



ABSTRAKSI (ABSTRACTION)

Pertemuan ke-7

Sub-CPMK

- *Mahasiswa mampu menyelesaikan masalah dengan menggunakan salah satu pilar OOP yaitu Abstraksi dalam konsep Pemrograman Berorientasi Objek (PBO). (C4, A4).*

Materi

1. Abstraction
2. Kelas abstrak
3. Interface
4. Final dan Static
5. Inner kelas
6. Exception
7. Package



1. Abstraksi

- Abstraksi adalah cara melihat objek dalam bentuk yang lebih sederhana. Abstraksi ini dapat kita analogikan seperti kita melihat objek komputer.
- Bila kita melihat objek komputer bukan sebagai sebuah kumpulan atau rangkaian elektronik, dan sebagainya. Walau kenyataan memang demikian, namun kita melihat komputer sebagai entitas tunggal (*single entity*) yang merupakan suatu objek mempunyai sifat dan karakteristik sendiri.

Lanj...

- Kita tidak perlu tahu bagaimana kompleksnya sistem komputer, tidak perlu tahu bagaimana cara kerja sistem komputer, dan lain sebagainya.
- Kita tidak melihat bagian-bagian dalam sebuah komputer sebagai suatu bagian yang berdiri-sendiri, tetapi sebagai satu kesatuan.

Lanj...

- Dengan abstraksi kita melihat suatu sistem yang kompleks sebagai kumpulan subsistem-subsistem yang lebih sederhana.
- Seperti objek komputer merupakan suatu sistem yang terdiri dari subsistem-subsistem.
- Abstraksi adalah salah satu dari 4 (empat) pilar konsep pemrograman berorientasi objek.

1.1 Class

- Kelas (class) merupakan templet (cetak biru) untuk objek.

```
public class nama_kelas
{
    ... atribut
    ... method
}
```

1.1 Class (Lanj...)

- Semua objek di dalam bahasa pemrograman Java diturunkan dari kelas, kelas ini bisa disebut sebagai superclass atau nenek moyang dari semua kelas yang ada di dalam bahasa pemrograman Java.

Nama_Kelas
- atribut : int
+ method() : void

Kelas: clsDokter

```
package dokter;  
import java.util.Scanner;  
  
//membuat kelas clsDokter  
public class clsDokter  
{  
  
}
```


1.2 Field

- Field/atribut adalah variabel yang didefinisikan di dalam kelas, dan disebut juga sebagai member variabel. Pada umumnya field merupakan variabel-variabel private dalam blok kelas.
- Apabila menggunakan kelas yang memungkinkan untuk mememodifikasi nilai suatu field, sebaiknya menggunakan property yang berkaitan. Property yang nantinya yang akan menyediakan akses ke field.

1.2 Field (Lanj...)

Access Modifier	Keterangan
Public	Untuk mendefinisikan tipe yang bisa diakses oleh siapa saja, baik kelas itu sendiri maupun kelas lain di dalam aplikasi.
Friend	Untuk mendefinisikan tipe yang hanya bisa diakses dari current project, atau dari assembly dimana tipe tersebut dideklarasikan.
Protected	Mendefinisikan tipe yang hanya bisa diakses oleh member-member kelas itu sendiri atau member kelas turunan.

1.2 Field (Lanj...)

Access Modifier	Keterangan
Protected Friend	Untuk mendefinisikan tipe yang bisa diakses oleh member-member dalam suatu assembly atau kelas turunannya. access modifier ini merupakan gabungan dari Protected dan Friend.
Private	Mendefinisikan tipe yang hanya bisa diakses oleh member-member di mana tipe tersebut di deklarasikan.

- Mendeklarasikan field/variabel:
tipe_data nama_variabel;

1.2 Field (Lanj...)

- Contoh mendeklarasikan field / variabel.

Kelas: clsDokter

```
package dokter;
import java.util.Scanner;

//membuat kelas clsDokter
public class clsDokter
{
    //mendeklarasikan variable/field/atribut
    private String IdDokter;
    private String Nama;
    private int Gaji;
}
```

1.3 Property Method

- Property method adalah sebuah method khusus yang digunakan untuk mendapatkan atau mengubah nilai dari sebuah field dalam kelas. Penggunaan property biasanya ditunjukkan pada field yang bersifat private.
- Field yang bersifat public dapat diakses langsung dari luar kelas, sedangkan field yang bersifat private hanya dapat diakses dari luar kelas melalui property.

1.3 Property Method (Lanj...)

- Contoh membuat property method.

```
public void setIdDokter(String newValue)
{
    IdDokter = newValue;
}

public String getIdDokter()
{
    return IdDokter;
}
```

1.4 Method

- Method / metoda adalah prosedur dan function yang dimiliki sebuah kelas.
- Dengan kata lain method adalah prosedur dan fungsi di lingkungan kelas.
- Method menunjukkan apa saja yang dapat dilakukan oleh objek hasil instansiasi kelas tersebut, method juga biasanya memiliki kata kunci **public** atau **private**.

1.4 Method (Lanj...)

- Method memiliki daftar paramter formal atau tanpa prameter, nilai kembalian atau tanpa nilai kembalian, dan shared atau non non-shared.
- Dimana method shared diakses melalui kelas, sedangkan method non-shared atau biasa disebut instance method, akan diakases melalui instansiasi kelas.

1.4 Method (Lanj...)

- Mendeklarasikan method dengan prosedur.

```
static void Luas(int Alas, int Tinggi)
{
    LuasSegitiga = 0.5 * Alas * Tinggi;
}
```

- Mendeklarasikan method dengan fungsi.

```
public double Luas(int Alas, int Tinggi)
{
    return 0.5 * Alas * Tinggi;
}
```

1.5 Objek

Objek
- atribut : int
+ method() : void

- Objek merupakan representasi nyata atau perwujudan (instance) dari kelas.
- Objek juga memiliki atribut, dan method yaitu tindakan yang dilakukan oleh objek.
- Sedangkan event adalah pemberitahuan yang diterima objek atau dikirimkan ke objek atau aplikasi lain.
- Event akan memicu suatu aksi.

1.5 Objek (Lanj...)

- Sintaks mendeklarasikan objek dari suatu kelas adalah:
 - `nama_kelas nama_objek = new nama_kelas();`
- Contoh:
 - `clsMhs objMhs = new clsMhs();`

1.6 Event

- Event adalah kejadian yang terjadi terhadap sebuah objek.
- Contoh event dalam program yang sering kita temukan adalah mengklik tombol (button) event yang diberikan pada objek button adalah klik.
- Dalam objek textbox misalnya kita memberikan event keypress (menekan enter), dan yang lainnya.

1.6 Event (Lanj...)

- Contoh even menekan enter.

```
private void txtKodeKeyPressed(java.awt.event.KeyEvent evt)
{
    ...
}
```

1.7 Overloading

- Overloading memungkinkan suatu kelas memiliki beberapa method dengan nama sama tetapi memiliki implementasi atau argumen yang berbeda, sepanjang deklarasi dan parameternya berbeda, atau disebut signaturenya berbeda.
- Hal ini dimungkinkan asalkan deklarasi method membuat penanda berbeda di satu kelas.

1.7 Overloading (Lanj...)

- Method overloading seperti ilustrasi berikut.

<u>return type</u>	<u>nama method</u>	<u>daftar parameter</u>
void	Coba	(int t1)
void	Coba	(int t1, int t2)
void	Coba	(int t1, int t2, int t3)
void	Coba	(int t1, int t2, int t3, int t4)
↓	↓	↓
sama	sama	berbeda

1.7 Overloading (Lanj...)

Kelas: clsHitung

```
package operasimatematik;  
public class clsHitung  
{  
    public double hasil(int a, int b)  
    {  
        return a*b;  
    }  
    public double hasil(double a, int b, int c)  
    {  
        return a+b*c;  
    }  
    public double hasil( int a, double b, int c, int d)  
    {  
        return a+b+c+d;  
    }  
}
```







1.7 Overloading (Lanj...)

Program: OperasiMatematik

```
package operasimatematik;  
public class OperasiMatematik  
{  
    public static void main(String[] args)  
    {  
        // TODO code application logic here  
        clsHitung objHitung = new clsHitung();  
  
        System.out.println("Hasil a+b="+objHitung.hasil(10,  
5));  
        System.out.println("Hasil  
a+b*c="+objHitung.hasil(10, 5, 2));  
        System.out.println("Hasil  
a+b+c+d="+objHitung.hasil(10, 5, 2, 4));  
    }  
}
```

1.7 Overloading (Lanj...)

- Contoh keluaran program seperti berikut.

Report Problems Window	iReport output	Output - Ope
	run:	
	Hasil a+b= 50.0	
	Hasil a+b*c= 20.0	
	Hasil a+b+c+d= 21.0	
	BUILD SUCCESSFUL (total time: 0 seconds)	

1.8 Overriding

- Overriding adalah method subclass sama dengan method super class, parameteranya sama tetapi pernyataan atau implementasinya berbeda.
- Untuk mengimplementasikan overriding berkaitan dengan konsep pewarisan (Inheritance).

1.8 Overriding (Lanj..)

Kelas: clsNama_Asli

```
package lat_overriding_1;

public class clsNama_Asli
{
    public void Namanya()
    {
        System.out.println("Nama: Dewi");
    }
}
```

1.8 Overriding (Lanj..)

Kelas: clsNama_Panggilan

```
package lat_overriding_1;  
  
public class clsNama_Panggilan extends clsNama_Asli  
{  
    public void Namanya()  
    {  
        System.out.println("Nama: Wati");  
    }  
}
```

1.8 Overriding (Lanj..)

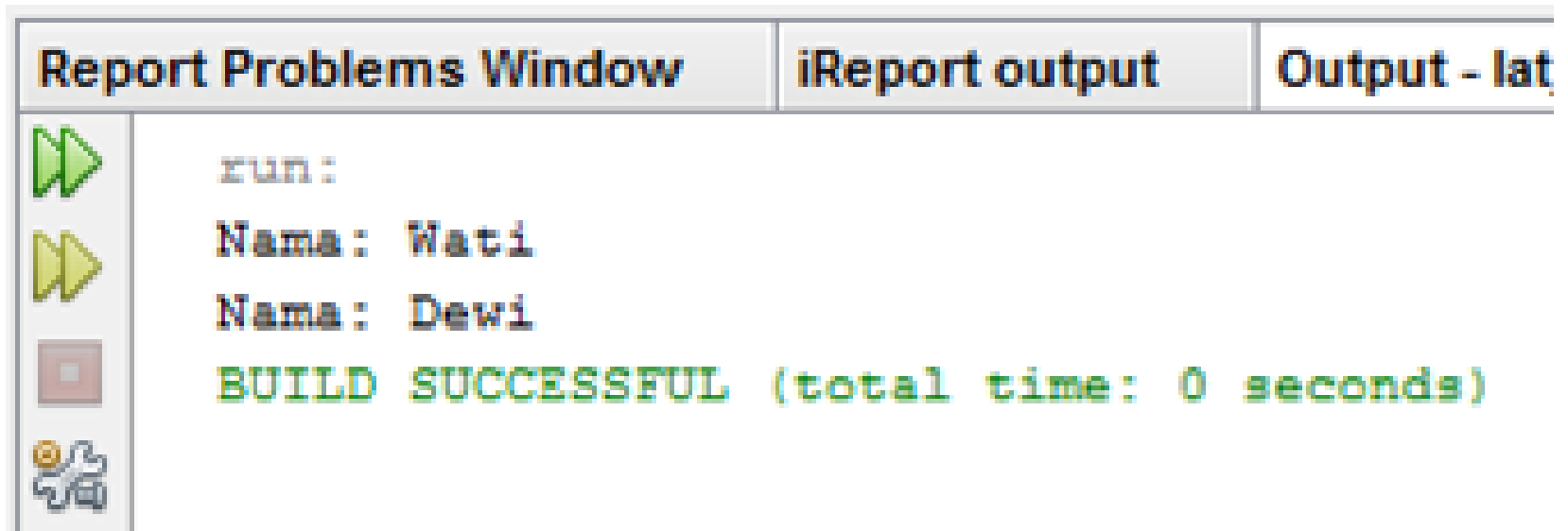
Main Program: Lat_overriding_1

```
package lat_overriding_1;
public class Lat_overriding_1 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        // TODO code application logic here
        clsNama_Panggilan obj_panggilan = new
clsNama_Panggilan();
        clsNama_Asli obj_namaasli = new clsNama_Asli();

        //memanggil method Namanya() pada kelas
clsNama_Panggilan
        obj_panggilan.Namanya();
        //memanggil method Namanya() pada kelas clsNama_Asli
        obj_namaasli.Namanya();
    }
}
```

1.8 Overriding (Lanj..)

- Contoh keluaran program seperti berikut.



The screenshot shows a window titled 'Report Problems Window' with three tabs: 'Report Problems Window', 'iReport output', and 'Output - lat'. The 'Output - lat' tab is active, displaying the following text:

```
run:  
Nama: Wati  
Nama: Dewi  
BUILD SUCCESSFUL (total time: 0 seconds)
```

1.8 Overriding (Lanj..)

- **Catatan**

Konsep overloading dan overriding akan banyak digunakan pada konsep inheritance (pewarisan), encapsulation (pembungkusan) dan polymorphism (banyak bentuk) berdasarkan studi kasus yang akan diselesaikan.

1.9 Konstruktor dan Destruktor

- Konstruktor (constructor) digunakan untuk menciptakan suatu objek, dan untuk menghancurkan suatu objek disebut destruktur (destructor).
- Pada bahasa pemrograman Java, menghancurkan objek dilakukan secara otomatis pada saat program selesai dijalankan.

1.8 Overriding (Lanj..)

Kelas: clsMahasiswa

```
public class clsMahasiswa
{
    private String NIM;
    private String Nama;

    public clsMahasiswa(String NIM, String Nama)
    {
        this.NIM = NIM;
        this.Nama = Nama;
    }
}
```



2. Kelas Abstrak

- Kelas abstrak adalah kelas yang tidak dapat dibuat instansnya.
- Kelas abstrak digunakan untuk menciptakan kelas yang hanya mendeklarasikan format generik tanpa mengimplementasikan secara detail fungsi-fungsi dari kelas abstrak.
- Akan tetapi semua hal yang berkaitan dengan fungsionalitas dari kelas tetap ada, seperti field, method, dan konstruktornya tetap dapat diakses.

Lanj...

- Sebuah kelas dikatakan kelas abstrak jika ada method hanya dideklarasikan saja (tidak diimplementasikan).
- Untuk membuat kelas abstrak menggunakan kata kunci "**abstract**", perintah abstract diletakkan sebelum nama Class dan nama Method.
- **Sintaks:** public **abstract** class kelasAbstrak

2.1 Meng-Extend Kelas Abtrak

- Kelas asbtrak juga dapat diturunkan (extend), caranya seperti contoh berikut.
 - `public class clsDosen extends kelasAbstrak`

2.2 Method Abtrak

- Bilamana menginginkan suatu kelas memiliki method tetapi implementasi methodnya dilakukan pada subkelas, maka dapat dilakukan dengan cara mendeklarasikan method tersebut sebagai asbtrak.
- Method abstrak tidak memiliki definisi, dan method tersebut diakhiri dengan tanda ";".
 - public **abstract** void Tunjangan();
 - public **abstract** void Bonus();

2.2 Method Abtrak (Lanj...)

- Mendeklarasikan sebuah method sebagai abstrak menyebabkan:
 1. Kelas tersebut juga harus didefinisikan sebagai abstrak.
 2. Semua subkelas harus meng-override method tersebut atau mendeklarasikan diri sebagai abstrak.

2.2 Method Abtrak (Lanj...)

Kelas Abstrak: kelasAbstrak

```
package latkelasabstrak;  
  
public abstract class kelasAbstrak  
{  
    public abstract void Tunjangan();  
    public abstract void Bonus();  
}
```

2.2 Method Abtrak (Lanj...)

Kelas: clsDosen

```
package latkelasabstrak;  
public class clsDosen extends kelasAbstrak  
{  
    @Override  
    public void Tunjangan()  
    {  
        System.out.println("Besar tunjangan: 1000000");  
    }  
  
    @Override  
    public void Bonus() {  
        System.out.println("Bonus: 10000000");  
    }  
}
```


2.2 Method Abtrak (Lanj...)

Main Program

```
package latkelasabstrak;  
  
public class LatKelasAbstrak {  
    public static void main(String[] args)  
    {  
        // TODO code application logic here  
        clsDosen objDosen = new clsDosen();  
  
        objDosen.Tunjangan();  
        objDosen.Bonus();  
    }  
}
```

2.2 Method Abtrak (Lanj...)

- Keluaran program seperti berikut.

Report Problems Window	iReport output	Output - LatK
	<code>run:</code>	
	<code>Besar tunjangan: 1000000</code>	
	<code>Bonus: 10000000</code>	
	<code>BUILD SUCCESSFUL (total time: 0 seconds)</code>	



3. Interface

- Interface merupakan kumpulan method-method abstrak.
- Sebuah kelas yang mengimplementasikan interface, mewarisi method-method abstrak dari interface tersebut.
- Persamaan interface dengan kelas:
 - Interface dapat memiliki banyak method.
 - Interface ditulis dalam file dengan ekstensi .java.

Lanj...

- Berikut ini perbedaan interface dengan kelas:
 - Kita tidak bisa membuat interface dari sebuah interface.
 - Interface tidak memiliki konstruktor.
 - Semua method dalam interface adalah asbtrak.
 - Interface tidak bisa memiliki field instance.
 - Interface tidak di-extand, tetapi di-implements oleh kelas.
 - Sebuah interface dapat meng-extend beberapa interface lainnya.

3.1 Mendeklar Interface

- Kata kunci interface digunakan untuk mendeklarasikan sebuah interface.

```
/* File: NamaInterface.java */  
Import java.lang.*;  
// Satu atau beberapa statemen import lainnya  
  
public interface NamaInterface  
{  
    //Satu atau beberapa field static final  
    //Satu atau beberapa deklarasi method abstrak  
    //catatan: modifier static tidak boleh digunakan  
    dalam interface  
}
```


3.1 Mendeklar Interface (Lanj..)

- Interface memiliki properti berikut:
 - Interface secara implisit merupakan abstrak. Kita tidak perlu menggunakan kata kunci abstract ketika mendeklarasikan sebuah interface.
 - Setiap method dalam interface juga secara implisit merupakan abstrak, sehingga kita tidak perlu menggunakan kata kunci abstract.
 - Method dalam interface secara implisit merupakan public.

3.1 Mendeklar Interface (Lanj..)

- Perhatikan contoh berikut.

```
Interface: intDosen  
package latihaninterface;  
  
public interface intDosen  
{  
    void Tunjangan();  
    void Bonus();  
}
```

3.2 Mengimplemen Interface

- Ketika sebuah kelas meng-implements interface, kelas tersebut harus mengimplementasikan method yang ada dalam interface.
- Perhatikan contoh berikut.

3.2 Mengimplemen Interface (Lanj..)

Kelas: clsDosen

```
package latihaninterface;  
public class clsDosen implements intDosen  
{  
    @Override  
    public void Tunjangan()  
    {  
        System.out.println("Tunjangan: 1000000");  
    }  
    @Override  
    public void Bonus()  
    {  
        System.out.println("Bonus: 9000000");  
    }  
}
```

3.2 Mengimplemen Interface (Lanj..)

Min program

```
package latihaninterface;

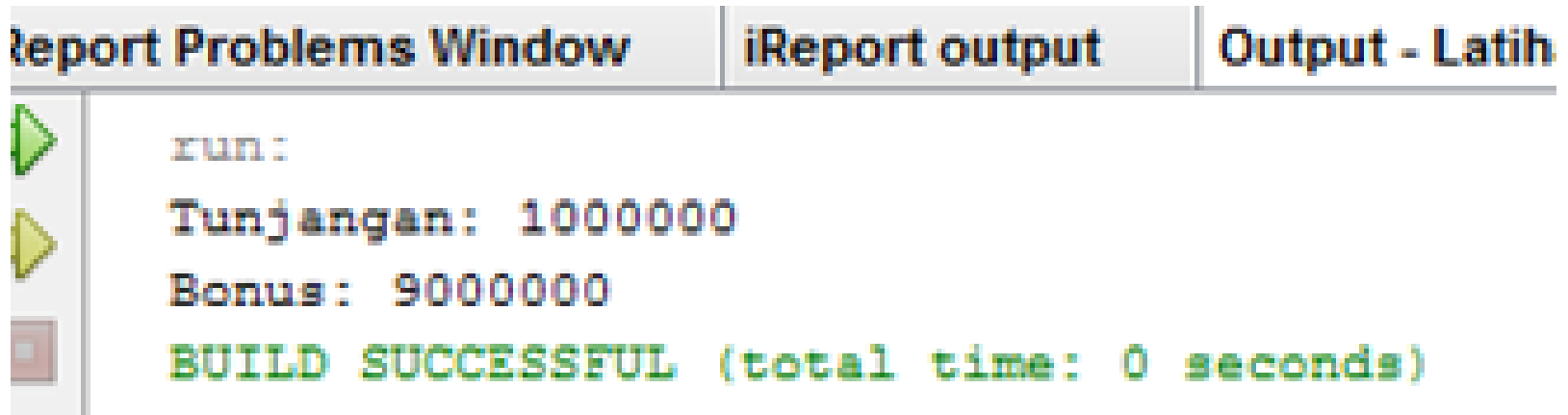
public class LatihanInterface {

    public static void main(String[] args)
    {
        // TODO code application logic here
        clsDosen objDosen = new clsDosen();

        objDosen.Tunjangan();
        objDosen.Bonus();
    }
}
```

3.2 Mengimplemen Interface (Lanj..)

- Contoh keluaran program seperti berikut.



The screenshot shows an IDE window with three tabs: 'Report Problems Window', 'iReport output', and 'Output - Latih'. The 'Output - Latih' tab is active, displaying the following text:

```
run:  
Tunjangan: 1000000  
Bonus: 9000000  
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.3 Meng-Extends Interface

- Sebuah interface dapat meng-extends interface lainnya.
- Perhatikan contoh berikut.

```
// File : Olahraga.java //  
public interface Olahraga  
{  
    public void setTim1(String nama);  
    public void setTim2(String nama);  
}
```

3.3 Meng-Extends Interface (Lanj..)

```
// File : Basket.java //
public interface Basket extends Olahraga
{
    public void skorTim1(int poin);
    public void skorTim2(int poin);
    public void akhirBabak(int babak);
}

// File : BolaVoli.java //
public interface BolaVoli extends Olahraga
{
    public void skorTim1(int poin);
    public void skorTim2(int poin);
    public void akhirBabak(int periode);
    public void akhirAkhir(int hasil);
}
```




4. Final dan Static

4.1 Final

- Penggunaan perintah final pada kelas mengakibatkan kelas tersebut tidak bisa diturunkan.
- Bila final digunakan pada method maka method tersebut tidak bisa dilakukan overriding.

4.2 Static

- Variabel atau method yang dideklarasikan dengan perintah static dapat dipanggil langsung, tidak perlu membuat instan dari kelas tersebut.



5. Inner Kelas

- Java membolehkan programmer menyisipkan suatu kelas ke dalam kelas lainnya. Kelas sisipan ini disebut kelas inner. Inner berguna untuk mendukung suatu proses yang akan dijalankan oleh kelas luarnya.
- Beberapa ketentuan kelas inner:
 - Kelas luar yang mengandung kelas inner, bila dikompilasi akan menghasilkan dua file *.class, yaitu Luar.class dan Luar\$Inner.class.

Lanj...

- Kelas Inner boleh tidak diberi nama, yang disebut Anonymous Inner.
- Kelas Inner dapat diberi modifier akses public, atau protected, atau default, atau private.
- Untuk mengakses referensi this dari kelas luar digunakan bentuk NamaKelasLuar.this.
- Kelas luar bertanggung jawab dalam instansiasi Inner (yang non static). Kalau objek kelas Luar adalah a, dan objek kelas Inner adalah b, maka sintaksnya adalah:

Lanj...

- `Luar a = new Luar();`
 - `Luar.Inner b = a.new Inner();`
- Jika kelas Inner bersifat static, maka objek milik kelas Inner dapat dibuat sendiri tanpa melalui kelas Luarnya, (Artinya kelas Inner tidak dapat mengakses attribute ataupun method non static milik kelas Luarnya).

Kelas Inner lazim digunakan untuk membuat handler di method `maind()` pada suatu aplikasi GUI. Handler merupakan bagian program yang memproses event-event yang dipicu ketika user berinteraksi dengan komponen-komponen GUI.



6. Axception

- Exception adalah singkatan dari Exceptional Events.
- Kesalahan (errors) yang terjadi saat runtime, menyebabkan gangguan pada alur eksekusi program. Beberapa tipe error yang dapat muncul, seperti error pembagian 0, mengakses elemen di luar jangkauan sebuah array, input yang tidak benar dan membuka file yang tidak ada.

Lanj...

- Jenis-jenis Exception, berdasarkan jenisnya kesalahan dalam pemrograman terbagi menjadi 3, yaitu:
 - Runtime Error
 - Logical Error
 - Syntax Error

Lanj...

- Pengertian Try, Catch, Finally, dan Throw
 - Try, digunakan untuk mencoba menjalankan block program kemudian mengenai dimana munculnya kesalahan yang ingin diproses. Keyword ini juga harus dipasangkan dengan keyword catch atau keyword finally.
 - Catch, catch harus dipasangkan dengan try. Kegunaan keyword ini adalah menangkap kesalahan atau bug yang terjadi dalam block try.

Lanj...

- Finally, yang menunjukkan bahwa block program tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada.
- Throw, digunakan untuk melemparkan suatu bug yang dibuat secara manual.



7. Package

- Untuk menggunakan java package / paket dilakukan dengan mendeklarasikan kata kunci package kemudian diikuti dengan nama paket.

7.1 Membuat Package

- Untuk membuat paket dilakukan dengan kata kunci package kemudian diikuti nama paket.

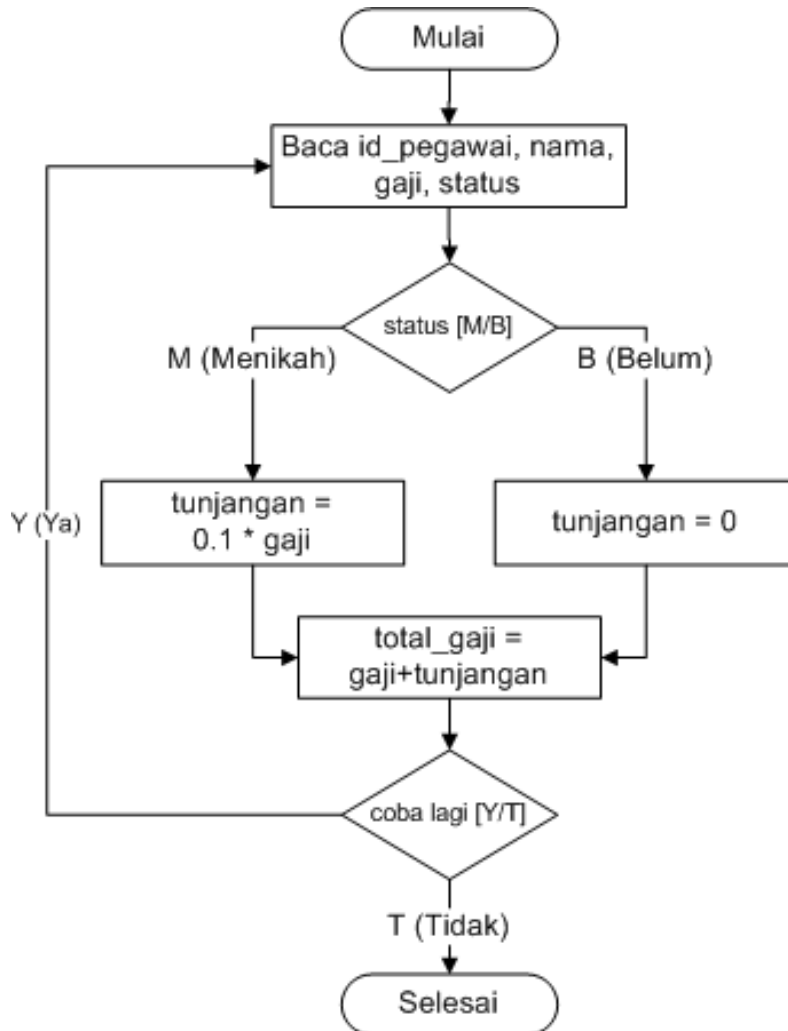
7.2 Memanggil Kelas Pada Package

- Untuk memanggil kelas yang berbeda paket dilakukan dengan import terhadap package tersebut.

Ringkasan:

- Salah satu dari empat pilar pemrograman berorientasi objek adalah abstraksi. Abstraksi adalah cara melihat objek dalam bentuk yang lebih sederhana. Dengan abstraksi kita melihat suatu sistem yang kompleks sebagai kumpulan subsistem-subsistem yang lebih sederhana. Seperti kita melihat laptop, kita tidak akan pernah berpikir tentang rangkian dalam laptop yang rumit.

Latihan Mandiri



- Analisis dari flowchart berikut, dan buat programnya dengan konsep berorientasi objek.
- Buat Progamnya:

Latihan Mandiri (Lanj...)

1. Buat program dengan konsep berorientasi objek.
2. Buat program dengan konsep berorientasi objek mengimplementasikan property method.
3. Buat program dengan konsep berorientasi objek dengan mengimplementasikan abstrak kelas.
4. Buat program dengan konsep berorientasi objek dengan mengimplementasikan interface.



TERIMA KASIH

U N I V E R S I T A S B U N D A M U L I A