masing-masing untuk class **CarClass** dan **Driver**. Statement menyatakan bahwa seorang driver/sopir mempunyai satu buah mobil berdasarkan car ID nya. Statement ini yang menyatakan asosiasi antar class adalah **1 .. 1**.

Berikut adalah contoh penerapan asosiasi pada java dengan tingkatan **MULTIPLICITY** adalah **1 .. *** berikut dengan contoh class diagramnya.



Nama File : JadwalSupir.java

Sintaks program :

```

class Sopir{
    String Nama;
    int Umur;
    Sopir(String name, int age){
        this.Nama=name;
        this.Umur=age;
    }
}

class Mobil{
    String Nopol;
    String Jenis;
    int Kapasitas;

    Mobil(String Id, String name, int kps)
    {
        this.Jenis=name;
        this.Kapasitas=kps;
        this.Nopol=Id;
    }
}

class JadwalSupir{
    public static void main(String args[])
    {
        Mobil mobil1 = new Mobil("BN 9978 CH", "Sedan", 4);
        Mobil mobil2 = new Mobil("BN 1234 HH", "MiniBus", 7);
        Sopir andi = new Sopir("Andy Haja", 45);

        System.out.print(andi.Nama+" adalah seorang sopir ");
        System.out.print("yang bertugas membawa mobil :\n");
        System.out.print("1. No. Polisi : "+mobil1.Nopol);
        System.out.print(", Jenis : "+mobil1.Jenis);
        System.out.print(", Kapasitas : "+mobil1.Kapasitas+"\n");
        System.out.print("2. No. Polisi : "+mobil2.Nopol);
        System.out.print(", Jenis : "+mobil2.Jenis);
        System.out.print(", Kapasitas : "+mobil2.Kapasitas+"\n");
    }
}
  
```

Penjelasan Program :

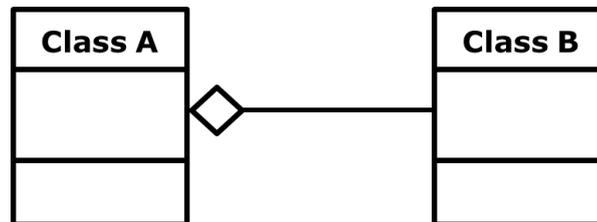
- Pada class utama terdapat tiga buah objek, masing-masing satu objek untuk class **Sopir** dan dua objek untuk class **Mobil**. Statement menyatakan bahwa seorang driver/sopir bertanggung jawab untuk dua buah kendaraan. Statement ini yang menyatakan asosiasi antar class adalah **1 .. *** .

3. PENGENALAN AGREGASI

Agregasi adalah salah satu bentuk asosiasi yang lain pada pemrograman berorientasi objek. Agregasi menyatakan bahwa salah satu class merupakan bagian dari class yang lain, tetapi walaupun class tersebut merupakan bagian dari class yang lain tetapi class tersebut dapat berdiri sendiri.

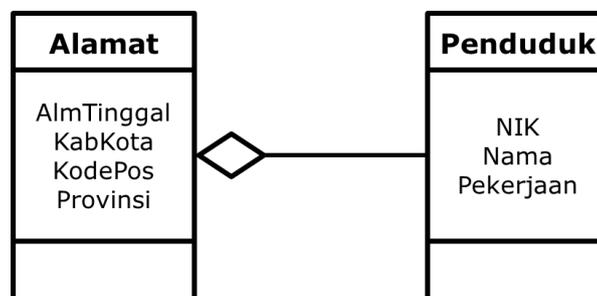
Agregasi sering disebut juga dengan relasi "**part of**" atau relasi "**whole-part**".

Pada class diagram, bentuk agregasi digambarkan sebagai berikut :



4. PENERAPAN AGREGASI PADA JAVA

Berikut contoh penggunaan agregasi pada java. Contoh menampilkan class **Alamat** dan class **Penduduk**. Agregasi yang dimaksudkan pada contoh berikut adalah Seorang penduduk pasti memiliki alamat, sehingga penduduk merupakan bagian (part of) alamat. Walaupun penduduk bagian dari alamat, tetapi data penduduk dapat dipergunakan tanpa melibatkan data alamat. Contoh proses ini yang dimaksud dengan sebuah class merupakan bagian dari class lain tetapi dapat berdiri sendiri.



Nama File : Agregasi.java

Sintaks program :

```
| class Alamat
| {
|     String AlmTinggal;
|     String KabKota;
|     String KodePos;
|     String Provinsi;
|     Alamat(String alm, String kab, String kode, String prov)
|     {
|         this.AlmTinggal=alm;
|         this.KabKota =kab;
|         this.KodePos = kode;
|         this.Provinsi = prov;
|     }
| }
| class Penduduk
| {
|     String NIK;
|     String Nama;
|     String Pekerjaan;
|     Alamat almt;
|     Penduduk(String NK, String Nm, String pkj, Alamat al){
|         this.NIK=NK;
|         this>Nama=Nm;
|         this.almt = al;
|     }
| }
| public class Agregasi
| {
|     public static void main(String args[])
|     {
|         Alamat pkp = new Alamat("Jln. Jendral Sudirman","Pangkalpinang","33117","Kepulauan Bangka Belitung");
|         Penduduk ani = new Penduduk("12345","Ani Rhoma","Karyawan Swasta",pkp );
|
|         System.out.println(" DATA PENDUDUK ");
|         System.out.println("-----");
|         System.out.println(" NIK      : "+ani.NIK);
|         System.out.println(" Nama    : "+ani>Nama);
|         System.out.println(" Alamat  : "+ani.almt.AlmTinggal);
|         System.out.println(" Kota    : "+ani.almt.KabKota);
|         System.out.println(" Provinsi : "+ani.almt.Provinsi);
|     }
| }
- }
```

TUJUAN :

- Dapat memahami mengenai konsep enkapsulasi
- Dapat memahami penggunaan enkapsulasi pada java

1. MEMAHAMI ENKAPSULASI

Enkapsulasi mempunyai beberapa nama. Ada yang menyebutnya pembungkusan, ada yang menyebutnya pengkapsulan dan ada pula yang menyebutnya access modifier. Mempunyai berbagai nama tetapi fungsinya adalah sama. Enkapsulasi adalah cara menyembunyikan implementasi detail sebuah class sehingga tidak bisa diakses sembarangan oleh class lainnya. Ada dua hal yang mendasar dari enkapsulasi ini adalah **INFORMATION HIDING** dan **INTERFACE TO ACCESS**.

Information Hiding adalah proses menyembunyikan data sebuah class sehingga tidak bisa diakses oleh class lain. Biasanya, data yang disembunyikan adalah atribut dari class, sehingga class yang lain tidak dapat mengetahui atribut apa saja yang dimiliki oleh sebuah class.

Interface to access adalah cara yang dilakukan sehingga dapat mengakses data yang telah disembunyikan. Maksudnya, yang dapat diakses adalah isi atributnya tanpa mengetahui apa saja atributnya, termasuk nama atributnya. Media yang digunakan pada cara ini biasanya adalah melalui sebuah method.

Ada 4 akses (modifier) yang tersedia untuk mengatur hak akses sebuah class. 4 akses tersebut dapat dilihat pada tabel berikut :

NO	MODIFIER	PADA CLASS DAN INTERFACE	PADA METHOD & ATRIBUT
1	Default (tak ada modifier)	Dapat diakses oleh yang sepaket.	Diwarisi oleh sub class di paket yang sama, dapat diakses oleh method-method yang sepaket
2	Public	Dapat diakses dimanapun	Diwarisi oleh sub classnya, dapat diakses dimanapun
3	Protected	Tidak bisa diterapkan	Diwarisi oleh sub classnya, dapat diakses oleh method-method yang sepaket.
4	Private	Tidak bisa diterapkan	Tidak dapat diakses dimanapun kecuali oleh method-method yang ada dalam kelas itu sendiri.

Apabila ditelusur dari kemampuan aksesnya, perbedaan 4 akses (modifier) tersebut dapat dilihat pada tabel berikut :

AKSESABILITAS	PUBLIC	PRIVATE	PROTECTED	DEFAULT
Dari kelas yang sama	Ya	Ya	Ya	Ya
Dari sembarang kelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari sembarang kelas di luar paket	Ya	Tidak	Tidak	Tidak
Dari sub kelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari sub kelas di luar paket	Ya	Tidak	Ya	Tidak

Khusus untuk 3 akses yaitu public, private dan protected sifatnya dapat diuraikan sebagai berikut :

- Private berkarakteristik agar variabel atau method pada sebuah object/class tidak dapat diakses oleh object/class yang lain.
- Protected berkarakteristik agar variabel atau method pada sebuah object/class dapat diakses oleh object/class turunannya, tetapi tidak dapat diakses oleh object/class yang lain.
- Public berkarakteristik agar variabel atau method pada sebuah object/class dapat diakses oleh object/class yang lain

Untuk akses (modifier) **Public** biasanya digunakan pada method, karena method berperan sebagai **Interface to Access**. Method sebagai jembatan untuk mengambil data / isi data dari sebuah atribut.

Untuk akses (modifier) **Private** biasanya digunakan pada atribut, karena berfungsi untuk menyembunyikan data atau **Information Hiding**.

Untuk akses (modifier) **Protected** biasanya digunakan pada atribut, sehingga atribut sebuah class dapat diakses oleh beberapa class lain. Class lain yang dapat mengakses atribut adalah class yang menjadi turunan dari class tersebut.

2. PENERAPAN ENKAPSULASI PADA JAVA

Berikut contoh penggunaan enkapsulasi pada program. Pada contoh berikut akses yang digunakan adalah **Public** dan **Private**. Untuk akses protected akan dibahas pada modul berikutnya, karena berkenaan dengan konsep turunan (inheritance).

Nama File : testEnkapsulasi.java

Sintaks program :

```
class mahasiswa
{
    private String Nama;

    public void setNama(String Nama)
    {
        this.Nama = Nama;
    }
    public String getNama()
    {
        return (Nama);
    }
}

class testEnkapsulasi
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        mhs1.setNama("Okki");

        System.out.println("Hai.. Saya "+mhs1.getNama());
        System.out.println("Salam Kenal ...");

    }
}
```

Penjelasan Program :

- Perhatikan class mahasiswa. Pendeklarasian atribut Nama sudah ditambahkan akses private. Method setNama() dan getNAMA() sudah ditambahkan akses public. Artinya class mahasiswa ini sudah mengatur hak akses untuk atribut dan methodnya.
- Pada class testEnkapsulasi, object mhs1 hanya dapat mengakses method saja dan tidak dapat mengakses atribut dari class mahasiswa. Method dapat diakses karena diberi akses public dan atribut tidak dapat diakses karena diberi akses private.

KONSTRUKTOR & DESTRUKTOR

TUJUAN :

- Dapat memahami mengenai konsep konstruktor dan destruktur
- Dapat memahami beberapa bentuk konstruktor
- Dapat memahami penggunaan konstruktor pada java

1. MEMAHAMI KONSTRUKTOR

Perhatikan contoh perintah instansiasi objek sebagai berikut :

```
mahasiswa mhs1 = new mahasiswa();
```

maksud perintah tersebut adalah :

- **mahasiswa** adalah nama sebuah class.
- **mhs1** adalah objek baru yang diciptakan untuk class mahasiswa.
- **new** adalah operator/keyword yang digunakan untuk membuat sebuah objek baru.
- **mahasiswa()** adalah sebuah **KONSTRUKTOR**. Dalam contoh ini adalah KONSTRUKTOR dari class mahasiswa.

Konstruktor merupakan sebuah method khusus yang secara otomatis dipanggil/dijalankan pada saat saat sebuah class diinstansi. Konstruktor dapat juga diartikan sebagai method khusus yang berfungsi untuk menciptakan objek dari sebuah class. Method konstruktor ini akan dipanggil secara otomatis oleh java ketika **new** digunakan untuk membuat sebuah objek.

Konstruktor dapat dibuat sendiri, tetapi apabila konstruktor tidak dibuat, maka compiler java akan secara otomatis membuat konstruktor default pada saat dilakukan kompilasi. Perlu untuk diketahui, bahwa konstruktor bukanlah anggota dari sebuah class, seperti atribut dan method.

Beberapa fungsi konstruktor diantaranya :

- Membuat sebuah objek pada saat program dieksekusi.
- Memberikan nilai awal sebagai dari atribut yang perlu diinisialisasi.
- Mengerjakan proses-proses yang diperlukan pada saat sebuah objek dibuat.

Beberapa hal yang diperlukan apabila ingin membuat konstruktor sendiri yaitu :

- nama konstruktor harus sama dengan nama classnya
- konstruktor tidak memiliki nilai kembalian/return value
- konstruktor tidak mempunyai / menggunakan **void** ataupun **tipe data**.
- konstruktor dapat memiliki satu atau banyak parameter, bahkan tanpa parameter sekalipun
- java tidak membatasi jumlah konstruktor pada satu class. Artinya sebuah class dapat memiliki lebih dari satu konstruktor.

Berdasarkan beberapa literatur, konstruktor pada java terdapat beberapa tipe/bentuk. Adapun bentuk/tipe dari konstruktor pada java yaitu :

- a. Default Constructor / Konstruktor Tunggal
- b. Parameterized Constructor / Konstruktor dengan parameter
- c. Overloading Constructor / Multiple Constructor
- d. Private Constructor
- e. Constructor Chaining / Rangkaian Konstruktor

2. PENERAPAN KONSTRUKTOR PADA JAVA

Berikut akan ditunjukkan contoh penggunaan dan pembuatan konstruktor pada java. Contoh program yang ditunjukkan mengikuti bentuk dari beberapa konstruktor.

- a. Default Constructor / Konstruktor Tunggal

Default Constructor adalah sebuah konstruktor yang tidak memiliki parameter dan hanya terdapat sebuah konstruktor pada sebuah class. Berikut contoh penerapan konstruktor tunggal pada pemrograman java.

Nama file : testKonsTunggal.java

Sintaks program :

```
class mahasiswa
{
    private String Nama;

    mahasiswa()
    {
        setNama("Okki");
    }
    public void setNama(String Nama)
    {
        this.Nama = Nama;
    }
    public String getNama()
    {
        return (Nama);
    }
}

class testKonsTunggal
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        System.out.println("Hai.. Saya "+mhs1.getNama());
        System.out.println("Salam Kenal ...");
    }
}
```

Penjelasan program :

- Konstruktor **mahasiswa()** tidak menggunakan awalan void ataupun tipe data.
- Pada konstruktor **mahasiswa()**, saat pembuatan objek dan dijalankan, konstruktor tersebut mengerjakan method `setNama()`. Sehingga objek pada class `main()` tidak perlu mengirimkan data ke class `mahasiswa`. Objek cukup memanggil method `getNama()` untuk menampilkan isi atribut.
- Pada contoh tersebut, konstruktor `mahasiswa()` berfungsi mendefinisikan data awal untuk atribut `Nama` menggunakan method `setNama()`.

b. Parameterized Constructor / Konstruktor dengan parameter

Parameterized Constructor adalah sebuah konstruktor yang memiliki parameter. Parameter adalah sebuah variabel yang berfungsi untuk menampung data yang dikirimkan melalui sebuah object. Berikut adalah contoh konstruktor yang menggunakan parameter pada pemrograman java.

Nama file : `testKonsParameter.java`

Sintaks program :

```
class mahasiswa
{
    private String Nama;

    mahasiswa(String Nm)
    {
        setNama(Nm);
    }
    public void setNama(String Nama)
    {
        this.Nama = Nama;
    }
    public String getNama()
    {
        return (Nama);
    }
}

class testKonsTunggal
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa("Okki");

        System.out.println("Hai.. Saya "+mhs1.getNama());
        System.out.println("Salam Kenal ...");
    }
}
```

Penjelasan Program :

- Konstruktor **mahasiswa()**, menggunakan sebuah parameter bernama Nm.
- Pada konstruktor **mahasiswa()**, saat pembuatan objek dan dijalankan, konstruktor tersebut mengirimkan data "Okki" ke parameter Nm. Lewat parameter Nm, konstruktor mengerjakan method setNama(). Objek pada class main() tidak perlu mengirimkan data ke class mahasiswa. Objek cukup memanggil method getName() untuk menampilkan isi atribut.
- Pada contoh tersebut, konstruktor mahasiswa() berfungsi mendefinisikan data awal untuk atribut Nama menggunakan method setNama() dengan menggunakan parameter.

c. Overloading Constructor / Multiple Constructor

Overloading Constructor adalah sebuah class yang memiliki lebih dari satu konstruktor yang mempunyai tugas yang berbeda. Overloading constructor dapat dipandang sebagai gabungan dari konstruktor tunggal (default constructor) dan konstruktor dengan parameter (parameterized constructor).

Berikut adalah contoh penerapan overloading constructor pada pemrograman java.

Nama file : testKonsMultiple.java

Sintaks program :

```
class mahasiswa
{
    private String Nama;

    mahasiswa()
    {
        setNama("Okki");
    }
    mahasiswa(String Nm)
    {
        setNama(Nm);
    }
    public void setNama(String Nama)
    {
        this.Nama = Nama;
    }
    public String getName()
    {
        return (Nama);
    }
}
```

```

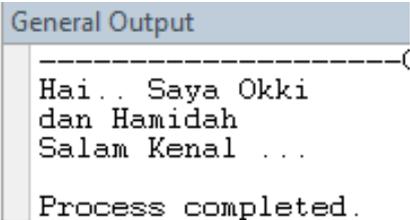
class testKonsMultiple
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();
        mahasiswa mhs2=new mahasiswa("Hamidah");

        System.out.println("Hai.. Saya "+mhs1.getNama());
        System.out.println("dan "+mhs2.getNama());
        System.out.println("Salam Kenal ...");

    }
}

```

Output program :



General Output

```

-----
Hai.. Saya Okki
dan Hamidah
Salam Kenal ...

Process completed.

```

Penjelasan program :

- Pada class main(), dibuat dua buah objek untuk class mahasiswa yaitu objek **mhs1** dan **mhs2**.
- Pada saat instansiasi objek mhs1, menggunakan konstruktor tunggal. Sedangkan saat instansiasi objek mhs2, menggunakan konstruktor parameter.

d. Private Constructor

e. Constructor Chaining / Rangkaian Konstruktor

Constructor Chaining adalah proses memanggil sebuah konstruktor dari konstruktor yang lain dalam satu buah class. Untuk memahami proses tersebut berikut adalah contoh penggunaan constructor chaining pada pemrograman java.

Nama file : chainingConstructor.java

Sintaks program :

```

class konstruktor
{
    konstruktor()
    {
        System.out.println("Hai..");
    }
    konstruktor(String str)
    {
        this();
        System.out.println("Nama Saya : "+str);
    }
    konstruktor(String str, int angk)
    {
        this("Okki");
        System.out.println("Mahasiswa "+str+" Angkatan "+angk);
    }
    konstruktor(int a, int b, int c)
    {
        this("STMIK Atma Luhur",14);
        System.out.println("Salam Kenal...");
    }
}

class chainingConstructor
{
    public static void main (String[] args)
    {
        konstruktor kons=new konstruktor(1,2,3);
    }
}

```

3. MEMAHAMI DESTRUKTOR

Destruktor merupakan kebalikan dari konstruktor. Konstruktor merupakan method khusus yang dijalankan secara otomatis pada saat objek diciptakan pada memory. Berbeda dengan destruktur, method ini akan dijalankan pada saat objek akan dihapus dari memory. Tujuannya adalah untuk membebaskan memory dari objek-objek yang tidak digunakan lagi.

Istilah destruktur sebenarnya dikenal dalam bahasa pemrograman C++. Pada bahasa Java tidak mengenal istilah destruktur. Hal ini dikarenakan objek dibuat secara dinamis pada saat run-time dan tidak secara eksplisit menulis perintah untuk menghapus suatu objek dari memory. Ketentuan apakah suatu objek akan dihapus dari memory atau tidak, ditentukan oleh Java sendiri. Proses ini dikenal dengan istilah **GARBAGE COLLECTION**.

Secara sederhana untuk menentukan apakah suatu objek akan dihapus dari memory, Java akan mengecek apakah masih ada yang memegang/menggunakan referensi ke objek tersebut. Jika tidak, berarti objek tersebut dapat dihapus dari memory sehingga memory yang terpakai dapat dipakai kembali untuk keperluan lainnya. Masalahnya

adalah proses garbage collection terjadi secara sporadis sehingga sulit untuk menentukan kapan proses ini akan berjalan. Akan tetapi, Java masih menyediakan perintah yang dapat digunakan untuk memaksa Java melakukan proses garbage collection ini, yaitu :

```
System.gc();
```

Kembali ke persoalan awal, Java tidak mengenal istilah destruktur, padahal ada kemungkinan pada saat membuat program perlu agar suatu code dieksekusi sebelum objek tersebut dihapus dari memory. Untuk mengatasi ini dapat menggunakan metode **finalize()**. Sesaat sebelum objek tersebut dihapus dari memory, Java run-time akan memanggil method **finalize()** dari objek tersebut. Bentuk pendeklarasiannya secara umum sebagai berikut :

```
protected void finalize()  
{  
    //implementasi  
}
```

Penggunaan akses protected untuk membatasi siapa saja yang boleh memanggil method ini karena method ini memang tidak seharusnya dipanggil oleh program yang dibuat, tetapi dipanggil oleh run-time Java sebelum objek dihapus dari memory.

Berikut contoh penggunaan desktruktor melalui proses garbage collection pada java.

Nama file : testDestruktor.java

Sintaks program :

```
class mahasiswa
{
    private int Jumlah;

    mahasiswa()
    {
        System.out.println("Konstruktor dijalankan");
    }
    public void setJumlah(int Jumlah)
    {
        this.Jumlah = Jumlah;
    }
    public int getJumlah()
    {
        return (Jumlah);
    }
    protected void finalize()
    {
        this.Jumlah=0;
        System.out.println("finalize dijalankan");
        System.out.println("Jumlah Siswa :"+this.Jumlah);
    }
}

class testDestruktor
{
    public static void main (String[] args)
    {
        mahasiswa jml1=new mahasiswa();
        mahasiswa jml2=new mahasiswa();

        jml1.setJumlah(10);
        jml2.setJumlah(20);
        System.out.println("Jumlah siswa kelas 1 : "+jml1.getJumlah());
        System.out.println("Jumlah siswa kelas 2 : "+jml2.getJumlah());
        jml1 = null;
        System.gc();
        System.out.println("Jumlah setelah gc :"+jml2.getJumlah());
    }
}
```

TUJUAN :

- Dapat memahami mengenai konsep Inheritance
- Dapat memahami penggunaan Inheritance pada java

1. MEMAHAMI INHERITANCE

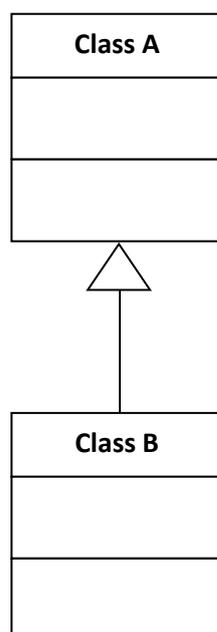
Nama lain dari Inheritance adalah Turunan atau Pewarisan. Inheritance adalah proses pewarisan data atau method dari suatu class yang sudah ada ke class yang baru. Pembuatan class baru dengan mengembangkan class yang sudah pernah dibuat sebelumnya.

Beberapa istilah dikenal pada proses inheritance diantaranya :

- Class yang menurunkan data atau method ke class lain disebut dengan super class, parent class, base class atau kelas induk.
- Class yang merupakan turunan dari kelas induk disebut dengan sub class, child class, derived class atau class turunan.

Class turunan adalah class yang menggunakan data atau method dari class induk. Konsepnya, secara otomatis class turunan memiliki sifat (variabel/atribut) dan kelakuan (behavior/method) yang sama yang dimiliki oleh kelas induknya. Class turunan dapat menambahkan fitur atau behavior dengan mendefinisikan suatu method didalam class turunan tersebut. Salah satu keuntungan dari konsep inheritance ini adalah cukup mendefinisikan satu kali saja atribut atau method yang sama pada class induknya dan dapat digunakan di seluruh class turunannya.

Pada Diagram UML (Unified Modelling Language), konsep Inheritance digambarkan pada class diagram sebagai berikut :

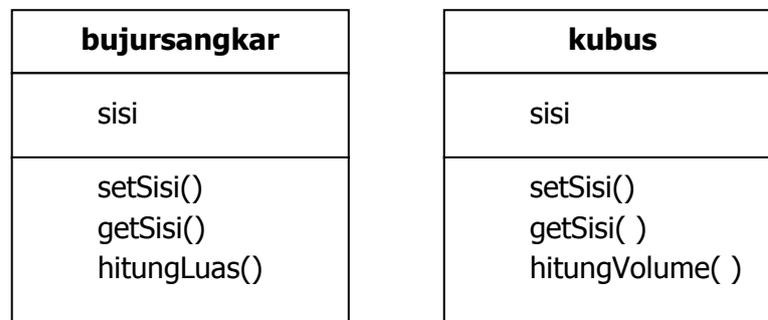


Contoh class diagram disamping menunjukkan bahwa Class A adalah class induk dari Class B, sebaliknya Class B adalah class turunan dari Class A. Panah terbuka menunjukkan bahwa kedua class tersebut menggunakan konsep Inheritance.

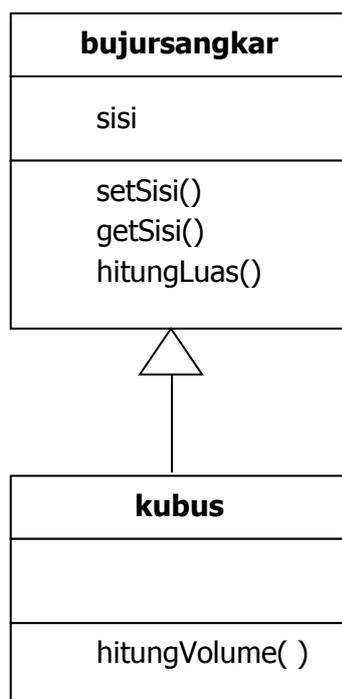
Manfaat dari konsep Inheritance dicontohkan sebagai berikut :

- Buatlah sebuah class untuk menangani objek bujursangkar. Class ini diberi dengan nama bujursangkar. Atribut yang dimiliki oleh class ini adalah sisi. Untuk memanipulasi atribut tersebut, buat dua buah method dengan nama setSisi() dan getSisi(). Selain kedua method tersebut, buat juga method untuk menghitung luas sebuah objek bujur sangkar. Method tersebut dinamai dengan hitungLuas().
- Buatlah sebuah class untuk menangani objek kubus. Class ini diberi dengan nama kubus. Atribut yang dimiliki oleh class ini adalah sisi. Untuk memanipulasi atribut tersebut, buat method dua buah method dengan nama setSisi() dan getSisi(). Selain kedua method tersebut, buat juga method untuk menghitung volume sebuah objek kubus. Method tersebut dinamai dengan hitungVolume().

Dari kedua soal tersebut, apabila dibuat dengan class diagram dapat digambarkan sebagai berikut :



Apabila diperhatikan, maka kedua class tersebut memiliki kesamaan atribut dan dua method yang sama. Sehingga kedua class ini dapat diterapkan konsep Inheritance yang digambarkan sebagai berikut :



Dengan konsep Inheritance ini, pada class kubus tidak perlu lagi dibuatkan atribut dan method yang sama dengan class bujursangkar.

class kubus dapat menggunakan atribut dan method yang sama pada class bujursangkar.

Pada pemrograman, bentuk ini dapat membantu programmer untuk meminimalkan jumlah baris perintah pada saat pembuatan program.

2. PENERAPAN INHERITANCE PADA JAVA

Pada pemrograman, apabila sebuah class merupakan class turunan dari class induknya ditunjukkan dengan keyword **extends**. Keyword extends digunakan pada pendeklarasian classnya. Bentuk umumnya sebagai berikut :

```
namasubclass extends namasuperclass
{
    //statements
}
```

Selain extends, terdapat keyword lain yang digunakan pada konsep Inheritance. Keyword tersebut diantaranya :

- **super**. Keyword super digunakan oleh subclass untuk memanggil/ menggunakan konstruktor, atribut atau method pada superclassnya.
- **protected**. Keyword protected adalah hak akses / access modifier. Fungsi access modifier sudah dijelaskan pada modul sebelumnya. Atribut atau method yang menggunakan akses protected, akan dapat digunakan pada class turunan.

Berikut contoh penerapan konsep Inheritance dalam pemrograman. Contoh berikut adalah implementasi pada contoh class diagram pada point sebelumnya.

Nama file : testInheritance.java

Sintaks program :

```
class bujursangkar
{
    protected int sisi ;
    public void setSisi (int s)
    {
        sisi = s;
    }
    public int getSisi()
    {
        return(sisi);
    }
    public int hitungLuas()
    {
        return(sisi*sisi);
    }
}

class kubus extends bujursangkar
{
    public int hitungVolume()
    {
        return (super.getSisi()*super.getSisi()*super.sisi);
    }
}
```

```

public class testInheritance
{
    public static void main (String args[])
    {
        bujursangkar objsangkar = new bujursangkar();
        kubus objkubus = new kubus();

        objsangkar.setSisi(10);
        objkubus.setSisi(20);

        System.out.println("          B U J U R S A N G K A R          ");
        System.out.println("-----");
        System.out.println(" Sisi          : "+objsangkar.getSisi());
        System.out.println(" Luas          : "+objsangkar.hitungLuas());
        System.out.println("=====");
        System.out.println();
        System.out.println("          K U B U S          ");
        System.out.println("-----");
        System.out.println(" Sisi          : "+objkubus.getSisi());
        System.out.println(" Volume       : "+objkubus.hitungVolume());
        System.out.println("=====");

    }
}

```

Output Program :

```

-----Configuration: <1
          B U J U R S A N G K A R
-----
Sisi          : 10
Luas          : 100
=====

          K U B U S
-----
Sisi          : 20
Volume       : 8000
=====

```

Penjelasan Program :

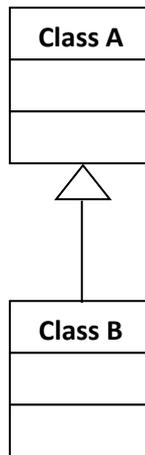
- class kubus merupakan class turunan dari class bujursangkar
- atribut sisi pada class bujursangkar menggunakan akses protected, sehingga atribut tersebut dapat digunakan di class kubus pada method hitungVolume(). Perhatikan pada method tersebut, selain menggunakan method getSisi(), juga menggunakan atribut sisi untuk menghitung volume kubus.
- Keyword super pada method hitungVolume(), menunjukkan bahwa method pada class kubus menggunakan method dan atribut dari class supernya yaitu class bujursangkar.

3. MACAM-MACAM TURUNAN (TYPE OF INHERITANCE)

Konsep Turunan memiliki beberapa bentuk. Bentuk dari turunan tersebut berdasarkan hubungan antara super class dan sub class. Macam-macam turunan tersebut adalah sebagai berikut :

a. Single Inheritance

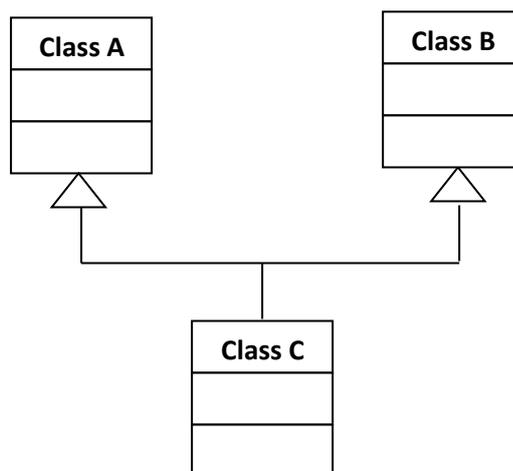
Sebuah turunan disebut dengan single inheritance apabila sebuah superclass hanya memiliki satu buah subclass. Ilustrasi single inheritance digambarkan sebagai berikut :



Contoh penerapan inheritance pada java dapat dilihat pada contoh program testInheritance.java di atas.

b. Multiple Inheritance

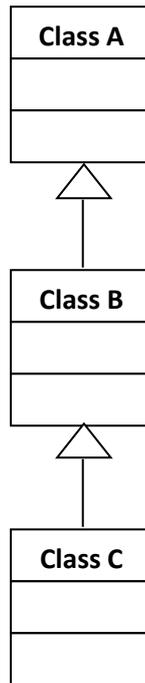
Konsep dari Multiple Inheritance adalah apabila ada sebuah sub class memiliki lebih dari satu super class. Adapun Multiple Inheritance di ilustrasikan sebagai berikut :



Berdasarkan berbagai literatur, tidak banyak bahasa pemrograman yang mendukung konsep Multiple Inheritance ini, termasuk diantaranya adalah Bahasa Pemrograman Java.

c. Multilevel Inheritance

Multilevel Inheritance terjadi apabila sebuah class menjadi sub class dari sebuah super class, tetapi kelas tersebut juga menjadi super class dari class yang lain. Ilustrasinya dapat dilihat pada gambar berikut :



Berikut adalah contoh penerapan Multilevel Inheritance pada Bahasa Pemrograman Java.

Nama Program :

Sintaks Program :

```
class A
{
    protected String nama;

    public void setNama (String Nama)
    { this.nama = Nama; }

    public String getNama ()
    { return (nama); }

    public void tampil ()
    {
        System.out.println("Salam Kenal....");
    }
}
```

```

class B extends A
{
    public void hello()
    {
        System.out.println("Hai...");
    }
}

class C extends B
{
    public void data()
    {
        System.out.println("Nama Saya : "+super.getNama());
    }
}

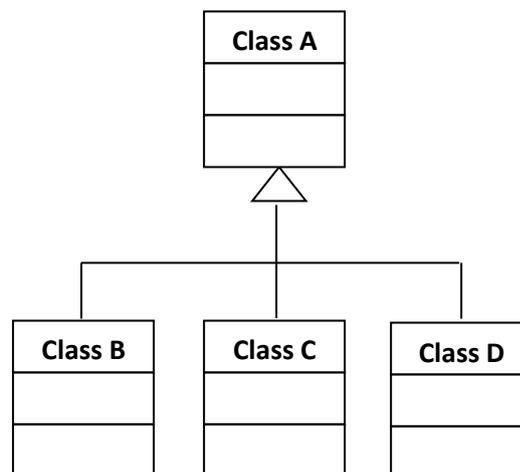
public class multiInheritance
{
    public static void main (String[] args)
    {
        C mhs1 = new C();

        mhs1.setNama("Okki");
        mhs1.hello();
        mhs1.data();
        mhs1.tampil();
    }
}

```

d. Hierarchical Inheritance

Bentuk turunan ini adalah apabila satu buah class yang menjadi super class memiliki lebih dari satu sub class. Ilustrasinya dapat dilihat pada contoh berikut :



Berikut adalah contoh penerapan Hierarchical Inheritance pada Bahasa Pemrograman Java.

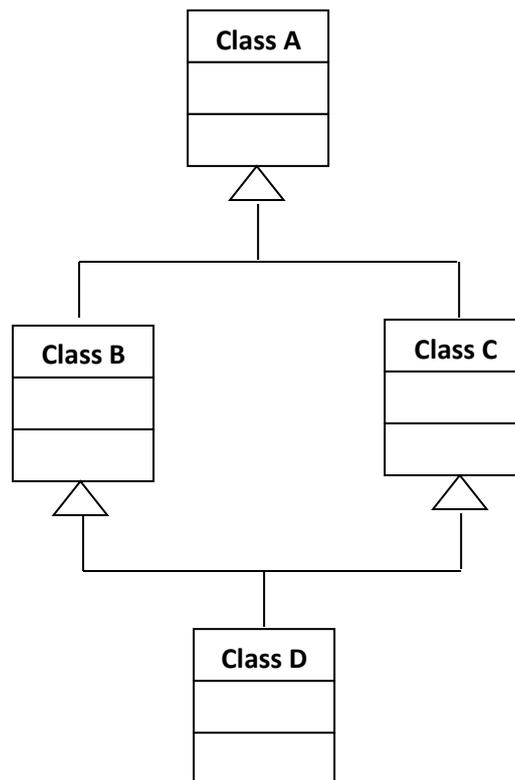
Nama Program : hierarchical.java

Sintaks Program :

```
| class atribut
| {
|     protected int sisi;
|
|     public void setSisi(int S)
|     { sisi = S ;}
|     public int getSisi()
|     { return (sisi);}
| }
|
| class bujursangkar extends atribut
| {
|     private int luas;
|
|     public int hitungLuas()
|     {
|         luas = super.getSisi() * super.getSisi();
|         return (luas);
|     }
| }
|
| class kubus extends atribut
| {
|     private int volume;
|
|     public int hitungVolume()
|     {
|         volume = super.getSisi()*super.getSisi()*super.getSisi();
|         return (volume);
|     }
| }
|
| public class hierarchical
| {
|     public static void main (String[] args)
|     {
|         bujursangkar objsangkar = new bujursangkar();
|         kubus objkubus = new kubus();
|
|         objsangkar.setSisi(10);
|         objkubus.setSisi(5);
|
|         System.out.println("          B U J U R S A N G K A R          ");
|         System.out.println("-----");
|         System.out.println(" Sisi          : "+objsangkar.getSisi());
|         System.out.println(" Luas          : "+objsangkar.hitungLuas());
|         System.out.println("=====");
|         System.out.println();
|         System.out.println("          K U B U S          ");
|         System.out.println("-----");
|         System.out.println(" Sisi          : "+objkubus.getSisi());
|         System.out.println(" Volume       : "+objkubus.hitungVolume());
|         System.out.println("=====");
|     }
| }
}
```

e. Hybrid Inheritance

Hybird Inheritance merupakan turunan yang merupakan kombinasi dari Single dan Multiple Inheritance. Untuk memahami konsep turunan ini, berikut ilustrasi dari Hybrid Inheritance.



Seperti yang telah dijelaskan di atas, Java tidak mendukung konsep Multiple Inheritance. Walaupun tidak mendukung konsep turunan tersebut, Java menyediakan cara lain untuk mengatasi bentuk pemrograman yang menyerupai Multiple Inheritance, yaitu menggunakan konsep **INTERFACE**. Untuk konsep interface akan dibahas pada modul berikutnya.

TUJUAN :

- Dapat memahami mengenai konsep polymorphism
- Dapat memahami mengenai konsep overloading
- Dapat memahami mengenai konsep overriding
- Dapat memahami penggunaan overloading pada java
- Dapat memahami penggunaan overriding pada java

1. MEMAHAMI POLYMORPHISM

Polymorphism atau polimorfisme dimaknai sebagai banyak bentuk. Dalam pemrograman berorientasi objek, konsep ini memungkinkan penggunaan interface yang sama untuk keperluan atau hasil berbeda, ini yang dimaksud dengan banyak bentuk. Interface sama tetapi mempunyai hasil yang berbeda.

Ada beberapa keuntungan konsep polymorphism diterapkan didalam program diantaranya :

- Dapat menghindari duplikasi objek, dengan cara menciptakan class baru dari class yang sudah ada, sehingga tidak perlu menuliskan code dari nol ataupun mengulanginya, namun tetap bisa menambahkan atribut dan atau method unik dari class itu sendiri. Dalam konsep yang lebih umum sering kali polymorphism disebut dalam istilah satu interface dengan banyak aksi.
- Dapat menggunakan class-class yang dibuat sebagai super class dan membuat class-class baru berdasarkan super class tersebut dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki oleh super class.
- Dapat membuat super class yang hanya mendefinisikan behaviour namun tidak memberikan implementasi dari method-method yang ada. Hal ini berguna apabila ingin membuat semacam template class, class semacam ini disebut dengan abstrak class. Pembahasan ini akan dibahas secara khusus pada modul berikutnya.

Pada pemrograman berorientasi objek, interface yang dapat diterapkan konsep polymorfisme adalah method. Atribut sendiri tidak bisa diterapkan konsep polymorfisme. Sehingga, polymorfisme adalah konsep yang memungkinkan penggunaan method yang sama dengan hasil yang berbeda. Ada dua penyebutan nama method yang diterapkan konsep polymorfisme sesuai dengan bentuk penggunaannya. Dua method tersebut disebut dengan :

METHOD OVERLOADING dan METHOD OVERRIDING

Lebih jauh mengenai perbedaan kedua method tersebut dan contoh penggunaannya pada java akan diuraikan dibawah ini.

2. MEMAHAMI OVERLOADING

Perhatikan penggalan baris perintah pada class berikut ini :

```
mahasiswa()  
{  
    setNama("Okki");  
}  
mahasiswa(String Nm)  
{  
    setNama(Nm);  
}
```

Penggalan baris perintah tersebut menunjukkan terdapat dua buah konstruktor yaitu konstruktor **mahasiswa()** pada sebuah class. Apabila diperhatikan, maka kedua konstruktor tersebut memiliki nama yang sama, yaitu sama-sama dengan nama mahasiswa. Hal inilah yang membedakan antara teknik pemrograman terstruktur dengan teknik pemrograman berorientasi objek.

Pada pemrograman terstruktur, tidak diizinkan apabila dalam satu program terdapat lebih dari satu method dengan nama yang sama. Berbeda dengan pemrograman berorientasi objek, diperbolehkan mempunyai lebih dari satu method dengan yang sama, dengan syarat perlakuan berbeda. Cara ini disebut dengan **OVERLOADING**.

Selain pada konstruktor, konsep overloading dapat diterapkan pada beberapa method pada sebuah class dengan syarat perlakuan berbeda. Perlakuan berbeda dari setiap method ditunjukkan dengan jumlah parameter digunakan pada setiap methodnya. Selain dengan jumlah parameternya, tipe data yang tidak sama yang digunakan pada parameter juga bisa dilakukan konsep overloading walaupun jumlah parameternya sama.

3. PENERAPAN OVERLOADING PADA JAVA

Berikut adalah contoh penggunaan konsep overloading pada sebuah class.

Pada contoh berikut class mahasiswa menggunakan 4 buah method dengan nama yang sama yaitu method **tampil()**.

Nama file : testOverloading.java

Sintaks program :

```

class mahasiswa
{
    private String Nama;
    private int Tahun;

    public void tampil()
    {
        System.out.println("Hai..");
    }
    public void tampil(String Nama)
    {
        this.Nama = Nama;
        System.out.println("Kami Mahasiswa "+this.Nama);
    }
    public void tampil(int Tahun)
    {
        this.Tahun = Tahun;
        System.out.println("Angkatan "+this.Tahun);
    }
    public void tampil (String Nm1, String Nm2)
    {
        System.out.println("Saya "+Nm1+" dan "+Nm2);
        System.out.println("Salam Kenal..");
    }
}

class testOverloading
{
    public static void main (String[] args)
    {
        mahasiswa mhs1=new mahasiswa();

        mhs1.tampil();
        mhs1.tampil("STMIK ATMA LUHUR");
        mhs1.tampil(2002);
        mhs1.tampil("Okki", "Hamidah");
    }
}

```

Output Program :

```

General Output
-----Configuratic
Hai..
Kami Mahasiswa STMIK ATMA LUHUR
Angkatan 2002
Saya Okki dan Hamidah
Salam Kenal..

Process completed.

```

4. MEMAHAMI OVERRIDING

Method Overriding biasanya diterapkan pada konsep Inheritance atau turunan. Sesuai dengan fungsinya, yang dimaksud dengan overriding adalah penggunaan method sub class sama dengan method super class. Pengertian yang lain adalah, pada sub class, nama sebuah method maupun parameter method nya sama dengan nama dan parameter sebuah method pada super class. Tetapi hasil akhirnya/perlakuannya berbeda.

Inilah yang membedakan antara method Overloading dengan method Overriding. Pada method overloading, sebuah class mempunyai lebih dari satu method dengan nama yang sama tetapi parameternya yang berbeda, yang menyebabkan hasil akhirnya/perlakuannya akan berbeda.

5. PENERAPAN OVERRIDING PADA JAVA

Berikut adalah contoh penerapan method Overriding pada konsep Inheritance pada java. Pada super class maupun sub classnya akan dibuatkan method dengan nama yang sama. Method yang di Overriding yaitu method **tampil()**.

Nama file : testOverriding.java

Sintaks Program :

```
| class judul
| {   protected String Nama1, Nama2;
|     public void setNama (String Nama1, String Nama2)
|     {
|         this.Nama1 = Nama1;
|         this.Nama2 = Nama2;
|     }
|
|     public String getNama1 ()
|     {
|         return (Nama1);
|     }
|     public String getNama2 ()
|     {
|         return(Nama2);
|     }
|     public void tampil ()
|     {
|         System.out.println("Hai..");
|         System.out.println("Kami Mahasiswa STMIK Atma Luhur");
|         System.out.println("Angkatan 2002");
|     }
| }
```

```

class nama extends judul
{
    public void tampil()
    {
        System.out.print("Saya "+super.getNama1());
        System.out.println(" dan "+super.getNama2());
        System.out.println("Salam Kenal..");
    }
}

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

        judul jdl=new judul();
        nama nm = new nama();

        nm.setNama("Okki","Hamidah");
        jdl.tampil();
        nm.tampil();
    }
}

```

Output Program :

```

General Output
-----Configuration
Hai..
Kami Mahasiswa STMIK Atma Luhur
Angkatan 2002
Saya Okki dan Hamidah
Salam Kenal..

Process completed.

```

TUJUAN :

- Dapat memahami mengenai konsep Interface
- Dapat memahami penggunaan Interface pada java

1. MEMAHAMI INTERFACE

Interface adalah sekumpulan deklarasi konstanta dan atau method tanpa memiliki implementasi dan semua property yang dimilikinya bersifat final. Interface mirip dengan class abstrak, tetapi interface tidak terikat dengan class hierarki. Interface biasa digunakan untuk mendeklarasikan kumpulan method dan konstanta yang dapat digunakan oleh satu atau lebih class.

Bentuk umum dari pendeklarasian interface adalah sebagai berikut :

```
interface namainterface  
{  
    //deklarasi method dan konstanta  
}
```

Beberapa aturan yang harus diperhatikan mengenai pendeklarasian interface adalah :

- Modifier/access modifier yang digunakan hanya public atau tidak sama sekali. Apabila tidak menggunakan modifier, maka interface tersebut hanya dapat diakses dalam package yang sama.
- Semua variabel yang dideklarasikan dalam interface, secara otomatis adalah static final. Sehingga, pada saat pendeklarasian harus diberikan nilai.
- Semua method adalah abstrak. Bedanya dengan class abstrak adalah tidak perlu menuliskan keyword abstract pada saat mendeklarasikan method dalam interface.
- Dapat mengimplementasikan lebih dari satu interface (multiple inheritance) dengan memisahkan nama dari setiap interface dengan tanda koma. Apabila saat mengimplementasikan lebih dari satu interface ternyata interface-interface tersebut memiliki method yang sama, maka method yang akan digunakan adalah method yang berada pada posisi pertama.
- Semua method yang diimplementasikan harus public. Apabila tidak mengimplementasikan semua method yang ada pada interface, maka interface tersebut harus dideklarasikan sebagai abstract class. Abstract class akan dibahas pada modul berikutnya.

2. PENERAPAN INTERFACE PADA JAVA

Pada bahasa Java, diperlukan keyword yang menunjukkan bahwa sebuah class mengimplementasikan sebuah atau beberapa interface. Keyword tersebut adalah **implements** dan disebutkan pada saat pendeklarasian sebuah class.

Berikut adalah contoh penerapan interface pada program. Contoh berikut menunjukkan sebuah class mengimplementasikan sebuah interface, yaitu interface **cetak**.

Nama file : testInterface.java

Sintaks program :

```
interface cetak
{
    public void tampil();
}

class judul implements cetak
{
    public void tampil()
    {
        System.out.println("Hai..");
        System.out.println("Kami Mahasiswa STMIK Atma Luhur");
    }
}

class nama implements cetak
{
    public void tampil()
    {
        System.out.println("Saya Okki dan Hamidah ");
        System.out.println("Salam kenal");
    }
}

public class testInterface
{
    public static void main (String args[])
    {
        judul jdl=new judul();
        nama nm = new nama();

        jdl.tampil();
        nm.tampil();
    }
}
```

TUJUAN :

- Dapat memahami macam-macam class
- Dapat memahami perbedaan dan kegunaan penerapan class pada java

Pada modul ini, secara khusus akan membahas berbagai bentuk dari class. Bentuk dari berbagai macam class ini akan menunjukkan perbedaan fungsi atau kegunaan maupun bentuk dari tiap tiap class. Untuk beberapa class yang sudah dibahas pada modul sebelumnya, tidak dicantumkan contoh penerapannya dalam pemrograman, hanya definisinya saja dalam modul ini. Berikut beberapa macam class dalam pemrograman berorientasi objek.

1. SUPER CLASS

Class ini digunakan pada konsep Inheritance. Nama lain dari class ini adalah parent class atau base class atau kelas induk. Pada Inheritance class ini akan mewariskan apa saja yang dimilikinya ke class turunannya. Pada pemrograman, yang diwarisi adalah atribut dan methodnya sehingga dapat digunakan di class turunannya. Contoh penerapan class ini sudah diuraikan pada modul sebelumnya.

2. SUB CLASS

Class ini digunakan juga pada konsep Inheritance. Nama lain dari class ini adalah child class atau derived class atau kelas turunan . Pada Inheritance class ini dapat menggunakan atribut dan method dari kelas induknya. Class ini dapat menambahkan atribut dan method sendiri. Pada pemrograman, class ini tidak dapat berdiri sendiri dan dipasangkan dengan class yang lain, yaitu class induknya (super class). Contoh penerapan class ini sudah diuraikan pada modul sebelumnya.

3. ABSTRACT CLASS

Sama dengan dua class diatas, class ini juga digunakan pada konsep Inheritance. Peran dari abstract class ini sama dengan super class, yaitu sebagai acuan bagi class-class turunan. Bedanya dengan super class, abstract class biasanya hanya berisi variabel-variabel umum (atribut umum) dan header-header method saja tanpa body methodnya atau bisa juga sebagian method ada body methodnya dan sebagian method yang lain tidak memiliki body method. Sehingga, class-class turunannya yang akan mendefinisikan secara detail body method-method tersebut.

Bentuk umum dari pendeklarasian abstract class adalah sebagai berikut :

```
akses_modifier abstract class namaabstract
{
    // statements
}
```

Java membuat beberapa aturan dalam penggunaan abstract method dan class abstract yaitu sebagai berikut :

- Class yang didalamnya terdapat abstract method harus dideklarasikan sebagai abstract class.
- Abstract class tidak dapat diinstansi, tetapi harus diturunkan.
- Abstract class tidak dapat diinstansi (menjadi objek dari class abstract), tetapi dapat mendeklarasikan suatu variabel yang bertipe abstract class dan membuat instansi dari variabel tersebut yang bertipe class turunan dari abstract class tersebut (teknik polymorphism).
- Sebuah class dapat dideklarasikan sebagai abstract class meskipun class tersebut tidak mempunyai abstract method.
- Abstract method tidak mempunyai body method dan demikian juga sebaliknya bahwa method yang tidak ditulis body methodnya maka harus dideklarasikan sebagai abstract method

Berikut adalah contoh penerapan abstract class pada pemrograman. Pada contoh berikut juga diperkenalkan mengenai abstract method, method yang mempunyai body method dan static method yang dibuat didalam abstract class.

Nama file : testAbstractClass.java

Sintaks Program :

```
abstract class bangun
{
    protected int sisi;

    abstract void setSisi(int s);

    void TampilLuas (int luas)
    {
        System.out.println("Luas : "+luas);
    }
    void TampilVolume (int volume)
    {
        System.out.println("Volume : "+volume);
    }
    static void Tutup()
    {
        System.out.println("Selesai ..");
    }
}
```

```

class bujursangkar extends bangun
{
    public void setSisi (int s)
    {
        super.sisi = s;
    }

    public int hitungLuas()
    {
        return(super.sisi*super.sisi);
    }
}

class kubus extends bangun
{
    public void setSisi (int s)
    {
        super.sisi = s;
    }
    public int hitungVolume()
    {
        return (super.sisi*super.sisi*super.sisi);
    }
}

public class testAbstractClass
{
    public static void main (String args[])
    {
        bujursangkar objsangkar = new bujursangkar();
        kubus objkubus = new kubus();

        objsangkar.setSisi(2);
        objkubus.setSisi(4);

        objsangkar.TampilLuas(objsangkar.hitungLuas());
        objkubus.TampilVolume(objkubus.hitungVolume());
        bangun.Tutup();
    }
}

```

Output Program :

```

General Output
-----
Luas : 4
Volume : 64
Selesai ..
Process completed.

```

4. NESTED CLASS

Istilah nested sudah tidak asing lagi didengar pada algoritma. Istilah ini digunakan pada struktur percabangan dan struktur perulangan. Apabila pada struktur percabangan disebut dengan NESTED IF, sedangkan di struktur perulangan disebut dengan NESTED LOOP. Maksud dari nested adalah bersarang, sehingga yang dimaksud dengan NESTED IF adalah percabangan bersarang dan NESTED LOOP adalah perulangan bersarang.

Selain pada algoritma, pada konsep object oriented dikenal juga dengan istilah nested. Pada objek oriented, nested digunakan untuk menunjukkan bentuk class yang bersarang. Dengan maksud lain adalah terdapat class didalam class yang lain. Sehingga namanya adalah **NESTED CLASS**.

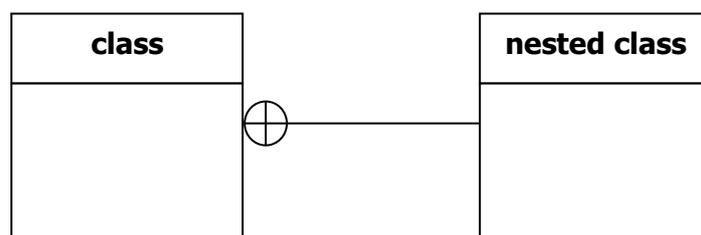
Nested class adalah class yang didefinisikan didalam class yang lain. Pada dasarnya, Nested Class terbagi kedalam dua kategori yaitu Static Class dan Non Static Class. Untuk Static Class, disebut dengan **STATIC NESTED CLASS**, sedangkan Non Static Class, disebut juga dengan **INNER CLASS**. Class yang didalamnya terdapat Static Nested Class dan atau Inner Class disebut dengan **OUTER CLASS**. Dengan maksud lain, bahwa Static Nested Class dan Inner Class merupakan anggota dari Outer Class.

Outer Class dan Nested Class saling berhubungan. Artinya, sebuah class tidak dapat dikatakan sebagai Outer Class apabila didalamnya tidak terdapat class lain (Static Nested Class dan atau Inner Class), begitu juga sebaliknya, sebuah class tidak dapat dikatakan sebagai nested class (Static Nested Class dan atau Inner Class) apabila class tersebut tidak berada didalam class yang lain (Outer Class).

Adapun bentuk umum dari nested class dicontohkan sebagai berikut :

```
class OuterClass
{
    //statements
    class NestedClass
    {
        //statements
    }
}
```

Dalam diagram UML, nested class digambarkan sebagai berikut :



a. Static Nested Class

Pada sebagian literatur menyebutkan bahwa Static Nested Class adalah bagian dari Inner Class. Pada dasarnya, Static Nested Class adalah hanyalah top-level class yang disarangkan didalam class lain. Static Nested Class adalah "Inner Class" yang memiliki modifier static.

Dikarenakan class ini menggunakan static, maka pada saat membuat instance/objek dari static nested class ini tidak diperlukan instance/objek dari outer classnya. Ini lah yang membedakan antara Static Nested Class dengan Inner Class. Sebuah Static Nested Class tidak dapat mengakses member/atribut non static dari Outer Classnya.

Adapun cara atau proses menginstansiasi Static Nested Class seperti ditunjukkan sebagai berikut :

**NamaOuterClass>NamaStaticNestedClass ObjectNestedClass = new
NamaOuterClass>NamaStaticNestedClass()**

Berikut contoh penerapan Static Nested Class pada pemrograman.

Nama file : testStaticNested.Java

Sintaks program :

```
class mahasiswa
{
    static class baru
    {
        void tampil()
        {
            System.out.println("Hai.. Saya Mahasiswa");
            System.out.println("Salam Kenal ...");
        }
    }
}

class testStaticNested
{
    public static void main (String[] args)
    {
        mahasiswa.baru mhs = new mahasiswa.baru();
        mhs.tampil();
    }
}
```

Output Program :

```
General Output
-----Configur
Hai.. Saya Mahasiswa
Salam Kenal ...

Process completed.
```

Penjelasan Program :

- class baru adalah sebuah class yang bertipe Static Nested Class. Outer class dari class baru adalah class mahasiswa.
- Objek mhs adalah objek dari class baru, bukan objek dari class mahasiswa. Sehingga pada saat instansiasi objek perlu menyebutkan class mahasiswa sebagai outer class dari class baru.

b. Inner Class

Inner class merupakan bagian/member dari class yang lain, Outer Classnya. Sehingga Inner Class dapat mengakses seluruh atribut ataupun method dari outer classnya.

Bentuk umum dari Inner Class ini adalah :

```
class namaOuterClass
{
    class namaInnerClass
    {
        // statement
    }
}
```

Berdasarkan struktur dan sifat dari classnya, Inner Class terbagi kedalam :

- "Regular" Inner Class
- Method-Local Inner Class
- Anonymous Inner Class

Pada beberapa literatur menambahkan satu lagi, jenis dari Inner Class yaitu :

- Static Nested Class

Untuk Static Nested Class sudah dibahas sebelumnya.

1) "Regular" Inner Class

Inner Class jenis ini merupakan Inner Class pada umumnya, artinya bukan method-local atau anonymous Inner Class, bukan juga Static Nested Class.

Beberapa hal dan aturan dari Inner Class ini adalah :

- Objek/instance dari Inner Class dapat mengakses semua member/atribut dari Outer Classnya, termasuk juga atribut dengan modifier private pada Outer Classnya.
- Sebuah Inner Class tidak dapat memiliki member dengan modifier static.
- Untuk membuat objek/instance dari Inner Class, maka harus terdapat objek/instance dari Outer Class terlebih dahulu, tidak ada pengecualian untuk aturan ini.

Mengenai aturan Inner Class dalam mereferensi dirinya sendiri atau objek/instance dari Outer Class adalah sebagai berikut :

- Untuk merujuk pada dirinya sendiri (instance/objek atau member dari Inner Class) dari dalam Inner Class, dapat menggunakan referensi this atau OuterClass.InnerClass.this.
- Untuk merujuk pada instance/member dari Outer Classnya dari dalam Inner Class, dapat digunakan referensi OuterClass.this.

Mengenai pembuatan objek/instance dari Inner Class, terdapat dua cara yaitu :

- Dari dalam Outer Class
- Dari luar Outer Class.

Untuk membuat objek/Instance dari dalam Outer Class, bentuk umumnya adalah sebagai berikut :

```
class NmOuterClass
{
    void NmMethod()
    {
        NmInnerClass NmObjek = new NmInnerClass();
    }
    class NmInnerClass
    {
        // statements
    }
}
```

Sedangkan untuk membuat Objek/Instance dari luar Outer Class, bentuk umumnya adalah sebagai berikut :

```
NamaOuterClass NmOuter = new NamaOuterClass();
NamaOuterClass>NamaInnerClass NmInner = NmOuter.new NamaInnerClass();
```

Berikut contoh penerapan "Regular" Inner Class dengan contoh pembuatan objek/instance dari dalam dan luar Outer Classnya.

Nama File : testRegulerInner.Java

Sintaks Program :

```
class mahasiswa
{
    void kenalan()
    {
        baru b = new baru();
        System.out.println("Hai.. Saya Mahasiswa");
        b.tampil();
    }

    class baru
    {
        void tampil()
        {
            System.out.println("Salam Kenal ...");
        }
    }
}

class testRegularInner
{
    public static void main (String[] args)
    {
        mahasiswa mhs = new mahasiswa();
        mahasiswa.baru br = mhs.new baru();
        br.tampil();
        mhs.kenalan();
    }
}
```

Output Program :

```
General Output
-----Conf
Salam Kenal ...
Hai.. Saya Mahasiswa
Salam Kenal ...
```

Penjelasan Program :

- Pada class mahasiswa dibuat sebuah method dengan nama kenalan(). Method tersebut dapat menggunakan method dari class baru karena dibuatkan objek untuk class baru.
- Objek dengan nama **b** dibuat didalam Outer Classnya, sedang objek dengan nama **br** dibuat diluar Outer Classnya.

2) Method-Local Inner Class

Method-Local Inner Class adalah sebuah Inner Class yang dideklarasikan didalam sebuah method. Pada sebagian literatur Method-Local Inner Class disebut juga dengan Local Class.

Beberapa hal yang perlu diketahui mengenai Method-Local Inner Class ini diantaranya :

- Objek dari Method-Local Inner Class ini hanya dapat diinstansiasi dari dalam method yang mendefinisikan/mendeklarasikan Method-Local Inner Class tersebut.
- Objek/instance dari Method-Local Inner Class dapat mengakses seluruh member dari Outer Classnya, termasuk member yang memiliki akses private.
- Objek/instance dari Method-Local Inner Class tidak dapat mengakses local variabel, termasuk juga parameter dari method dimana method-local inner class tersebut didefinisikan, kecuali bila variabel tersebut bermodifier final.
- Mendeklarasikan Method-Local Inner Class tidak berarti membuat objek/instance diluar dari method tempat mendeklarasikan class tersebut. Sehingga, sebelum menggunakan Inner Class tersebut, harus membuat objeknya dari suatu tempat didalam method dan setelah definisi inner class tersebut.
- Method-Local Inner Class yang didefinisikan didalam static method tidak dapat mengakses member non static dari Outer Classnya.

Berikut adalah contoh penerapan Method-Local Inner Class pada pemrograman :

Nama File : testMethodLocal.java

Sintaks Program :

```
class mahasiswa
{
    void tampil()
    {
        class baru
        {
            void kenalan()
            {
                System.out.println("Hai.. Saya Mahasiswa");
                System.out.println("Salam Kenal..");
            }
        }
        baru b = new baru();
        b.kenalan();
    }
}
```

```

class testMethodLocal
{
    public static void main (String[] args)
    {
        mahasiswa mhs = new mahasiswa();
        mhs.tampil();
    }
}

```

Output Program :

```

Hai.. Saya Mahasiswa
Salam Kenal..

Process completed.

```

Penjelasan Program :

- Pada contoh tersebut class baru adalah sebuah class berjenis Method-Local Inner Class. Yang menjadi Outer Classnya adalah class mahasiswa. Class tersebut didefinisikan/dideklarasikan didalam method **tampil()**.
- Class baru mempunyai objek bernama **b** yang diinstansiasi didalam didalam method **tampil()**.
- Objek dari class mahasiswa yang bernama **mhs** menggunakan method tampil, secara tidak langsung menjalankan method **kenalan()** karena sudah dibuat objek dari class baru didalam class mahasiswa.

3) Anonymous Inner Class

Anonymous Inner Class adalah sebuah Inner Class yang dideklarasikan tanpa nama class. Penerapan Anonymous Inner Class adalah pasti dari dua hal berikut ini :

- Menjadi sub class dari suatu class yang telah dideklarasikan
- Menjadi Class yang mengimplementasi sebuah interface.

Beberapa hal yang perlu diketahui dari Anonymous Inner Class ini adalah :

- Suatu Anonymous Inner Class tidak dapat secara bersamaan menjadi subclass dari class yang telah dideklarasikan dan juga menjadi class yang mengimplementasi suatu interface.
- Tujuan utama dari Anonymous Inner Class adalah mengoverride satu atau lebih method dari super classnya atau mengimplementasi semua method dari suatu interface.
- Anonymous Inner Class tidak dapat mengimplementasi lebih dari sebuah interface.
- Anonymous Inner Class selalu dibuat sebagai bagian dari suatu statement.

- Anonymous Inner Class adalah salah satu bentuk dari polymorphisme, oleh karena itu, method yang dapat dipanggil dari Anonymous Inner Class adalah method yang dideklarasikan di super class atau interfacenya, meskipun di dalam Anonymous Inner Class dapat dideklarasikan method-method yang tidak ada di super class atau interfacenya.
- Anonymous Inner Class dapat diletakkan sebagai argument dari satu method.

Berikut adalah contoh penerapan Anonymous Inner Class pada pemrograman :

Nama File : testAnonymous.java

Sintaks program :

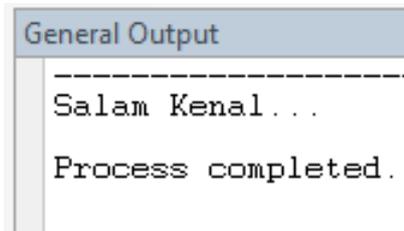
```

class mahasiswa
{
    void tampil()
    {
        System.out.println("Hai.. Saya Mahasiswa");
    }
}

class testAnonymous
{
    public static void main (String[] args)
    {
        mahasiswa mhs = new mahasiswa() {
            void tampil()
            {
                System.out.println("Salam Kenal...");
            }
        };
        mhs.tampil();
    }
}

```

Output Program :



General Output

```

-----
Salam Kenal...
Process completed.

```

Penjelasan Program :

- Perhatikan pada **class main()**, terdapat sebuah objek dengan nama **mhs**.
- mhs adalah variabel referensi yang berfungsi sebagai objek Anonymous Inner Class yang merupakan sub class dari class mahasiswa. Jadi objek mhs bukan mereferensi objek dari class mahasiswa.

5. FINAL CLASS

Final class adalah sebuah class yang mempunyai keyword final pada nama classnya. Class dengan jenis ini tidak dapat menjadi super class bagi class lain. Artinya class ini tidak dapat diturunkan/digunakan atribut atau methodnya untuk class lain. Apabila sebuah class dirasa tidak perlu diturunkan ke class yang lain, misalnya class tersebut dirasa sudah memiliki atribut dan method yang sempurna, maka gunakanlah keyword final.

Berikut adalah contoh penerapan Final Class pada pemrograman

Nama File : testFinalClass.java

Sintaks program :

```
final class mahasiswa
{
    void tampil()
    {
        System.out.println("Hai.. Saya Mahasiswa");
        System.out.println("Salam Kenal...");
    }
}

class testFinalClass
{
    public static void main (String[] args)
    {
        mahasiswa mhs = new mahasiswa();
        mhs.tampil();
    }
}
```

TUJUAN :

- Dapat memahami mengenai Class JFrame
- Dapat memahami atribut dan method pada Class JFrame
- Dapat menerapkan Class JFrame dalam pemrograman

1. MEMAHAMI CLASS JFrame

Class JFrame adalah sebuah class yang sudah disiapkan oleh JDK untuk dipergunakan dalam pembuatan aplikasi. Artinya class ini bisa langsung digunakan oleh para programmer dalam pembuatan aplikasi.

Adapun tugas dari Class JFrame ini adalah untuk membuat sebuah frame. Frame adalah sebuah objek yang merepresentasikan suatu area dilayar yang berisi sejumlah objek lain untuk menyampaikan informasi kepada user. Pada frame dapat ditambahkan sejumlah komponen lain dengan tujuan yang sesuai dengan fungsi komponen tersebut. Bagi para pembuat program, nama lain dari frame bisa disebut dengan window atau form. Apabila sebuah aplikasi sudah terdapat form, maka aplikasi tersebut sudah berbasis GUI atau Graphical User Interface.

Class JFrame adalah class yang berfungsi membuat Frame. Sehingga, dapat disimpulkan bahwa class JFrame adalah class utama yang diperlukan untuk membuat aplikasi berbasis GUI. Tanpa Class JFrame, sebuah aplikasi yang dibuat tidak dapat menjadi aplikasi berbasis GUI.

Pada Java Development Kit (JDK), tersedia beberapa package yang berisi Class-Class dan komponen pendukung GUI. Pada modul praktikum ini, akan menggunakan dua buah package. Kedua package tersebut adalah **AWT** dan **SWING**. Perbedaan kedua package ini adalah lokasi tempat penyimpanan packagenya. Package AWT terdapat pada **java.awt**, sedangkan SWING terdapat pada **javax.swing** dan package SWING memiliki komponen yang lebih banyak dari AWT. Beberapa komponen dari SWING adalah button, combo box, check box, label, list, scrollbar, textfield, radio button, option pane, progress bar, tabel, menu dan text area. Setiap komponen memiliki fungsi dan cara pembuatan yang berbeda. Semua komponen pada swing diawali dengan huruf "J", misalkannya JButton, JTextArea. Sehingga, dapat diketahui bahwa Class JFrame terdapat pada package SWING.

2. PENERAPAN METHOD CLASS JFrame DALAM PEMROGRAMAN

Berikut adalah contoh pembuatan frame menggunakan Class JFrame. Untuk menggunakan Class JFrame, maka perlu untuk menyebutkan/menambahkan lokasi penyimpanan package dari Class JFrame tersebut. Dalam hal ini package javax.swing. Pada bahasa Java, keyword yang digunakan untuk menambahkan package kedalam program adalah **import**.

Pada pembuatan frame diperlukan beberapa method dari Class JFrame untuk membuat satu buah frame yang utuh. Untuk lebih jelasnya akan diuraikan satu persatu method yang diperlukan dalam pembuatan Frame. Satu persatu, contoh penggunaan method akan melengkapi method sebelumnya, sehingga menjadi satu buah frame yang utuh.

a. Penggunaan method show()

Method ini berfungsi untuk menampilkan sebuah frame ketika program dijalankan. Selain method ini, terdapat method lain yang berfungsi untuk menampilkan frame. Method tersebut adalah method setVisible(). Berikut adalah contoh program menggunakan method show().

Nama Program : framedasar.java

Sintaks Program :

```
import javax.swing.JFrame;

public class framedasar
{
    public static void main (String args[])
    {
        JFrame objJFrame = new JFrame();

        objJFrame.show();
    }
}
```

Output Program :



Penjelasan Program :

- import javax.swing.JFrame adalah perintah untuk memanggil class JFrame dari package Swing.
- JFrame objJFrame = new JFrame() adalah perintah mendeklarasi objek dari class JFrame. Objek yang dibuat bernama objJFrame.
- objJFrame.show() adalah perintah untuk menampilkan Frame.

b. Penggunaan Konstruktor Class JFrame

Judul pada sebuah frame berguna untuk menunjukkan kegunaan dari aplikasi yang dibuat secara umum. Melalui Class JFrame, sebuah judul dapat dibuat menggunakan konstruktor dari class tersebut. Berikut adalah contoh programnya.

Nama Program : judulframe.java

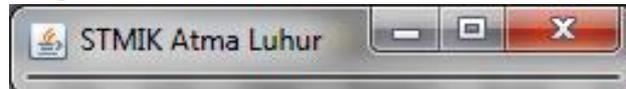
Sintaks Program :

```
import javax.swing.JFrame;

public class judulframe
{
    public static void main (String args[])
    {
        JFrame objJFrame = new JFrame("STMIK Atma Luhur");

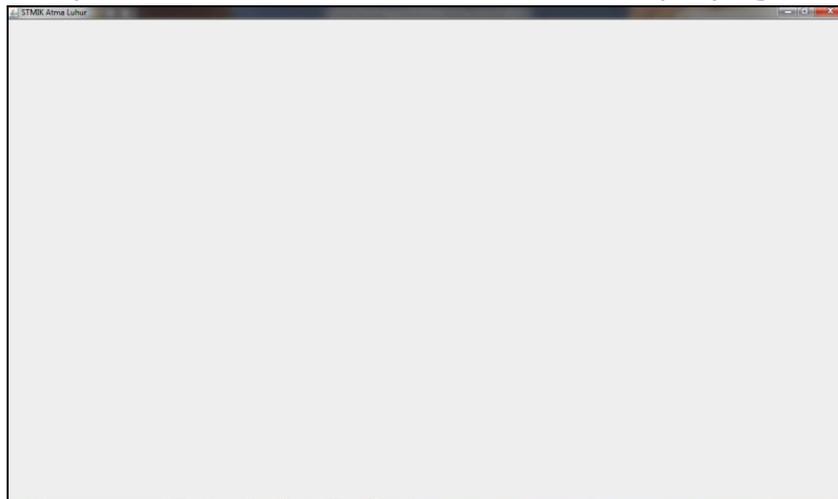
        objJFrame.show();
    }
}
```

Output Program :

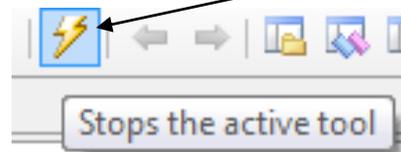


Penjelasan Program :

- Melalui konstruktor dari Class JFrame, dibuat judul dari framenya dengan nama "STMIK Atma Luhur".
- Apabila tombol maximize  diklik, maka output program menjadi :



- c. Penggunaan method `setDefaultCloseOperation()` dan method `EXIT_ON_CLOSE()`
Berdasarkan contoh program pada point a dan b, frame yang sudah dibuat dan dijalankan, frame tidak dihapus dari memori computer, walaupun sudah menutupnya dengan mengklik tombol close. Dengan kata lain, frame tersebut masih berjalan tetapi tidak kelihatan. Sehingga apabila ingin meng*compile* ulang program, user harus mengklik icon **Stops the active tool**.



Hal ini disebabkan karena java memberlakukan method **HIDE_ON_CLOSE** pada saat sebuah windows di close. Melalui method ini, begitu ditutup, frame tidak dikeluarkan dari memory (`exit`), tetapi disembunyikan dari pandangan(`hide`).

Berikut beberapa metode `setDefaultCloseOperation(int)`, yang digunakan untuk menentukan aksi pada saat sebuah frame ditutup. Java menyediakan empat konstanta yang bisa dipilih, untuk aksi yang dilakukan :

- `DO_NOTHING_ON_CLOSE` (didefinisikan di `WindowConstant`)
Menyatakan bahwa tidak ada proses yang akan dilakukan. Dengan kata lain frame tersebut tidak ditutup. Hal ini pada suatu kondisi perlu dimana frame dicegah dari tindakan ditutup tanpa sengaja.
- `HIDE_ON_CLOSE` (didefinisikan di `WindowConstant`)
Menyatakan bahwa frame akan dihilangkan dari layar, bukan dari memori dan merupakan nilai default dari `setDefaultCloseOperation()`.
- `DISPOSE_ON_CLOSE` (didefinisikan di `WindowConstant`)
Menyatakan bahwa frame akan dihilangkan dari layar dan dari memori computer.
- `EXIT_ON_CLOSE` (didefinisikan di `JFrame`)
Menyatakan bahwa penutupan frame ini akan menyebabkan keseluruhan program ditutup dan dihapus dari memori. Gunakan konstanta ini pada tempat yang tepat, misalnya kita membuat program yang membuka beberapa frame, maka jangan gunakan pada sembarang frame tersebut. Gunakan pada frame yang benar-benar akan menutup dan kembali kesistem.

Pada uraian tersebut, method yang diperlukan agar sebuah frame dikeluarkan dari memory komputer adalah method `EXIT_ON_CLOSE()`.

Berikut contoh program menggunakan method `EXIT_ON_CLOSE()` melalui method `setDefaultCloseOperation()`.

Nama program : hapusframe.java

Sintaks program :

```
import javax.swing.JFrame;

public class hapusframe
{
    public static void main (String args[])
    {
        JFrame objJFrame = new JFrame("STMIK Atma Luhur");

        objJFrame.show();
        objJFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

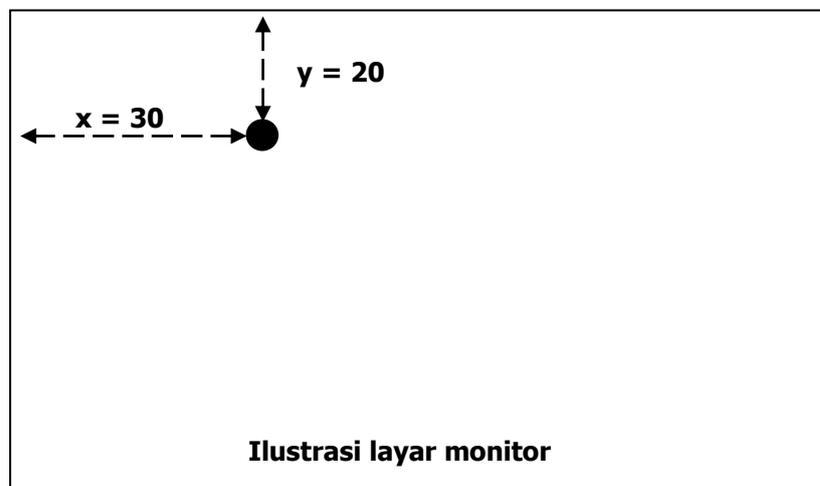
- Output program :



d. Penggunaan method setLocation()

Pada contoh sebelumnya, ketika program dijalankan frame akan ditampilkan pada posisi pojok kiri atas pada layar. Hasil yang diperoleh seperti itu dikarenakan frame diberikan pada posisi defaultnya yaitu pada koordinat 0,0.

Koordinat sebuah objek biasanya menggunakan x dan y. Penulisan titik koordinat diwakili dengan (x,y). Pada layar monitor, titik x menunjukkan posisi objek dari kiri ke kanan sedangkan y menunjukkan posisi objek dari atas ke bawah. Sebagai contoh apabila objek mendapat koordinat (30,20), artinya objek tersebut berada pada posisi 30 titik dari kiri dan 20 titik dari atas. Diilustrasikan dengan gambar sebagai berikut :



Dari uraian tersebut dapat dijelaskan bahwa, apabila sebuah frame mendapatkan posisi defaultnya yaitu 0,0, maka frame tersebut berada pada koordinat 0 (nol) baik secara horizontal maupun secara vertikal, sehingga frame tersebut akan berada pada pojok kiri atas pada layar.

Class JFrame menyediakan method untuk mengatur posisi frame pada layar. Salah satunya adalah method setLocation(). Untuk menggunakan method ini diperlukan dua buah parameter. Parameter ini lah yang berfungsi untuk menentukan poisisi frame baik secara horizontal maupun secara vertikal. Berikut contoh penggunaan method setLocation() dalam pemrograman.

Nama program : posisiframe.java

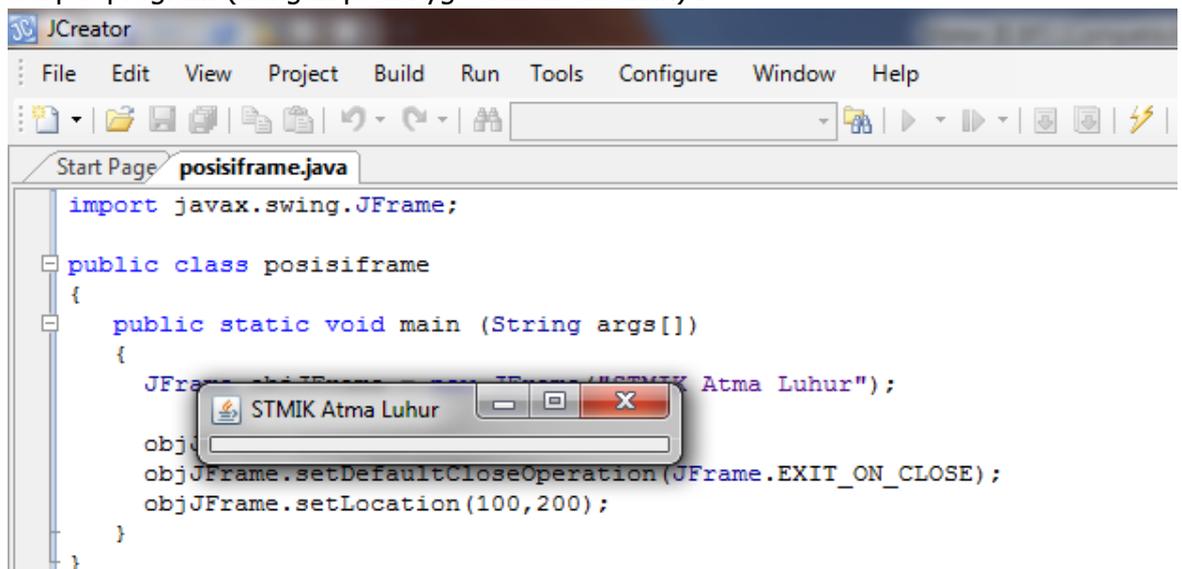
Sintaks program :

```
import javax.swing.JFrame;

public class posisiframe
{
    public static void main (String args[])
    {
        JFrame objJFrame = new JFrame("STMIK Atma Luhur");

        objJFrame.show();
        objJFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        objJFrame.setLocation(100,200);
    }
}
```

Output program (dengan posisi yg telah ditentukan) :

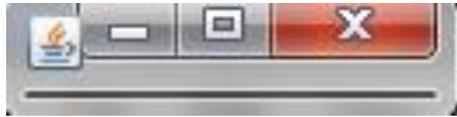


Penjelasan program :

- objJFrame.setLocation(100,200) adalah perintah menentukan lokasi frame pada layar monitor, artinya 100 titik dari kiri frame dan 200 titik dari atas frame (darikiri,dariatas)

e. Penggunaan method `setSize()`

Perhatikan bentuk frame yang dibuat berdasarkan contoh program diatas. Frame yang dihasilkan sebagai berikut :



Pada contoh tampilan frame tersebut menunjukkan bahwa frame tersebut belum mempunyai ukuran, sehingga frame tersebut tidak dapat menampung berbagai komponen, contohnya class `JLabel`.

Pada dasarnya, walaupun frame tidak mempunyai ukuran, frame tersebut dapat menampung sebanyak apapun komponen, tetapi komponen tersebut tidak dapat ditampilkan karena area/tempat untuk menampilkan komponen tidak ada. Sehingga tidak ada manfaat sama sekali komponen dimasukkan kedalam frame yang tidak mempunyai ukuran.

Untuk mengatasinya, Class `JFrame` sudah menyediakan method untuk menentukan ukuran dari sebuah frame. Salah satu methodnya adalah method `setSize()`. Penggunaan method ini memerlukan dua buah parameter untuk menentukan ukuran framenya, lebar (`width`) dan tinggi (`height`) frame. Satuan ukuran frame sama dengan satuan posisi frame, yaitu `pixel`. Berikut adalah contoh program yang menggunakan method `setSize()`.

Nama File : `ukuranframe.java`

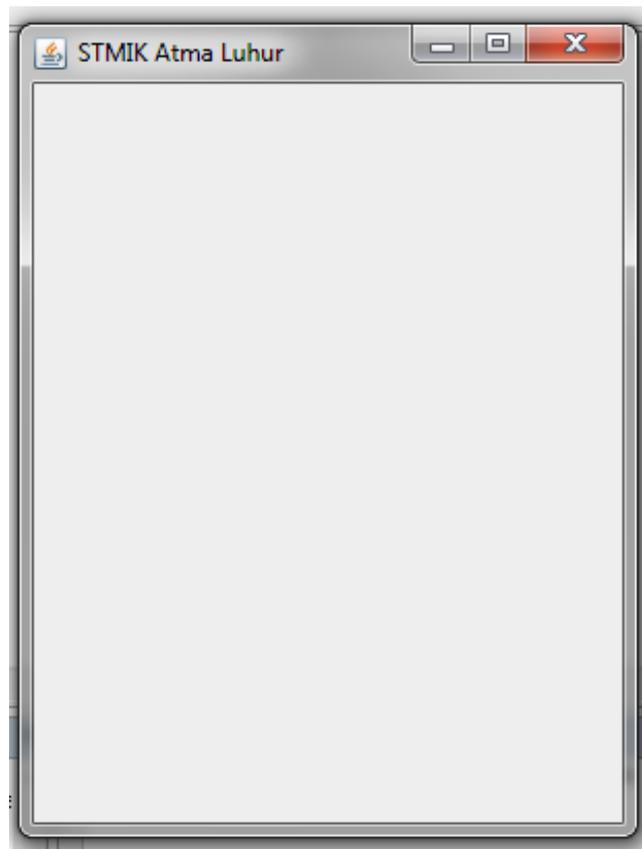
Sintaks Program :

```
import javax.swing.JFrame;

public class ukuranframe
{
    public static void main (String args[])
    {
        JFrame objJFrame = new JFrame("STMIK Atma Luhur");

        objJFrame.show();
        objJFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        objJFrame.setLocation(100,200);
        objJFrame.setSize(300,400);
    }
}
```

Output Program :



Penjelasan Program

- `objJFrame.setSize(300,400)` adalah perintah untuk menentukan ukuran frame, artinya 300 pixel secara horizontal dan 400 pixel secara vertikal.

3. PENERAPAN KONSEP TURUNAN PADA CLASS JFrame

Class JFrame bukan merupakan class yang bertipe Final Class. Artinya class tersebut masih dapat diturunkan ke class yang lain.

Pada contoh berikut class JFrame diturunkan ke class yang lain sehingga menghasilkan sebuah frame. Class tersebut akan menggunakan keyword `extends` yang menunjukkan bahwa class tersebut adalah class turunan dari Class JFrame.

Pada class yang menjadi turunan dari class JFrame, akan menggunakan method yang diperlukan pada sebuah frame. Method-method yang akan digunakan ditempatkan pada konstruktor pada class tersebut.

Menggunakan konsep ini, class yang dibuat cukup dipanggil dalam bentuk **anonymous objek** pada class `main()`nya (class utamanya). Output/tampilan frame menggunakan konsep ini sama dengan contoh program sebelumnya.

Nama Program : ukuranframe2.java

Sintaks Program :

```
import javax.swing.JFrame;

class clsframe extends JFrame
{
    clsframe()
    {
        super("STMIK Atma Luhur");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocation(100,200);
        setSize(300,400);
        show();
    }
}

public class ukuranframe2
{
    public static void main (String args[])
    {
        new clsframe();
    }
}
```

Penjelasan Program :

- class clsframe extends JFrame adalah perintah membuat sebuah class yang bernama clsfram yang merupakan turunan dari class JFrame. Keyword **extends** menandakan class tersebut turunan dari class lain.
- Pada Class clsframe, method yang diperlukan ditempatkan didalam **konstruktor**. Ciri dari konstruktor adalah memiliki nama yang sama dengan nama classnya. Pada contoh tersebut konstruktor yang digunakan adalah konstruktor tanpa parameter.
- super ("STMIK Atma Luhur") menunjukkan perintah tersebut menggunakan atribut dan method dari class induknya. Pada Class JFrame, perintah tersebut berfungsi untuk membuat judul pada frame.
- new clsframe() adalah perintah untuk memanggil class clsframe. Ini adalah contoh dari **Anonymous Object**, yaitu object dari sebuah class yang tidak mempunyai nama.

TUJUAN :

- Dapat memahami package dari Class-Class yang berbasis GUI
- Dapat memahami mengenai Class-Class yang berbasis GUI
- Dapat memahami atribut dan method pada Class-Class yang berbasis GUI
- Dapat menerapkan Class-Class yang berbasis GUI dalam pemrograman.

1. PENGENALAN PACKAGE AWT DAN SWING

Java Development Kit (JDK) menyediakan dua buah package yang dapat digunakan untuk membangun objek GUI, yaitu package AWT dan SWING. Perbedaan dari kedua package ini adalah :

- Lokasi penyimpanan kedua package ini berbeda pada library. Package AWT terdapat pada **java.awt** sedangkan package SWING terdapat pada **javax.swing**.
- Nama dari komponen atau class yang terdapat pada package SWING diawali dengan huruf "j". Berbeda dengan package AWT, nama dari komponen atau class tidak mempunyai ciri tertentu.

AWT merupakan singkatan dari Abstract Windows Toolkit yang merupakan tampilan dasar dari setiap target platform, seperti windows, macintosh, solaris atau berdiri sendiri sesuai dengan mekanisme sebuah platform.

Pada pembuatan sebuah program yang simpel, package AWT bagus untuk menjawab kebutuhan tersebut, seperti slogan dari bahasa Java yaitu Write once run everywhere. Tetapi pada saat membuat sebuah program yang lebih kompleks AWT menjadi tidak bagus dan tidak berlaku pada tujuan Java (write once, debug everywhere). Hal ini karena sulit diimplementasikan dan harus sesuai dengan target platform.

Sebagai contoh, class GUI seperti Button, Textfield dan Scrollbar memiliki tingkah laku (behaviour) yang berbeda di setiap platform. Pada AWT library class GUI juga memiliki bug yang berbeda di setiap platform.

Kekurangan ini diantisipasi dengan hadirnya package SWING yang menawarkan tampilan yang lebih kaya dan bagus. SWING tidak berdasarkan platform yang dituju tetapi menggunakan metode "Painted" yaitu, setiap objek dari class GUI digambar ke dalam window/frame kosong. Sehingga setiap objek dari class GUI akan tampil dan berperilaku sama di setiap platform.

Package SWING bukanlah pengganti dari package AWT dan package SWING tidak dapat berdiri sendiri, karena setiap program butuh berkomunikasi dengan mekanisme dasar dari target platform seperti Event Handling yang ada di package AWT. Jadi, package SWING akan selalu berdampingan dengan package AWT.

2. CLASS GUI PADA PACKAGE SWING

Berbagai class GUI didalam java merupakan turunan dari class JComponent. Sesuai dengan namanya yang diawali dengan huruf "j" class JComponent terdapat didalam package SWING.

Berikut beberapa Class GUI turunan dari Class JComponent yang terdapat pada package SWING.

a. Class JLabel

Versi lamanya bernama Label yang terdapat dalam package AWT. Class JLabel digunakan untuk merepresentasikan suatu tulisan di frame.

Sebuah label ditujukan sebagai media informasi bagi user, bukan sebagai media interaksi dengan user. JLabel tidak bisa mengartikan sesuatu sebagai respon didalam sebuah label. JLabel hanya bisa membaca informasi yang tertulis diatasnya.

Bentuk deklarasi class JLabel adalah sebagai berikut :

```
JLabel Nama_Objek = new JLabel (String, ImageIcon, Format)
```

Contoh :

```
JLabel lblNama = new JLabel ("Nama :");
```

Beberapa methode yang dimiliki oleh class JLabel adalah :

- 1) getText() = mendapatkan teks pada JLabel
- 2) setForeground(Color) = mengubah warna tulisan JLabel
- 3) setFont(Font) = mengubah model huruf pada JLabel
- 4) setHorizontalAlignment(int) = mengatur perataan tampilan label
(LEFT, CENTER, RIGHT)
- 5) setText(String) = mengubah string pada JLabel

b. Class JTextField

Versi lamanya bernama TextField yang terdapat didalam package AWT. JTextField digunakan untuk menerima inputan dari user, pada saat komputer berinteraksi dengan User. Objek ini biasanya dipasangkan dengan objek label pada saat melakukan interaksi dengan user.

Bentuk deklarasi class JTextField adalah sebagai berikut :

```
JTextField Nama_Objek = new JTextField (text_default);
```

Contoh :

```
JTextField txtnim = new JTextField("0222300005");
```

Beberapa methode yang dimiliki oleh class JTextField adalah sebagai berikut :

- 1) getText() = mendapatkan string pada JTextField
- 2) selectAll() = memblok seluruh string pada JTextField
- 3) setFont(Font) = mengubah model huruf pada JTextField

- 4) `setEnabled(boolean)` = dapat diakses (true) atau tidak (false)
- 5) `setForeground(color)` = mengubah warna font pada JTextField
- 6) `setText(String)` = mengubah string pada JTextField
- 7) `setToolTipText()` = menampilkan tulisan saat mouse ditempelkan

c. Class JButton

Versi lamanya bernama Button yang terdapat didalam package AWT. JButton digunakan untuk melakukan suatu proses jika user mengkehendakinya. Bukan merupakan proses yang akan terjadi secara otomatis.

Bentuk deklarasi class JButton adalah sebagai berikut :

```
JButton Nama_Objek = new JButton (String);
```

Contoh :

```
JButton tombolOK = new JButton ("OK");
```

Beberapa method yang dimiliki oleh class JButton adalah sebagai berikut :

- 1) `setEnabled(boolean)` = bisa diakses atau tidak.
- 2) `setVisible(boolean)` = bisa terlihat oleh user atau tidak.
- 3) `setText(String)` = mengubah string yang ditampilkan pada button.
- 4) `setFont(Font)` = mengganti model huruf yang digunakan.
- 5) `setForeground(Color)` = mengubah warna pada text button.

d. Class JCheckBox

Versi lamanya bernama CheckBox yang terdapat didalam package AWT. JCheckBox digunakan untuk memilih minimal satu dari sekian banyak pilihan yang disediakan untuk user. Sebagai contoh memilih hobi, hobi bisa saja lebih dari satu.

Bentuk deklarasi class JButton adalah sebagai berikut :

```
JCheckBox Nama_Objek = new JCheckBox (String);
```

Contoh :

```
JCheckBox chkBaca = new JCheckBox ("Hobi Membaca");
```

Beberapa method yang dimiliki oleh class JCheckBox adalah sebagai berikut :

- 1) `setEnabled(Boolean)` = bisa diakses atau tidak.
- 2) `isSelected()` = true bila dipilih dan false sebaliknya
- 3) `setSelected(Boolean)` = true bila dipilih dan false sebaliknya
- 4) `setMnemonic(KeyEvent.VK_O)` = shortcut, O = Huruf
- 5) `getText()` = mendapatkan String
- 6) `setFont(Font)` = mengubah model huruf
- 7) `setForeground(Color)` = mengubah warna font
- 8) `setText(String)` = mengubah string pada JCheckBox

e. JRadioButton

Versi lamanya bernama RadioButton yang terdapat didalam package AWT. JRadioButton digunakan untuk memilih salah satu dari sekian banyak pilihan yang disediakan untuk user. Berbeda dengan class JCheckBox yang dapat memilih minimal satu dan dapat memilih semua pilihan yang tersedia. Sebagai contoh memilih jenis kelamin, harus memilih satu. Sehingga diperlukan class lain untuk pengelompokkan. Class tersebut adalah class ButtonGroup.

Bentuk deklarasi class JRadioButton adalah sebagai berikut :

```
JRadioButton Nama_Objek = new JRadioButton (String,Boolean);
```

Contoh :

```
JRadioButton rdPria = new JRadioButton ("Pria",true);
```

Beberapa method yang dimiliki oleh class JRadioButton adalah sebagai berikut :

- 1) `setEnabled(boolean)` = bisa diakses atau tidak.
- 2) `isSelected()` = true bila dipilih dan false sebaliknya
- 3) `setSelected(Booleam)` = true bila dipilih dan false sebaliknya
- 4) `setMnemonic(KeyEvent.VK_O)` = shortcut, O = Huruf
- 5) `getText()` = mendapatkan String
- 6) `setFont(Font)` = mengubah model huruf
- 7) `setForeground(Color)` = mengubah warna font
- 8) `setText(String)` = mengubah string pada JRadioButton

f. JComboBox

JComboBox digunakan memilih salah satu dari sekian banyak pilihan yang disediakan untuk user. Prinsipnya sama dengan class JRadioButton. JComboBox ini sering juga disebut dengan Drop-down List. Untuk memasukkan data kedalam objek combo box menggunakan array 1 dimensi yang diurutkan berdasarkan index.

Bentuk deklarasi class JComboBox adalah sebagai berikut :

```
JComboBox Nama_Objek = new JComboBox (Array_of_list);
```

Contoh :

Dijelaskan lebih lanjut ke program. ☺

Beberapa method yang dimiliki oleh class JComboBox adalah sebagai berikut :

- 1) `setFont(Font)` = mengubah model huruf.
- 2) `setForeground(Color)` = mengubah warna font.
- 3) `setEnabled(boolean)` = dapat diakses(true) atau tidak (false)
- 4) `setSelectedIndex(index)` = menentukan pilihan indeks awal (dari 0)
- 5) `setMaximumRowCount(int)` = Tampilan JComboBox begitu di-click.
- 6) `addItemListener()` = untuk interaksi dengan user.
- 7) `getItemAt(indeks)` = mengambil String yang dipilih.
- 8) `getSelectedIndex()` = mengambil indeks yang dipilih.

3. PENERAPAN CLASS GUI PADA PEMROGRAMAN

a. Memahami Class Container

Objek dari Class GUI tidak dapat langsung dimasukkan ke dalam frame menggunakan Class JFrame. Class JFrame berfungsi untuk membuat bingkai dari sebuah form (framanya), hanya bingkainya saja tanpa alas. Apabila objek dari Class GUI ingin ditampilkan ke form maka diperlukan sebuah Class lain yang berfungsi untuk menampung objek dari Class GUI. Salah satu Class tersebut adalah Class Container.

Class Container terdapat pada package AWT. Class ini berfungsi untuk menampung objek dari class GUI pada sebuah form. Tanpa class ini objek tidak bisa ditampung pada sebuah form. Urutannya dapat disederhanakan bahwa objek GUI ditampung pada Class Container dan Class Container ditempel ke Class JFrame, sehingga Class Container tidak dapat berdiri sendiri dan ukuran Class Container mengikuti ukuran formnya (Class JFramanya).

b. Program menggunakan Class Container

Berikut adalah contoh penggunaan beberapa class GUI pada pemrograman. Pada contoh berikut digunakan Class Container dan Class JLabel. Class Container akan menampung Class JLabel.

Nama Program : cthcontainer.java

Sintaks Program :

```
import javax.swing.*;
import java.awt.*;

class clsframe extends JFrame
{
    private Container konten = new Container();
    private JLabel objLabel = new JLabel("Hai, Saya Label");

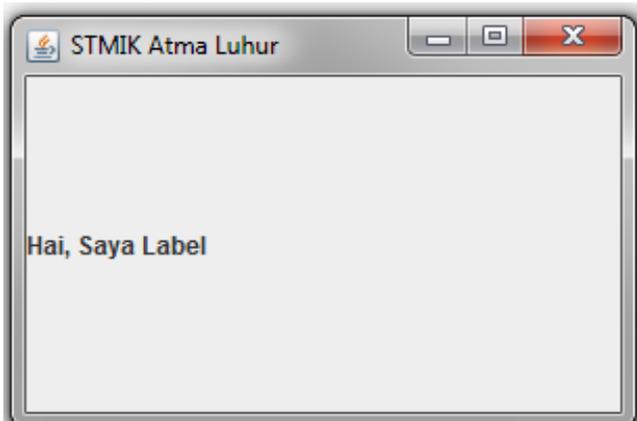
    clsframe()
    {
        super("STMIK Atma Luhur");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocation(100,200);
        setSize(300,200);
        konten = getContentPane();
        konten.add(objLabel);
        show();
    }
}
```

```

public class cthcontainer
{
    public static void main (String args[])
    {
        new clsframe();
    }
}

```

Output Program :



Penjelasan Program :

- Class clsframe merupakan turunan dari Class JFrame.
- import java.awt.* adalah perintah untuk memanggil seluruh class yang terdapat dalam package awt. Perintah ini digunakan karena menggunakan class container ke dalam program.
- import javax.swing.* adalah perintah untuk memanggil seluruh class yang terdapat dalam package swing. Perintah ini digunakan karena menggunakan Class JFrame dan Class JLabel ke dalam program.
- private Container konten = new Container() adalah perintah membuat objek dari class Container. Objek yang dibuat bernama konten.
- Private menandakan objek tersebut hanya dapat digunakan didalam class itu sendiri. private JLabel objlabel = new JLabel("Hai, Saya Label") adalah proses mendeklarasikan sebuah objek yang bernama objlabel dari class JLabel, sekaligus menginstansiasikan objek tersebut berisikan "Hai, Saya Label".
- konten = getContentPane() adalah perintah yang berfungsi untuk menghubungkan class container dengan class JFrame, melalui objek konten.
- konten.add(objlabel) adalah perintah yang berfungsi menempelkan objek objlabel ke dalam class container melalui objek konten.
- Pada class main(), class clsframe dipanggil menggunakan prinsip anonymous object.

• CLASS LAYOUT MANAGER

TUJUAN :

- Dapat memahami mengenai Class-Class Layout Manager
- Dapat menerapkan Class-Class Layout Manager pada pemrograman.

1. MEMAHAMI CLASS LAYOUT MANAGER

Pada modul sebelumnya, sudah diperkenalkan mengenai Class Container. Dengan Class Container ini, maka objek dari class GUI dapat ditampung dan ditampilkan.

Tetapi, permasalahan timbul apabila hanya menggunakan Class Container diantaranya:

- Class Container hanya bisa menampung satu objek saja, sehingga apabila ada objek yang lain yang ingin ditampung pada Class Container, maka objek yang lama akan digantikan dengan objek yang baru.
- Class Container tidak dapat mengatur posisi objek pada form. Objek akan ditempatkan ditengah sesuai dengan ukuran formnya.

Maka diperlukan cara yang lain sehingga Class Container dapat menampilkan lebih dari satu objek dan mengatur tata letak objeknya pada form.

Salah satu cara yang dapat dilakukan adalah dengan menggunakan Class lain yang dapat mengatur tata letak objek dan secara otomatis dapat menempatkan lebih dari satu objek pada Class Container. Class seperti ini disebut dengan Class Layout Manager.

Pada Java, sudah disediakan beberapa Class Layout Manager diantaranya adalah :

- Class FlowLayout
- Class GridLayout
- Class BorderLayout
- Class BoxLayout
- Class CardLayout
- Class GroupLayout
- Class SpringLayout

Untuk menggunakan class ini, Class Container memerlukan method yang berfungsi untuk menggunakan class tersebut. Method tersebut adalah method **setLayout()**.

Pada point berikutnya, akan ditunjukkan penerapan Class Layout Manager pada pemrograman sekaligus ditunjukkan perbedaan dan karakteristik dari tiap-tiap class. Class yang dicontohkan ada tiga yaitu Class Flowlayout, Class GridLayout dan Class BorderLayout.

2. PENERAPAN CLASS FLOWLAYOUT PADA PEMROGRAMAN

Class FlowLayout merupakan layout manager yang sangat sederhana. Menggunakan class ini, maka objek pada container akan ditempatkan sesuai dengan urutan penulisan objeknya dan mengikuti ukuran formnya. Sehingga urutan penambahan komponen menjadi penting.

Class FlowLayout mempunyai konstanta untuk mengatur tata letak objek pada Container. Konstanta tersebut adalah LEFT, CENTER dan RIGHT. Default konstanta dari Class ini adalah CENTER, sehingga objek akan ditempatkan ditengah-tengah Container mengikuti ukuran formnya.

Berikut contoh penerapan Class FlowLayout dengan beberapa Class GUI lainnya pada pemrograman.

Contoh Program : cthflowlayout.java

Sintaks Program :

```
import javax.swing.*;
import java.awt.*;

class clsframe extends JFrame
{
    private Container konten = new Container();
    private JLabel lblNIM = new JLabel("NIM :");
    private JLabel lblNama = new JLabel("Nama :");
    private JTextField txtNIM = new JTextField(10);
    private JTextField txtNama = new JTextField(15);
    private JButton btnTampil = new JButton("Tampil");

    clsframe()
    {
        super("Contoh Flowlayout");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocation(100,200);
        setResizable(false);
        setSize(200,200);

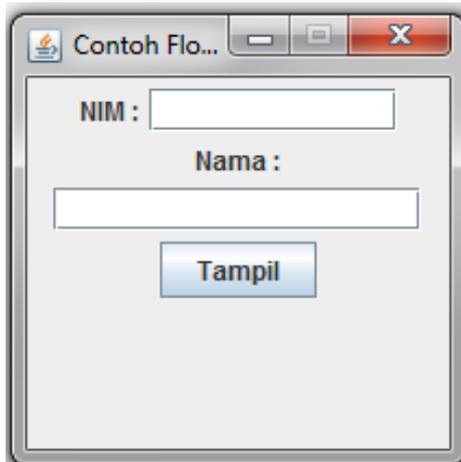
        konten = getContentPane();
        konten.setLayout(new FlowLayout(FlowLayout.CENTER));

        konten.add(lblNIM);
        konten.add(txtNIM);
        konten.add(lblNama);
        konten.add(txtNama);
        konten.add(btnTampil);

        show();
    }
}

public class cthflowlayout
{
    public static void main (String args[])
    {
        new clsframe();
    }
}
```

Output Program :



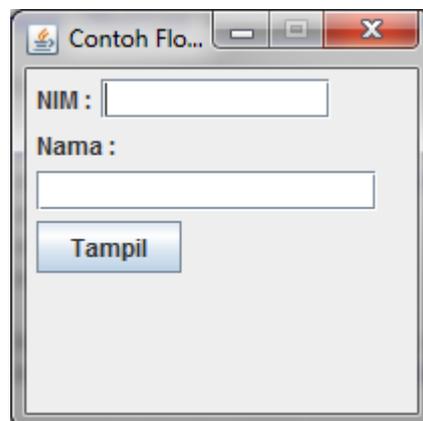
Penjelasan Program :

- `setResizable()` adalah sebuah method yang berfungsi untuk menentukan apakah ukuran frame bisa diubah atau tidak. Ada dua parameter yang disediakan yaitu `true` dan `false`. Gunakan parameter `false` apabila menginginkan ukuran form tidak dapat diubah-ubah.
- `konten.setLayout(new FlowLayout(FlowLayout.CENTER))`, objek dari class `Container` (objek `Konten`) menggunakan method `setLayout` untuk menggunakan Class `FlowLayout`. Konstanta yang digunakan adalah `CENTER`, sehingga objek akan ditempatkan ditengah-tengah container mengikuti ukuran formnya.
- Penggunaan Class `FlowLayout` menggunakan konsep `Anonymous Object`, sehingga pada contoh program diatas, objek dari class `FlowLayout` tidak memiliki nama.

Apabila pada sintaks program :

`konten.setLayout(new FlowLayout(FlowLayout.CENTER))`, diubah dengan :

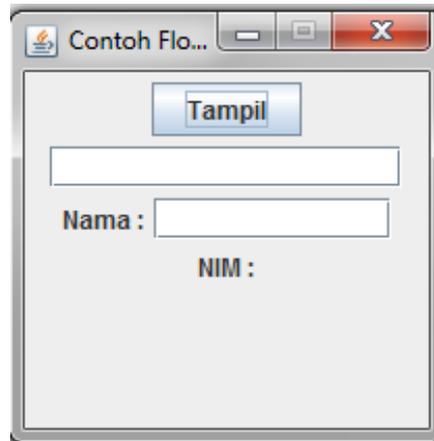
`konten.setLayout(new FlowLayout(FlowLayout.LEFT))`, maka output menjadi :



Apabila program diubah pada sintaks penambahan objek ke container menjadi sebagai berikut :

```
konten.add(btnTampil);  
konten.add(txtNama);  
konten.add(lblNama);  
konten.add(txtNIM);  
konten.add(lblNIM);
```

Sehingga outputnya menjadi sebagai berikut :



Kesimpulannya, dengan metode flowlayout urutan dalam proses menambahkan objek ke class container menentukan tampilan (layout) frame.

3. PENERAPAN CLASS GRIDLAYOUT PADA PEMROGRAMAN

Karakteristik dari Class GridLayout akan membagi container menjadi sejumlah tempat dalam bentuk matrik dengan ukuran yang sama mengikuti ukuran form. Menggunakan Class GridLayout ini, area pada container akan dibagi dalam format baris dan kolom. Akibatnya setiap objek akan memiliki ukuran yang sama, tidak peduli bagaimana ukuran sebenarnya. Setiap kali terjadi perubahan ukuran form, ukuran setiap objek juga akan berubah. Pada prinsipnya yang dipertahankan hanya jumlah baris dan jumlah kololmnya saja.

Untuk menggunakan Class GridLayout ini, diperlukan dua buah parameter. Parameter ini berfungsi untuk menentukan jumlah baris dan kolom.

Perhatikan contoh penggalan program berikut ini :

- `container.setLayout(new GridLayout(1,2))`, arti perintah tersebut adalah membagi area container menjadi 1 baris dan 2 kolom.
- `container.setLayout(new GridLayout(2,1))`, arti perintah tersebut adalah membagi area container menjadi 2 baris dan 1 kolom.
- `container.setLayout(new GridLayout(2,2))`, arti perintah tersebut adalah membagi area container dengan 2 baris dan 2 kolom

Berikut adalah contoh penerapan Class GridLayout dalam pemrograman. Container akan dibagi kedalam 2 baris dengan 3 kolom.

Nama Program : cthgridlayout.java

Sintaks Program :

```
import javax.swing.*;
import java.awt.*;

class clsframe extends JFrame
{
    private Container konten = new Container();
    private JLabel lblNIM = new JLabel("NIM :");
    private JLabel lblNama = new JLabel("Nama :");
    private JTextField txtNIM = new JTextField(10);
    private JTextField txtNama = new JTextField(15);
    private JButton btnTampil = new JButton("Tampil");

    clsframe ()
    {
        super("Contoh Flowlayout");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocation(100,200);
        setResizable(false);
        setSize(200,100);

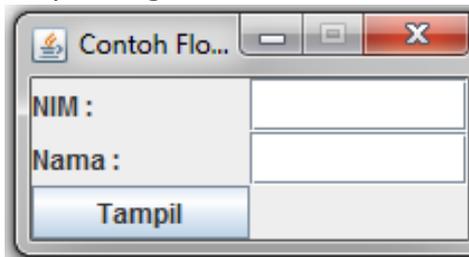
        konten = getContentPane();
        konten.setLayout(new GridLayout(3,2));

        konten.add(lblNIM);
        konten.add(txtNIM);
        konten.add(lblNama);
        konten.add(txtNama);
        konten.add(btnTampil);

        show();
    }
}

public class cthgridlayout
{
    public static void main (String args[])
    {
        new clsframe();
    }
}
```

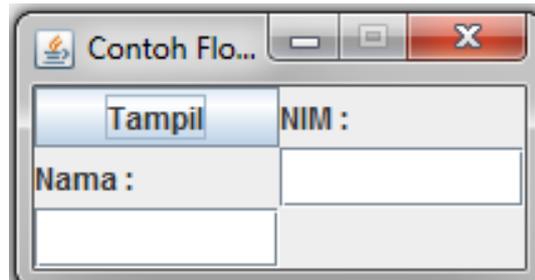
Output Program :



Apabila sintaks program pada pada proses penambahan objek diubah menjadi sebagai berikut :

```
konten.add(btnTampil);  
konten.add(lblNIM);  
konten.add(lblNama);  
konten.add(txtNIM);  
konten.add(txtNama);
```

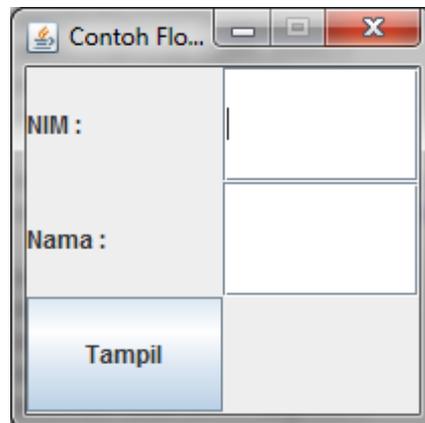
Sehingga outputnya sebagai berikut :



Kesimpulannya adalah urutan penambahan object pada class container menentukan posisi objek pada container.

Apabila pada sintaks program :

`setSize(200,100)`, diubah menjadi `setSize(200,200)`, maka output menjadi :



Kesimpulannya Setiap kali terjadi perubahan ukuran frame, ukuran setiap komponen juga akan berubah, mengikuti ukuran frame. Dan pada prinsipnya yang dipertahankan hanya jumlah baris dan jumlah kolomnya saja

4. PENERAPAN CLASS BORDERLAYOUT PADA PEMROGRAMAN

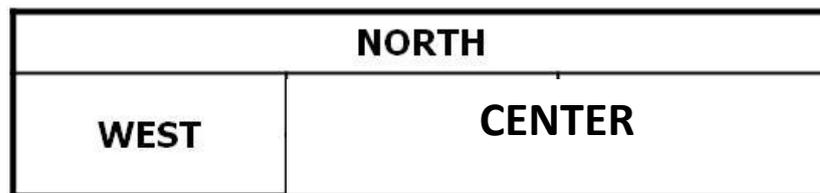
Class BorderLayout memiliki karakteristik yang mirip dengan Class GridLayout, yaitu membagi area pada Container. Karakteristik yang mirip tetapi tidaklah sama. Apabila pada Class GridLayout akan dibagi dalam bentuk matrik (baris dan kolom) dengan jumlah yang tidak ditentukan, maka pada Class BorderLayout membagi area dengan jumlah yang sudah pasti (konstan).

Jumlah area maksimal yang dapat dibagi pada container adalah sebanyak 5 area. 4 penamaan area mengikuti arah mata angin, sesuai dengan posisinya yaitu : NORTH (Utara), EAST (Timur), SOUTH (Selatan) dan WEST (Barat). Area yang terakhir terletak ditengah-tengah 4 area tersebut. Area tersebut dinamai dengan CENTER (tengah). Penamaan ini yang kemudian akan menjadi parameter untuk menggunakan Class BorderLayout ini.

Untuk lebih jelasnya mengenai pembagian area tersebut, dapat dilihat pada gambar berikut :



Jika salah satu dari area tidak digunakan maka, area yang berdekatan akan menggunakan area tersebut sebagai perluasan dari wilayahnya. Sebagai contoh jika pemasangan komponen di NORTH, WEST dan CENTER maka bentuk tampilan akan menjadi sebagai berikut :



Sehingga area WEST akan melebar mengambil area SOUTH sedangkan area CENTER akan melebar mengambil area SOUTH dan EAST.

Berikut adalah contoh program yang menggunakan Class BorderLayout untuk mengatur objek pada container. Pada contoh berikut, objek yang ditempatkan ke container merupakan objek dari class JButton.

Nama Program : cthborderlayout.java

Sintaks Program :

```
import java.awt.*;
import javax.swing.*;

class clsframe extends JFrame
{
    private Container konten = new Container();
    private JButton btnUtara = new JButton("UTARA");
    private JButton btnSelatan = new JButton("SELATAN");
    private JButton btnTimur = new JButton("TIMUR");
    private JButton btnBarat = new JButton("BARAT");
    private JButton btnTengah = new JButton("TENGAH");

    clsframe()
    {
        super("Contoh BorderLayout");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocation(100,200);
        setResizable(false);
        setSize(300,110);

        konten = getContentPane();
        konten.setLayout(new BorderLayout());
        konten.add(btnUtara, BorderLayout.NORTH);
        konten.add(btnTimur, BorderLayout.EAST);
        konten.add(btnTengah, BorderLayout.CENTER);
        konten.add(btnBarat, BorderLayout.WEST);
        konten.add(btnSelatan, BorderLayout.SOUTH);

        show();
    }
}

public class cthborderlayout
{
    public static void main (String args[])
    {
        new clsframe();
    }
}
```

Output Program :



Apabila perintah `konten.add(btnUtara, BorderLayout.NORTH)` dihilangkan, maka output program menjadi sebagai berikut :



Apabila perintah `konten.add(btnUtara, BorderLayout.EAST)` dihilangkan, maka output program menjadi sebagai berikut :



Apabila perintah `konten.add(btnUtara, BorderLayout.NORTH)` dan `konten.add(btnUtara, BorderLayout.EAST)` dihilangkan, maka output program menjadi sebagai berikut :



Kesimpulannya apabila salah satu bagian tidak ada, maka bagian yang lain akan memperluas wilayahnya, mengisi bagian yang tidak ada.

TUJUAN :

- Dapat memahami mengenai Class JPanel
- Dapat menerapkan Class JPanel pada pemrograman.

1. MEMAHAMI CLASS JPANEL

Pada modul sebelumnya, sudah diperkenalkan bagaimana menyusun beberapa objek pada Container. Container tersebut dapat menampung dan mengatur beberapa objek karena telah diatur menggunakan Class Layout Manager. Tetapi, dalam menyusun objek hanya mengandalkan container dengan class Layout Manager ternyata masih memiliki kekurangan.

Kekurangan tersebut diantaranya :

- Dengan satu buah container hanya bisa menggunakan satu Class Layout Manager saja dalam satu waktu. Hal ini mengakibatkan sulit untuk meletakkan berbagai objek dari Class GUI yang memiliki karakteristik yang berbeda satu sama lain.
- Dengan satu buah container dengan banyak objek dari class GUI didalamnya, sulit untuk mengatur tata letaknya sesuai dengan keinginan.

Adapun kekurangan ini dapat diatasi dengan menggunakan class yang lain, yang dapat mengatur banyak objek pada container. Class tersebut adalah class JPanel.

Pada dasarnya class JPanel mempunyai fungsi yang sama dengan Class Container, yaitu berfungsi untuk menampung objek class GUI. Sehingga class JPanel dapat juga diatur menggunakan Class Layout Manager. Class JPanel tidak dapat berdiri sendiri karena Class JPanel sebagai pelengkap dari container.

Perbedaan antara container tanpa class JPanel dan Container dengan Class JPanel adalah :

- Container tanpa Class JPanel
Setelah container diatur menggunakan Class Layout Manager, maka objek dari class GUI dapat diletakkan/ditampung kedalam Container.
- Container dengan Class JPanel
Class JPanel diatur tata letaknya menggunakan Class Layout Manager, maka objek dari class GUI dapat diletakkan/ditampung ke dalam Class JPanel. Container diatur menggunakan Class Layout Manager, maka Class JPanel yang berisi objek dari class GUI dapat diletakkan/ditampung kedalam Container.

Apabila diperlukan dan sesuai dengan kebutuhan, didalam Class JPanel dapat diletakkan lagi Class JPanel yang diatur dengan Class Layout Manager.

Kesimpulannya, Container tetap sebagai alas dari frame dengan pengaturan tata letak dan penampungan objek menggunakan Class JPanel.

2. PENERAPAN CLASS JPANEL PADA PEMROGRAMAN

Berikut adalah contoh penerapan Class JPanel pada pemrograman. Pada contoh tersebut, dibuat 4 objek dari Class JPanel. Masing-masing dari objek ini berisi objek dari class GUI, kemudian 4 objek dari Class JPanel ini ditampung ke alasnya, yaitu Container.

Contoh Program : cthjpanel.java

Sintaks Program :

```
import javax.swing.*;
import java.awt.*;

class clsframe extends JFrame
{
    private Container konten = new Container();
    private JPanel panel1 = new JPanel();
    private JPanel panel2 = new JPanel();
    private JPanel panel3 = new JPanel();
    private JPanel panel4 = new JPanel();
    private JLabel lblJudul = new JLabel("B I O D A T A");
    private JLabel lblNIM = new JLabel("NIM :");
    private JLabel lblNama = new JLabel("Nama :");
    private JTextField txtNIM = new JTextField(10);
    private JTextField txtNama = new JTextField(15);
    private JButton btnTampil = new JButton("Tampil");

    clsframe()
    {
        super("Contoh Dengan JPanel");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocation(100,200);
        setResizable(false);
        setSize(200,150);

        panel1.setLayout(new FlowLayout(FlowLayout.CENTER));
        panel1.add(lblJudul);
        panel2.setLayout(new GridLayout(2,1));
        panel2.add(lblNIM);
        panel2.add(lblNama);

        panel3.setLayout(new GridLayout(2,1));
        panel3.add(txtNIM);
        panel3.add(txtNama);

        panel4.setLayout(new FlowLayout(FlowLayout.CENTER));
        panel4.add(btnTampil);
    }
}
```

```

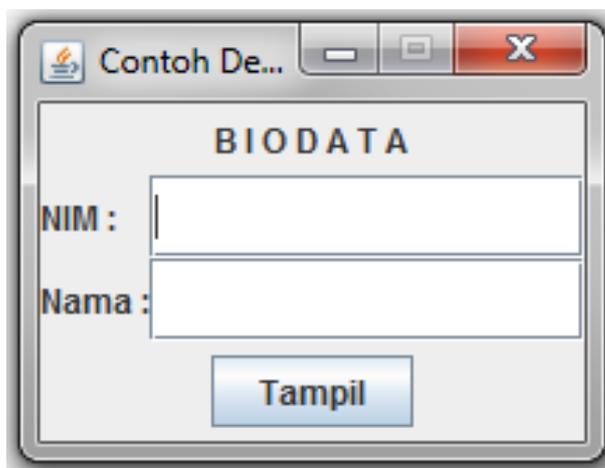
        konten = getContentPane();
        konten.setLayout(new BorderLayout());
        konten.add(panel1, BorderLayout.NORTH);
        konten.add(panel2, BorderLayout.WEST);
        konten.add(panel3, BorderLayout.CENTER);
        konten.add(panel4, BorderLayout.SOUTH);

        show();
    }
}

public class cthjpanel
{
    public static void main (String args[])
    {
        new clsframe();
    }
}

```

Output Program :



• INTERFACE LISTENER

TUJUAN :

- Dapat memahami mengenai Interface Listener
- Dapat memahami mengenai macam-macam Interface Listener
- Dapat menerapkan Interface Listener pada pemrograman.

1. MEMAHAMI INTERFACE LISTENER

Pada modul sebelumnya, sudah diperkenalkan program yang menggunakan beberapa class GUI dan class penampung objek dari class GUI sehingga menjadi satu buah form. Tetapi form yang dihasilkan tersebut belum dapat menerima respon yang diberikan oleh user pada saat form tersebut dijalankan. Artinya form belum dapat berinteraksi dengan user.

Contoh :

Seorang programmer diminta untuk membuat sebuah form yang dapat menghitung jumlah angka dari dua angka yang telah diinput. Maka, programmer dapat membuat form dengan memanfaatkan dua buah Class GUI yaitu Class JTextField dan Class JButton. Untuk Class JTextField dibuatkan 3 objek, dua objek berfungsi untuk menampung angka yang diinput dan satu objek untuk menampilkan hasil penjumlahan dari angka yang diinput tersebut. Sedangkan untuk Class JButton dibuatkan satu buah objek yang berfungsi sebagai triger/pemicu untuk mendapatkan hasil perhitungan.

Algoritma dari contoh tersebut adalah :

- Input angka pada dua objek dari Class JTextField yang telah disediakan
- Klik objek dari Class JButton yang telah disediakan
- Tampilkan hasil penjumlahan pada objek dari Class JTextField yang telah disediakan

Apabila program disusun menggunakan cara yang diperkenalkan pada modul sebelumnya, maka hasilnya adalah program tidak dapat menampilkan hasil penjumlahan dari angka yang telah diinput sebelumnya, walaupun objek dari Class JButton diklik berulang kali.

Hasil ini terjadi karena objek tidak dapat menerima/mengerti apa yang diinginkan oleh user. Pada Java, kejadian ini diartikan objek tidak dapat **mendengarkan** apa yang diinginkan oleh user. Sehingga perlu cara agar objek bisa menjadi pendengar (**LISTENER**) dan dapat mendengar apa yang diinginkan oleh user.

Pada contoh diatas, keinginan dari user adalah apabila mengklik sebuah tombol (objek dari Class JButton), maka hasil penjumlahan akan ditampilkan pada text field (objek dari Class JTextField).

Pada pemrograman, proses mengklik sebuah tombol merupakan salah satu contoh dari kejadian yaitu kejadian mengklik. Istilah ini disebut dengan **EVENT**.

Kesimpulannya, agar sebuah objek dapat mengerti apa yang diinginkan oleh user, maka user tersebut harus dapat mendengar (**LISTENER**) sehingga dapat melaksanakan kejadian (**EVENT**) yang diberikan oleh user.

Pada Bahasa Java, sudah menyiapkan beberapa Interface sehingga objek dapat melaksanakan event dari user. Interface yang disediakan sesuai dengan event yang diberikan. Beberapa Interface tersebut diantaranya :

- a. Interface ActionListener
Befungsi untuk menerima event yang ditimbulkan melalui mouse dan keyboard
- b. Interface MouseListener
Befungsi untuk menerima event yang ditimbulkan melalui mouse
- c. Interface MouseMotionListener
Befungsi untuk menerima event yang ditimbulkan melalui mouse
- d. Interface KeyListener
Befungsi untuk menerima event yang ditimbulkan melalui keyboard

Beberapa Event yang ditimbulkan oleh keyboard diantaranya :

- Menekan tombol enter pada sebuah objek
- Menekan tombol tertentu (selain enter) pada sebuah objek
- Melepaskan tombol tertentu (selain enter) pada sebuah objek
- Cursor meninggalkan sebuah objek pada saat menekan tombol tabulasi (tab).
- dan beberapa event lainnya.

Beberapa Event yang ditimbulkan oleh mouse diantaranya :

- Mengklik objek
- Mendouble klik objek
- Menggeser pointer mouse
- Cursor meninggalkan sebuah objek pada saat mouse mengklik objek yang lain
- dan beberapa event lainnya.

Pada modul sebelumnya sudah diperkenalkan bagaimana cara menggunakan Interface dalam pemrograman, yaitu dengan mengimplementasikan interface pada class yang menggunakan. Cirinya adalah dengan menggunakan keyword **implements**.

Selain implements, untuk menggunakan interface, sebuah class juga menggunakan method-method yang sudah disiapkan oleh interface tersebut. Keempat Interface tersebut memiliki method-method tersendiri dan tidak sama satu dengan yang lain. Berikut dibawah ini keempat Interface yang akan dibahas satu persatu.

2. PENERAPAN INTERFACE ACTIONLISTENER PADA PEMROGRAMAN

Interface ini terdapat dapat package AWT. Sehingga perlu mengimport package **java.awt.event** untuk menggunakan Interface ini. Interface ini berfungsi untuk menerima event yang ditimbulkan oleh mouse dan keyboard.

Interface ActionListener menyediakan sebuah method agar sebuah objek dapat menerima (mendengar) event dari user. Method tersebut bernama method **addActionListener()**. Method ini membutuhkan sebuah parameter. Parameternya adalah sebuah class lain yang akan mengerjakan event yang sudah diterima oleh Interface ActionListener. Class tersebut bernama **ActionPerformed**. Class tersebut mempunyai method yang bernama method **actionPerformed()**. Method ini yang akan mengerjakan event yang telah diterima oleh Interface ActionListener.

Dengan pengertian yang lain, pada Interface ActionListener terdapat sebuah Class ActionPerformed dan didalam Class tersebut terdapat method actionPerformed(). Jadi apabila terdapat sebuah class yang mengimplementasi Interface ActionListener, maka secara otomatis class tersebut juga mengimplementasi Class ActionPerformed, sehingga class tersebut dapat menggunakan method dari Class ActionPerformed untuk melaksanakan event yaitu method actionPerformed().

Apabila method actionPerformed() ditempatkan di Class yang sama dengan objek yang menggunakan method addActionListener(), maka parameter pada method addActionListener() cukup menggunakan perintah **this**. Dengan perintah **this**, menandakan class tersebut adalah class yang akan mengerjakan eventnya.

Berikut contoh penerapan Interface ActionListener. Pada contoh berikut, event tersebut pada saat sebuah tombol diklik dan menampilkan sebuah pesan. Parameter pada method addActionListener() menggunakan **this** karena method actionPerformed() dari Class ActionPerformed ditempatkan di class yang sama, tempat objek yang menerima event diciptakan.

Nama Program : cthactionlistener.java

Sintaks Program :

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class clsframe extends JFrame implements ActionListener
{
    private Container konten = new Container();
    private JPanel panel1 = new JPanel();
    private JPanel panel2 = new JPanel();
    private JPanel panel3 = new JPanel();
    private JPanel panel4 = new JPanel();
    private JLabel lblJudul = new JLabel("B I O D A T A");
    private JLabel lblNIM = new JLabel("NIM :");
    private JLabel lblNama = new JLabel("Nama :");
    private JTextField txtNIM = new JTextField(10);
    private JTextField txtNama = new JTextField(15);
    private JButton btnTampil = new JButton("Tampil");
```

```

clsframe ()
{
    super("ActionListener");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocation(100,200);
    setResizable(false);
    setSize(200,150);

    panel1.setLayout(new FlowLayout(FlowLayout.CENTER));
    panel1.add(lblJudul);

    panel2.setLayout(new GridLayout(2,1));
    panel2.add(lblNIM);
    panel2.add(lblNama);

    panel3.setLayout(new GridLayout(2,1));
    panel3.add(txtNIM);
    panel3.add(txtNama);

    panel4.setLayout(new FlowLayout(FlowLayout.CENTER));
    panel4.add(btnTampil);

    konten = getContentPane();
    konten.setLayout(new BorderLayout());
    konten.add(panel1,BorderLayout.NORTH);
    konten.add(panel2,BorderLayout.WEST);
    konten.add(panel3,BorderLayout.CENTER);
    konten.add(panel4,BorderLayout.SOUTH);

    btnTampil.addActionListener(this);

    show();
}

public void actionPerformed (ActionEvent e)
{
    if (e.getSource()==btnTampil)
    {
        String tampil ="";
        tampil+="B I O D A T A"+ "\n\n";
        tampil+="N I M      : "+txtNIM.getText()+"\n";
        tampil+="N a m a : "+txtNama.getText()+"\n";

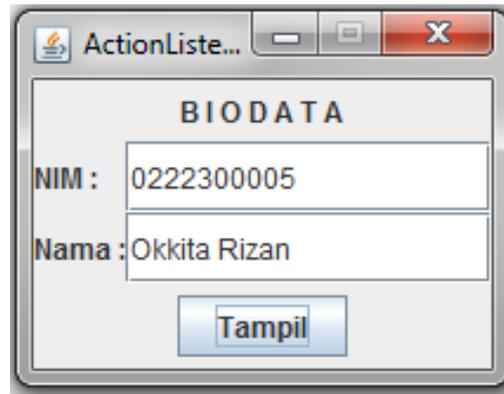
        JOptionPane.showMessageDialog(null,tampil,"Biodata",
        JOptionPane.INFORMATION_MESSAGE);

    }
}

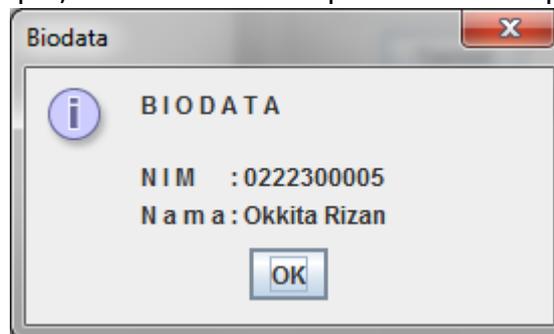
public class cthactionlistener
{
    public static void main (String args[])
    {
        new clsframe();
    }
}

```

Output Program :



txtNIM & txtNama diinput, ketika Tombol Tampil diklik muncul pesan sebagai berikut :



Penjelasan Program :

- `import java.awt.event.*`, adalah perintah untuk memanggil seluruh class event yang terdapat dalam package `awt`. Perintah ini digunakan karena menggunakan class `actionPerformed` ke dalam program.
- `class clsframe extends JFrame implements ActionListener` adalah perintah menamai sebuah class bernama `clsframe` dan merupakan turunan dari class `JFrame`. Class tersebut juga mengimplementasikan sebuah interface, dalam hal ini interface `ActionListener`.
- `btnTampil.addActionListener(this)` adalah perintah menghubungkan sebuah objek dengan interface, dalam hal ini interface `ActionListener`. Perintah `this` menandakan objek yang dihubungkan dengan interface, menempati class yang sama terhadap class yang diimplementasikan dengan interface. Dalam hal ini berada dalam satu class **clsframe**.
- `public void actionPerformed (ActionEvent e)` adalah sebuah metode dari class `actionPerformed`, parameter yang diberikan bernama `e`.
- `e.getSource()==btnTampil` adalah sebuah perintah memeriksa sebuah objek dihubungkan dengan sebuah interface. Dalam hal ini objek `btnTampil` yang diperiksa.
- `txtNIM.getText()` adalah perintah mengambil teks dari objek `txtNIM`.
- `JOptionPane.showMessageDialog()` adalah sebuah perintah untuk menampilkan sebuah pesan.

3. PENERAPAN INTERFACE MOUSELISTENER PADA PEMROGRAMAN

Interface ini juga terdapat pada package **java.awt.event**. Interface ini berfungsi untuk menangkap event yang ditimbulkan melalui mouse. Event yang ditimbulkan melalui keyboard tidak dapat diterima/ditangkap oleh Interface ini.

Interface `MouseListener` menyediakan sebuah method agar sebuah objek dapat menerima (mendengar) event dari user. Method tersebut bernama method **`addMouseListener()`**. Method ini membutuhkan sebuah parameter. Parameternya adalah sebuah class lain yang akan mengerjakan event yang sudah diterima oleh Interface `MouseListener`. Sama dengan Interface `ActionListener`, apabila sebuah class terdapat objek yang menggunakan method `addMouseListener()` dan pada class itu juga yang akan mengerjakan eventnya, maka parameter pada method `addMouseListener()` adalah perintah **`this`**.

Untuk class yang akan mengerjakan event menggunakan Interface ini, harus menyertakan beberapa method didalamnya walaupun tidak ada sintaks program yang dikerjakan. Method tersebut adalah :

- a. Method `mouseClick(mouseEvent)`
Method dikerjakan pada saat tombol mouse ditekan lalu dilepaskan.
- b. Method `mouseEvent(mouseEvent)`
Method dikerjakan pada saat kursor mouse memasuki area objek.
- c. Method `mouseExited(mouseEvent)`
Method dikerjakan pada saat kursor mouse meninggalkan area objek.
- d. Method `mousePressed(mouseEvent)`
Method dikerjakan pada saat tombol mouse ditekan.
- e. Method `mouseReleased(mouseEvent)`
Method dikerjakan pada saat tombol mouse dilepaskan.

Dari beberapa method tersebut terlihat bahwa method `mouseClick()` merupakan perpaduan antara method `mousePressed()` dengan method `mouseReleased()`.

Untuk Class yang akan mengerjakan event yang ditimbulkan oleh mouse, maka class tersebut harus mengimplementasi (implements) Interface `MouseListener`. Sama seperti Interface `ActionListener`.

Berikut contoh program yang menerapkan Interface `MouseListener`. Contoh program berikut akan ditunjukkan maksud dari 5 method diatas, sehingga dapat berinteraksi dengan user.

Nama Program : cthmouselistener.java

Sintaks Program :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class MyFrame extends JFrame implements MouseListener
{
    private Container container = new Container();
    private JLabel jLabelClicked = new JLabel("");
    private JLabel jLabelEntered = new JLabel("");
    private JLabel jLabelExited = new JLabel("");
    private JLabel jLabelPressed = new JLabel("");
    private JLabel jLabelReleased = new JLabel("");
    private JLabel jLabelCekMouse = new JLabel("Event Mouse Objek Ini");

    MyFrame ()
    {
        super("Test MouseListener");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300,150);
        setLocation(100,100);
        // mengasosiasikan seluruh objek dengan event handler

        jLabelCekMouse.addMouseListener(this);
        // menambah komponen ke frame
        container = getContentPane();
        container.add( new JLabel(""));
        container.add( new JLabel("Method Yang Aktif"));
        container.setLayout(new GridLayout(8,2));
        container.add( new JLabel("mouseClicked      : "));
        container.add( jLabelClicked);
        container.add( new JLabel("mouseEntered      : "));
        container.add( jLabelEntered);
        container.add( new JLabel("mouseExited      : "));
        container.add( jLabelExited);
        container.add( new JLabel("mousePressed     : "));
        container.add( jLabelPressed);
        container.add( new JLabel("mouseReleased    : "));
        container.add( jLabelReleased);
        container.add( new JLabel(""));
        container.add( jLabelCekMouse);
        container.add( new JLabel(""));
        container.add( new JLabel(""));
        show();
    }

    public void mouseClicked (MouseEvent e)
    {
        jLabelClicked.setText("AKTIF");
        jLabelEntered.setText("");
        jLabelExited.setText("");
        jLabelPressed.setText("");
        jLabelReleased.setText("");
    }
}
```

```

public void mouseEntered (MouseEvent e)
{
    jLabelClicked.setText("");
    jLabelEntered.setText("AKTIF");
    jLabelExited.setText("");
    jLabelPressed.setText("");
    jLabelReleased.setText("");
}

public void mouseExited (MouseEvent e)
{
    jLabelClicked.setText("");
    jLabelEntered.setText("");
    jLabelExited.setText("AKTIF");
    jLabelPressed.setText("");
    jLabelReleased.setText("");
}

public void mousePressed (MouseEvent e)
{
    jLabelClicked.setText("");
    jLabelEntered.setText("");
    jLabelExited.setText("");
    jLabelPressed.setText("AKTIF");
    jLabelReleased.setText("");
}

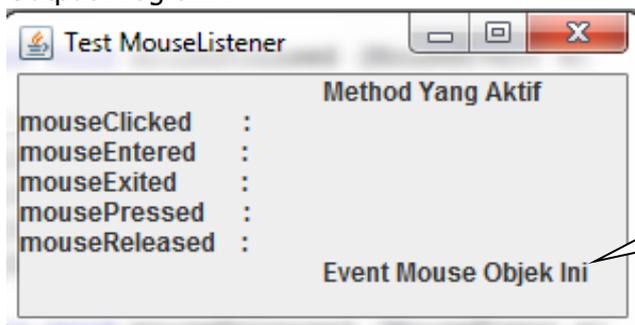
public void mouseReleased (MouseEvent e)
{
    jLabelClicked.setText("");
    jLabelEntered.setText("");
    jLabelExited.setText("");
    jLabelPressed.setText("");
    jLabelReleased.setText("AKTIF");
}

- }

public class cthmouselistener
{
    public static void main(String[] args)
    {
        new MyFrame ();
    }
- }

```

Output Program :



Tunjuk atau klik mouse pada objek ini, maka akan terlihat event mouseListener yang sedang dikerjakan

4. PENERAPAN INTERFACE MOUSEMOTIONLISTENER PADA PEMROGRAMAN

Interface yang terdapat pada package java.awt.event ini merupakan Interface yang digunakan sebagai pendukung dari Interface MouseListener.

Interface MouseMotionListener menyediakan dua buah method yaitu :

- a. Method mouseDragged (MouseEvent)
Method ini berfungsi untuk memantau pergerakan mouse pada saat melintasi sebuah objek pada saat mouse ditekan. Sama seperti pergerakan pada saat memindahkan objek di windows.
- b. Method mouseMoved (MouseEvent)
Method ini berfungsi untuk memantau pergerakan mouse pada saat melintasi area sebuah objek. Method ini memantau pergerakan pointer mouse yang melintasi area objek tanpa menekan tombol pada mouse. Ini yang membedakan method mouseDragged dengan method mouseMoved.

Sebagai pendukung dari Interface MouseListener, maka urutan pelaksanaan method dari kedua Interface ini adalah :

- Method mouseEntered(MouseEvent), dijalankan saat pointer mouse digerakkan memasuki area objek.
- Method mouseMoved(MouseEvent), dijalankan saat pointer mouse digerakkan melintasi objek dalam kondisi tidak ada tombol mouse yang ditekan.
- Method mousePressed(MouseEvent), dijalankan saat tombol mouse ditekan.
- Method mouseDragged(MouseEvent), dijalankan saat pointer mouse digeser melintasi objek sambil tombol mouse ditekan.
- Method mouseClicked(MouseEvent), dijalankan saat tombol mouse dilepas setelah ditekan dengan rentan waktu yang tidak terlalu lama.
- Method mouseReleased(MouseEvent), dijalankan saat tombol mouse dilepas.
- Method mouseExited(MouseEvent), dijalankan saat pointer mouse meninggalkan area objek.

Interface MouseMotionListener menyediakan sebuah method agar sebuah objek dapat menerima (mendengar) event dari user. Method tersebut bernama method **addMouseMotionListener()**. Method ini membutuhkan sebuah parameter. Parameternya adalah sebuah class lain yang akan mengerjakan event yang sudah diterima oleh Interface MouseMotionListener. Sama dengan Interface ActionListener maupun Interface MouseListener, apabila sebuah class terdapat objek yang menggunakan method addMouseMotionListener() dan pada class itu juga yang akan mengerjakan eventnya, maka parameter pada method addMouseMotionListener() adalah perintah **this**

Untuk Class yang akan mengerjakan event yang ditimbulkan oleh mouse, maka class tersebut harus mengimplementasi (implements) Interface MouseMotionListener. Sama seperti Interface ActionListener maupun Interface MouseMotionListener.

Berikut contoh program yang menerapkan Interface MouseMotionListener. Contoh program berikut akan ditunjukkan maksud dari 2 method diatas, sehingga dapat berinteraksi dengan user.

Nama Program : testMouseMotionListener.java

Sintaks Program :

```
| import javax.swing.*;
| import java.awt.*;
- import java.awt.event.*;

| class FrameKu extends JFrame implements MouseMotionListener
| {
|     private Container container = new Container();
|     private JLabel jLabelDragged = new JLabel("");
|     private JLabel jLabelMoved = new JLabel("");
|     private JLabel jCekMouse = new JLabel("Event Mouse Objek Ini");

|     FrameKu()
|     {
|         super("Test MouseMotionListener");
|         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
|         setSize(300,150);
|         setLocation(100,100);

|         // mengasosiasikan seluruh objek dengan event handler
|         jCekMouse.addMouseListener(this);

|         // menambah komponen ke frame
|         container = getContentPane();
|         container.setLayout(new GridLayout(5,2));
|         container.add( new JLabel());
|         container.add( new JLabel("Event Mouse yang Aktif"));
|         container.add( new JLabel("mouseDragged : "));
|         container.add( jLabelDragged);
|         container.add( new JLabel("mouseMoved : "));
|         container.add( jLabelMoved);
|         container.add( new JLabel());
|         container.add( jCekMouse);

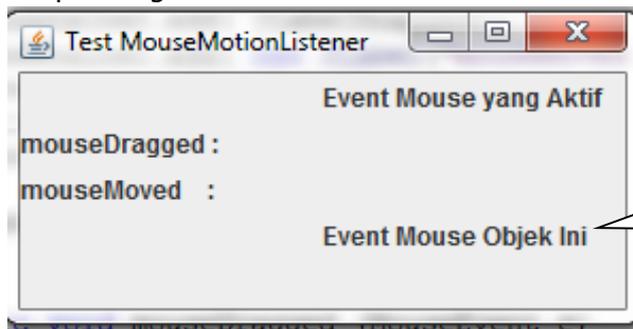
|         show();
|     }

|     public void mouseDragged (MouseEvent e)
|     {
|         jLabelDragged.setText("Aktif");
|         jLabelMoved.setText("");
|     }

|     public void mouseMoved (MouseEvent e)
|     {
|         jLabelDragged.setText("");
|         jLabelMoved.setText("Aktif");
|     }
- }

| public class testMouseMotionListener
| {
|     public static void main(String[] args)
|     {
|         new FrameKu();
|     }
- }
```

Output Program :



geser atau drag mouse pada objek ini, maka akan terlihat event MouseMotionListener yang sedang dikerjakan

5. PENERAPAN INTERFACE KEYLISTENER PADA PEMROGRAMAN

Interface KeyListener adalah Interface yang mengerjakan event yang ditimbulkan melalui keyboard. Interface yang terdapat pada package java.awt.event ini memiliki tiga buah method untuk mengerjakan event.

Ketiga method pada Interface KeyListener ini adalah :

- a. Method keyPressed (KeyEvent)
Method ini akan dikerjakan apabila sebuah tombol pada keyboard ditekan.
- b. Method keyReleased (KeyEvent)
Method ini akan dikerjakan apabila sebuah tombol pada keyboard dilepas.
- c. Method keyTyped (KeyEvent)
Method ini akan dikerjakan apabila sebuah tombol pada keyboard ditekan dan dilepas dalam waktu yang tidak lama.

Walaupun tidak semua method akan digunakan pada pemrograman, class yang mengimplementasikan method ini harus menyertakan ketiga methodnya dalam program.

Interface KeyListener menyediakan sebuah method agar sebuah objek dapat menerima (mendengar) event dari user. Method tersebut bernama method **addKeyListener()**. Method ini membutuhkan sebuah parameter. Parameternya adalah sebuah class lain yang akan mengerjakan event yang sudah diterima oleh Interface KeyListener. Sama dengan ketiga Interface sebelumnya, apabila sebuah class terdapat objek yang menggunakan method addKeyListener() dan pada class itu juga yang akan mengerjakan eventnya, maka parameter pada method addKeyListener() adalah perintah **this**.

Berikut contoh program yang menerapkan Interface KeyListener. Contoh program berikut akan ditunjukkan maksud dari 3 method diatas, sehingga dapat berinteraksi dengan user.

Nama Program : testKeyListener.java

Sintaks Program :

```
| import javax.swing.*;
| import java.awt.*;
| import java.awt.event.*;

| class BingkaiKu extends JFrame implements KeyListener
| {
|     private Container container = new Container();
|     private JTextField jtestKey = new JTextField("",8);
|     private JTextField hasilKey = new JTextField("",8);

|     BingkaiKu()
|     {
|         super("Test KeyListener");
|         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
|         setSize(200,180);
|         setLocation(100,100);

|         // mengasosiasikan seluruh objek dengan event handler
|         jtestKey.addKeyListener(this);

|         // menambah komponen ke frame
|         container = getContentPane();
|         container.setLayout(new GridLayout(5,1));
|         container.add( new JLabel("Coba Tekan Tombol Enter"));
|         container.add( new JLabel("Text Field Dibawah"));
|         container.add( jtestKey);
|         container.add( new JLabel("Hasil dari KeyListener"));
|         container.add( hasilKey);
|         show();
|     }

|     public void keyTyped (KeyEvent e)
|     {
|         if (e.getKeyCode() == e.VK_ENTER)
|             hasilKey.setText("ENTER ->> ditekan + dilepas");
|     }

|     public void keyPressed (KeyEvent e)
|     {
|         if (e.getKeyCode() == e.VK_ENTER)
|             hasilKey.setText("Tombol ENTER ->> ditekan");
|     }

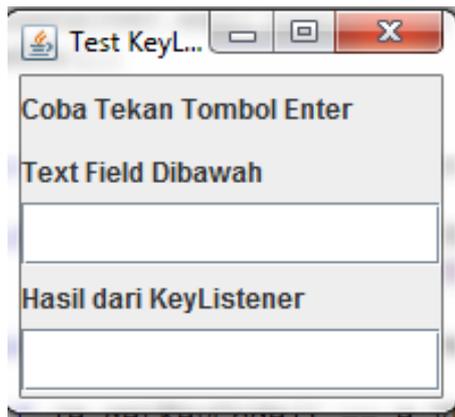
|     public void keyReleased (KeyEvent e)
|     {
|         if (e.getKeyCode() == e.VK_ENTER)
|             hasilKey.setText("Tombol ENTER ->> dilepas");
|     }
| }
```

```

public class testKeyListener
{
    public static void main(String[] args)
    {
        new BingkaiKu();
    }
}

```

Output Program :



Penjelasan Program :

- Class BingkaiKu mengimplementasi Interface KeyListener. Terlihat pada pemberian nama classnya.
- Event terjadi pada objek jtestKey dari Class JTextField karena method addKeyListener diberikan pada objek tersebut.
- Hasil dari event terlihat apabila tombol enter ditekan atau dilepas saat cursor berada pada objek jtestKey.
- Hasil dari eventnya akan ditunjukkan pada objek hasilKey dari Class JTextField.

DAFTAR PUSTAKA

An Introduction to Object Oriented Programming with Java, C. Thomas Wu, McGraw-Hill, New York, 2010.

An Object-Oriented Approach to Programming Logic and Design, Joyce Farrel, USA, 2013

Diktat Mata Kuliah Pemrograman Berorientasi Objek, M. Anif – Jati Lestari , UBL, 2009

Modul Praktikum Pemrograman Berorientasi Objek, STIKOMP Surabaya

Modul Praktikum Object Oriented Programming, Institut Teknologi Malang

Pemrograman Berbasis Objek dengan Bahasa Java, Elex Media Komputindo, Indrajani – Martin, Jakarta, 2007.

<http://attentedelune.blogspot.com>, diakses Februari 2014

<http://docs.oracle.com>, diakses Februari 2014

<http://gutti211.blogspot.com> , diakses Februari 2014

<http://ilmuduniainformatika.blogspot.com>, diakses Maret 2014

<http://javaprogkomp.blogspot.com>, diakses Februari 2014

<http://rizki-java.blogspot.com>, diakses Februari 2014

<http://rpl.if.its.ac.id>, diakses Februari 2014

<http://sangwidy.wordpress.com>, diakses Februari 2014

<http://top-ilmu.blogspot.com>, diakses Februari 2014

<http://www.beingjavaguys.com>, diakses Februari 2014

<http://www.termasmedia.com>, diakses Februari 2014

<http://www.varia.web.id>, diakses Februari 2014

<http://zamzambadruzaman.blog.com>, diakses Februari 2014

<http://zegyjib.wordpress.com>, diakses Februari 2014