

IF34348 - PEMROGRAMAN LANJUT

OBJECT ORIENTED PROGRAMMING (OOP)

06

Oleh : Andri Heryandi, M.T.

DEFINISI

IF34348 - Pemrograman Lanjut

- **Object Oriented Programming (OOP) / Pemrograman Berbasis Objek (PBO)** adalah konsep pemrograman yang menerapkan konsep objek, dimana objek terdiri dari atribut (informasi-informasi mengenai objek) dan method (prosedur/proses) yang bisa dilakukan oleh objek tersebut.
- **Software/Perangkat lunak** terdiri dari objek-objek yang saling berinteraksi.



Oleh : Andri Heryandi, M.T.

CONTOH

IF34348 - Pemrograman Lanjut

■ Contoh :

■ Object : Manusia

- **Attribut** : Nama, tinggi, umur, berat badan dll
- **Method** : Makan, Minum, Berjalan, Bekerja

■ Object : Windows

- **Attribut** : Left, Top, Width, Height, BackgroundColor,
- **Method** : OnClick, OnClose, OnDoubleClick

■ Object : Keluarga

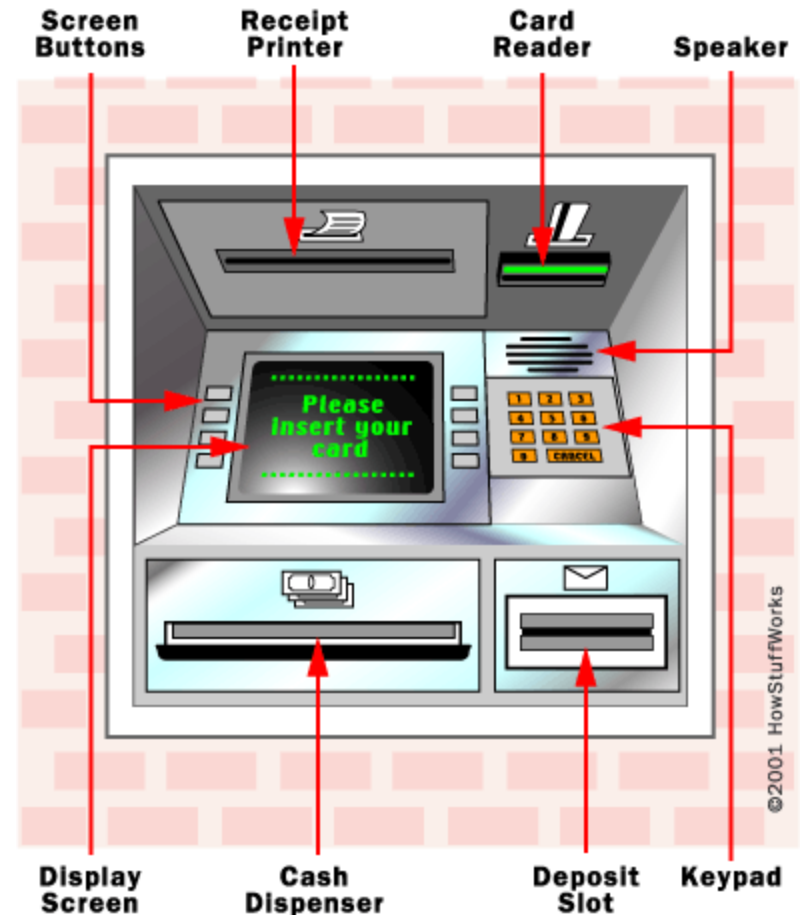
- **Attribut** : ayah, Ibu, Anak[]
- **Method** : TambahAnak,



CONTOH

IF34348 - Pemrograman Lanjut

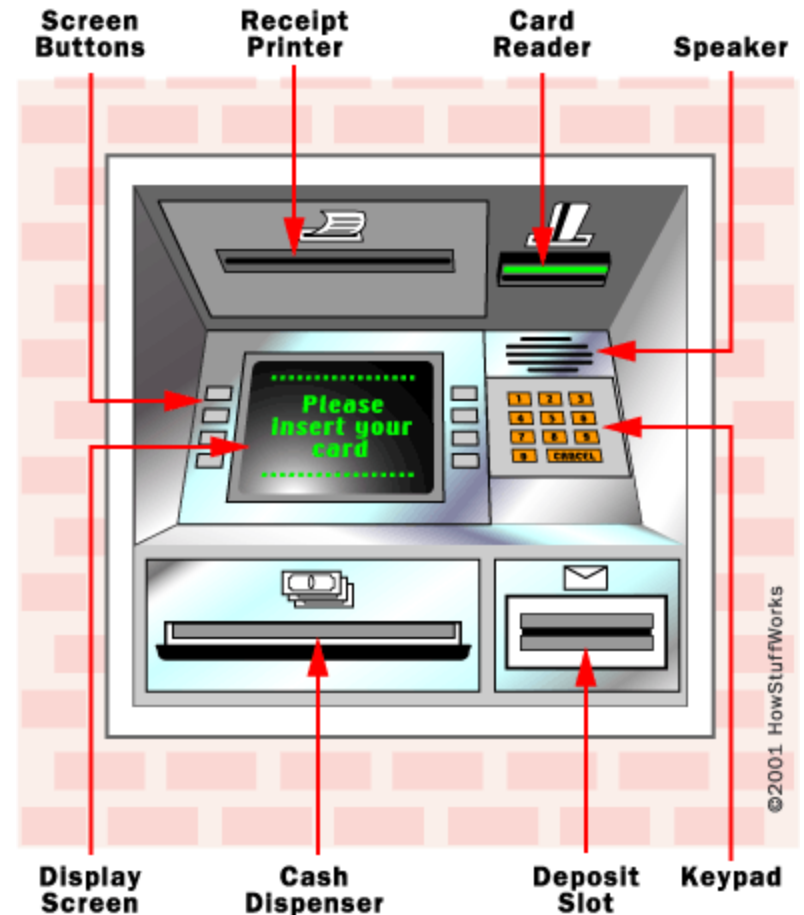
- Perhatikan mesin ATM
- Sebuah mesin ATM terdiri dari elemen-elemen berikut :
 - Display Screen (Layar)
 - Screen Button (Tombol)
 - Receipt Printer
 - Card Reader
 - Speaker
 - Cash Dispenser
 - Deposit Slot
 - Keypad
- Di dalam konsep OOP, setiap setiap elemen tersebut berinteraksi dengan mengirimkan pesan (message) tertentu.



CONTOH

IF34348 - Pemrograman Lanjut

- Interaksi ketika penekanan tombol Ambil Uang Rp. 200.000
 - Speaker mengeluarkan bunyi beep.
 - CashDispenser mengeluarkan uang 200.000
 - Jika uang diambil, maka Receipt Printer mencetak faktur,
 - Jika uang tidak diambil, Uang dimasukkan kembali ke Cash Dispenser,
 - Layar kembali ke menu Utama.



FITUR-FITUR OOP

IF34348 - Pemrograman Lanjut

- **Encapsulation**

Penggabungan antara data (attribut) dengan prosedur (method) yang mengolahnya.

- **Inheritance**

Penurunan sifat (attribut dan method) dari Class Parent (SuperClass) ke Class Child (SubClass). Ini menandakan bahwa OOP mendukung konsep code reuse dimana data-data yang ada di class parent bisa di kenal di kelas child.

- **Polymorphism**

Sebuah kemampuan dari sebuah objek untuk bekerja dalam berbagai bentuk. Penggunaan umum polymorphism biasanya digunakan ketika sebuah reference dari class parent digunakan untuk mengacu ke class child.



CLASS DAN OBJECT

IF34348 - Pemrograman Lanjut

■ Class

Class adalah cetak biru/prototipe/pendefinisian dari suatu benda. Didalam class-lah attribut dan method suatu object didefinisikan.

Contoh : Manusia, Window

■ Object

Object adalah bentuk instance/nyata/real/hidup dari sebuah class.

Contoh :

- Shelly:Manusia (Object Shelly mempunyai Class Manusia)
- Form1:Window (Object Form1 mempunyai class Window)



CLASS DAN OBJECT

IF34348 - Pemrograman Lanjut

- Setiap object pasti memiliki class (sebagai templatanya)
- Setiap object harus diinstansiasi/dihidupkan terlebih dahulu sebelum digunakan. Instansiasi sebuah objek dapat dilakukan dengan keyword new. Contoh berikut :

```
NamaClass NamaObject;
```

```
NamaObject=new NamaClass(parameter_konstruktornya);
```

- Untuk mengakses atribut atau method suatu object, gunakan tanda titik setelah nama objeknya.

```
Kucing catty=new Kucing("Catty");
```

```
catty.warna="putih";
```

```
catty.jalan();
```

Mengakses
atribut/method



CONTOH PENGGUNAAN CLASS

IF34348 - Pemrograman Lanjut

- **String di java sebenarnya adalah sebuah Class.**
- **Method-method yang ada di class String**
 - **charAt(index) :** Mereturnkan huruf pada posisi ke-index. Index 0 menunjukkan huruf pertama.
 - **equals(string_lain) :** Mereturnkan true jika isi string sama dengan isi string_lain (case sensitive).
 - **equalsIgnoreCase(string_lain) :** Mereturnkan true jika isi string sama dengan string lain dengan mode perbandingan case insensitive.
 - **length() :** Mereturnkan berapa banyak huruf dalam string.
 - **toUpperCase() :** Mereturnkan string yang berisi bentuk kapital dari stringnya.
 - **toLowerCase() :** Mereturnkan string yang berisi bentuk huruf kecil dari stringnya.



CONTOH PENGGUNAAN CLASS

IF34348 - Pemrograman Lanjut

```
public class TestString {  
    public static void main(String[] args) {  
        String nama="Universitas Komputer Indonesia";  
        System.out.println("ISI STRING : "+nama);  
        System.out.println("Panjang      : "+nama.length());  
        System.out.println("Upper Case : "+nama.toUpperCase());  
        System.out.println("Lower Case : "+nama.toLowerCase());  
        System.out.println("=UNIKOM      : "+nama.equals("UNIKOM"));  
    }  
}
```

```
ISI STRING : Universitas Komputer Indonesia  
Panjang    : 30  
Upper Case : UNIVERSITAS KOMPUTER INDONESIA  
Lower Case : universitas komputer indonesia  
=UNIKOM    : false
```



MEMBUAT CLASS SEDERHANA

IF34348 - Pemrograman Lanjut

■ Sintak pembuatan class sederhana

```
class NamaKelas{  
    tipe_data nama_atribut;  
    tipe_data nama_atribut;  
    nama_kelas nama_object;  
  
    NamaKelas(parameter){  
        ... // isi konstruktor  
    }  
    void nama_method(parameter){  
        ... // isi method berbentuk procedure  
    }  
    tipe_data nama_method_function(paramter){  
        ... // isi method berbentuk function  
        return ...;  
    }  
}
```

Daftar Atribut

Daftar Method
(Procedure/Function
/Constructor)



CONTOH CLASS SEDERHANA

IF34348 - Pemrograman Lanjut

- **Buatlah sebuah class bernama Titik yang digunakan untuk menyimpan sebuah titik koordinat.**
 - Setiap titik mempunyai atribut posisi X dan atribut posisi Y.
 - Class ini harus dapat melakukan hal berikut :
 - Memberikan nilai default ($X=0$, $Y=0$) ketika X dan Y belum didefinisikan.
 - Mengisi Nilai X
 - Mengisi Nilai Y
 - Mengisi Nilai X dan Y (sekaligus)
 - Menampilkan nilai X dan Y
 - Pindah ke Koordinat Lain (berdasarkan jarak atau ke titik tertentu)
 - Menghitung Jarak Ke Titik Lain



CONTOH CLASS SEDERHANA

IF34348 - Pemrograman Lanjut

```
class Titik {  
    double x;  
    double y;  
    Titik(){  
        x=0;  
        y=0;  
    }  
    Titik(double x1,double y1){  
        x=x1;  
        y=y1;  
    }  
    void tampil(){  
        System.out.println("("+x+", "+y+")");  
    }  
    void pindah(double x1, double y1){  
        x=x1;  
        y=y1;  
    }  
}
```



Oleh : Andri Heryandi, M.T.

CONTOH CLASS SEDERHANA

IF34348 - Pemrograman Lanjut

```
void pindah(Titik t){
    x=t.x;
    y=t.y;
}
void isiX(double x1){
    x=x1;
}
void isiY(double y1){
    y=y1;
}
void isiXY(double x1,double y1){
    x=x1;
    y=y1;
}
```



CONTOH CLASS SEDERHANA

IF34348 - Pemrograman Lanjut

```
double jarakKe(double x1, double y1){  
    double jarak;  
    jarak=Math.pow(Math.pow(x-x1,2)+Math.pow(y-  
y1,2),0.5);  
    return jarak;  
}  
double jarakKe(Titik t2){  
    double jarak;  
    jarak=Math.pow(Math.pow(x-t2.x,2)+Math.pow(y-  
t2.y,2),0.5);  
    return jarak;  
}  
}
```


$$\text{Math.pow}(x,y) = x^y$$

MEMBUAT KELAS TESTER

IF34348 - Pemrograman Lanjut

- Kelas tester digunakan sebagai class untuk melakukan uji coba terhadap class yang telah dibuat.
- Sebaiknya kelas tester dibuat terpisah dari file classnya.



Oleh : Andri Heryandi, M.T.

MEMBUAT KELAS TESTER

IF34348 - Pemrograman Lanjut

```
public class TitikTester {  
    public static void main(String[] args) {  
        Titik t1,t2;  
        t1=new Titik();// Tanpa parameter x=0, y=0  
        t2=new Titik(9,4);// x=9, y=4  
        System.out.print("T1 : ");  
        t1.tampil();  
        System.out.print("T2 : ");  
        t2.tampil();  
        t1.pindah(5,1);  
        System.out.print("Setelah pindah T1 : ");  
        t1.tampil();  
        double jarak;  
        jarak=t1.jarakKe(t2);  
        System.out.println("Jarak dari T1 ke T2 : "+jarak);  
    }  
}
```

```
T1 : (0.0,0.0)  
T2 : (9.0,4.0)  
Setelah pindah T1 : (5.0,1.0)  
Jarak dari T1 ke T2 : 5.0
```



Oleh : Andri Heryandi, M.T.

CLASS CALCULATOR

IF34348 - Pemrograman Lanjut

■ Buatlah class Calculator

■ **Attribut** :

- Operan1 bertipe double
- Operan2 bertipe double

■ **Method** :

- isiOperan1(double x) : Mengisi atribut operan1 dengan nilai x
- isiOperan2(double x) : Mengisi atribut operan2 dengan nilai x
- tambah() : Mereturnkan nilai Operan1 + Operan2
- kurang() : Mereturnkan nilai Operan1 - Operan2
- kali() : Mereturnkan nilai Operan1 * Operan2
- bagi() : Mereturnkan nilai Operan1 / Operan2
- pangkat() : Mereturnkan nilai Operan1^Operan2

■ Kelas tersebut harus bisa dijalankan dengan menggunakan Class CalculatorTester (di slide berikutnya)



CLASS CALCULATORTESTER

IF34348 - Pemrograman Lanjut

```
public class CalculatorTester {  
    public static void main(String[] args) {  
        Calculator c=new Calculator();  
        c.isiOperan1(7);  
        c.isiOperan2(5);  
        System.out.println("Tambah      : "+c.tambah());  
        System.out.println("Kurang     : "+c.kurang());  
        System.out.println("Kali       : "+c.kali());  
        System.out.println("Bagi       : "+c.bagi());  
        System.out.println("Pangkat    : "+c.pangkat());  
    }  
}
```

```
Tambah      : 12.0  
Kurang     : 2.0  
Kali       : 35.0  
Bagi       : 1.4  
Pangkat    : 16807.0
```



Oleh : Andri Heryandi, M.T.

CLASS NILAI

IF34348 - Pemrograman Lanjut

■ Buatlah class Nilai

- **Attribut :**
 - Quis bertipe double
 - UTS bertipe double
 - UAS bertipe double
- **Method :**
 - `setQuis(double x)` : Mengisi nilai quis
 - `setUTS(double x)` : Mengisi nilai UTS
 - `setUAS(double x)` : Mengisi nilai UAS
 - `getNA()` : Mereturnkan nilai akhir berupa double
 - `getIndex()` : Mereturnkan index berupa char
 - `getKeterangan()` : Mereturnkan keterangan berupa String

■ Rumus NilaiAkhir

$$NA = 20\% \text{ QUIZ} + 30\% * UTS + 50\% * UAS$$

■ Aturan Index

- NA 80..100 Index='A'
- NA 68..80 Index='B'
- NA 56..68 Index='C'
- NA 45..56 Index='D'
- NA 0..45 Index='E'

■ Aturan Keterangan

- Index='A' : Sangat Baik
- Index='B' : Baik
- Index='C' : Cukup
- Index='D' : Kurang
- Index='E' : Sangat Kurang



CLASS NILAITESTER

IF34348 - Pemrograman Lanjut

```
public class NilaiTester {  
    public static void main(String[] args) {  
        Nilai n=new Nilai();  
        n.setQuis(60);  
        n.setUTS(80);  
        n.setUAS(75);  
        System.out.println("Quis           : "+n.Quis);  
        System.out.println("UTS           : "+n.UTS);  
        System.out.println("UAS           : "+n.UAS);  
        System.out.println("NA           : "+n.getNA());  
        System.out.println("Index        : "+n.getIndex());  
        System.out.println("Keterangan   : "+n.getKeterangan());  
    }  
}
```

```
Quis           : 60.0  
UTS            : 80.0  
UAS            : 75.0  
NA             : 73.5  
Index          : B  
Keterangan     : Baik
```



Oleh : Andri Heryandi, M.T.

PART II

06

PACKAGE

IF34348 - Pemrograman Lanjut

- Package adalah sekumpulan class/interface/enumerasi yang berelasi.
- Package menyediakan mekanisme untuk mengatur *class* dan *interface* dalam jumlah banyak dan menghindari konflik pada penamaan
- 2 buah class yang mempunyai nama yang sama (tetapi isinya beda) dapat digunakan sekaligus jika berada di masing-masing package.
- Jika anda membuat class tanpa dimasukkan ke dalam package, maka secara default akan dianggap sebagai unnamed package (package tanpa nama). Package tanpa nama tidak bisa diimport oleh class dari package yang mempunyai nama.



MEMBUAT PACKAGE

IF34348 - Pemrograman Lanjut

- Jika anda ingin sebuah class berada dalam suatu package, maka didalam file classnya tambahkan baris berikut
`package <namapackage>`
- Penamaan package sebaiknya menggunakan huruf kecil semua.
- Nama package harus disesuaikan dengan nama folder tempat menyimpan class tersebut.

```
package mylib;  
class Titik {  
    double x;  
    double y;  
}
```



ATURAN PENAMAAN PACKAGE

IF34348 - Pemrograman Lanjut

- Nama package sebaiknya ditulis dengan huruf kecil.
- Nama package harus sama dengan nama folder.
- Suatu file kode sumber hanya boleh menuliskan 1 statement package.
- Perusahaan besar pengembang software biasanya memberikan nama class sesuai dengan nama domainnya.

Contoh :

- Facebook : package com.facebook.android
- Twitter : package com.twitter.interop
- Google maps : package com.google.android.maps



IMPORT PACKAGE

IF34348 - Pemrograman Lanjut

- Import package berarti anda ingin menggunakan member/isi dari sebuah package.
- Import suatu package dilakukan dengan keyword import.
`import namapackage.namaclass; // import 1 class`
atau
`import namapackage.*; // import seluruh isi package.`
- Contoh :
`import mylib.Titik;`
Atau
`import mylib.*;`
- Secara default, jika anda tidak mengimport satu package-pun maka sebenarnya anda mengimport package `java.lang.*` yang contoh class yang ada di dalamnya adalah class `String` dan `Math`. Inilah yang membuat anda tidak usah import package apa pun ketika akan menggunakan class `String` atau `Math`.



IMPORT PACKAGE

IF34348 - Pemrograman Lanjut

- Jika aplikasi yang anda buat menggunakan 2 buah class yang namanya sama tetapi berbeda package yang berbeda maka sebagai pembeda anda harus menuliskan nama packagenya.
 - Contoh berikut akan menampilkan cara pembuatan 2 objek yang mempunyai nama class yang sama.

```
package1.Titik t1=new package1.Titik();  
package2.Titik t2=new package2.Titik();
```



ACCESS MODIFIER

IF34348 - Pemrograman Lanjut

- Pengaturan akses terbagi menjadi 2 level,

- Di level kelas

Pengaturan akses di level kelas terdiri dari 2 cara yaitu

1. Tanpa menuliskan keyword apa pun (default/package-private)
2. Menuliskan keyword **public**

- Di level member

Pengaturan akses di level member terdiri dari 4 cara yaitu :

1. Tanpa menulis keyword apa pun (default/package-private)
2. Menuliskan keyword **public**
3. Menuliskan keyword **private**
4. Menuliskan keyword **protected**



PENGATURAN AKSES DI LEVEL KELAS

IF34348 - Pemrograman Lanjut

- **Pengaturan akses di level kelas hanya memiliki 2 kemungkinan yaitu :**
 - **Default/Package-Private :**

Jika sebuah class dibuat dengan menggunakan akses ini maka class tersebut hanya akan dikenal di class-class pada package yang sama saja.
 - **Public**

Jika sebuah class dibuat dengan menggunakan akses public maka class tersebut dapat dikenal di mana pun, baik dari dalam package yang sama atau dari luar package (world).




PENGATURAN AKSES DI LEVEL KELAS

IF34348 - Pemrograman Lanjut

■ Akses : Default/Package-Private

```
// Nama File : package1/Titik.java
package package1;
class Titik {
    public Titik(){
        System.out.println("Titik di package 1");
    }
}
```

Kedua kelas berada pada package yang sama



```
// Nama File : package1/Tester.java
package package1;
class Tester {
    public static void main(String[] args) {
        Titik t1=new Titik();
    }
}
```

PENGATURAN AKSES DI LEVEL KELAS

IF34348 - Pemrograman Lanjut

■ Akses : Default/Package-Private

```
// Nama File : package1/Titik.java
package package1;
class Titik {
    public Titik(){
        System.out.println("Titik di package 1");
    }
}
```

Class Titik berada di package bernama package1, Class Tester tidak memiliki package

X

```
// Nama File : Tester.java
import package1.Titik;
class Tester {
    public static void main(String[] args) {
        Titik t1=new Titik();
    }
}
```


PENGATURAN AKSES DI LEVEL KELAS

IF34348 - Pemrograman Lanjut

■ Akses : public

```
// Nama File : package1/Titik.java
package package1;
public class Titik {
    public Titik(){
        System.out.println("Titik di package 1");
    }
}
```

Kedua kelas berada pada package yang sama



```
// Nama File : package1/Tester.java
package package1;
class Tester {
    public static void main(String[] args) {
        Titik t1=new Titik();
    }
}
```



PENGATURAN AKSES DI LEVEL KELAS

IF34348 - Pemrograman Lanjut

■ Akses : public

```
// Nama File : package1/Titik.java
package package1;
public class Titik {
    public Titik(){
        System.out.println("Titik di package 1");
    }
}
```

Class Titik berada di package bernama package1, Class Tester tidak memiliki package



```
// Nama File : Tester.java
import package1.Titik;
class Tester {
    public static void main(String[] args) {
        Titik t1=new Titik();
    }
}
```

PENGATURAN AKSES DI LEVEL MEMBER

IF34348 - Pemrograman Lanjut

- **Pengaturan akses di level member memiliki 4 kemungkinan yaitu :**
 - **Public**

Jika member memiliki access modifier ini, maka member ini bisa diakses dari mana saja (world) yaitu dari class sendiri atau dari class-class lain (walaupun berada di package yang berbeda). Member ini dikenal di subclassnya
 - **Protected**

Jika member memiliki access modifier ini, maka member ini bisa diakses dari class sendiri atau dari class-class lain yang se-package. Member ini bisa diakses di subclassnya (class turunannya)
 - **Default/Package-Private :**

Jika member memiliki access modifier ini, maka member ini hanya bisa diakses dari class sendiri dan dari class-class pada package yang sama. Member ini tidak dikenal di subclassnya (class turunan).
 - **Private**

Jika member memiliki access modifier ini, maka member ini bisa diakses dari class sendiri saja. Member ini tidak bisa diakses dari class lain walaupun se-package ataupun dari subclassnya (class turunannya)



PENGATURAN AKSES DI LEVEL MEMBER

IF34348 - Pemrograman Lanjut

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	T
tanpa modifier	Y	Y	T	T
private	Y	T	T	T

Keterangan :

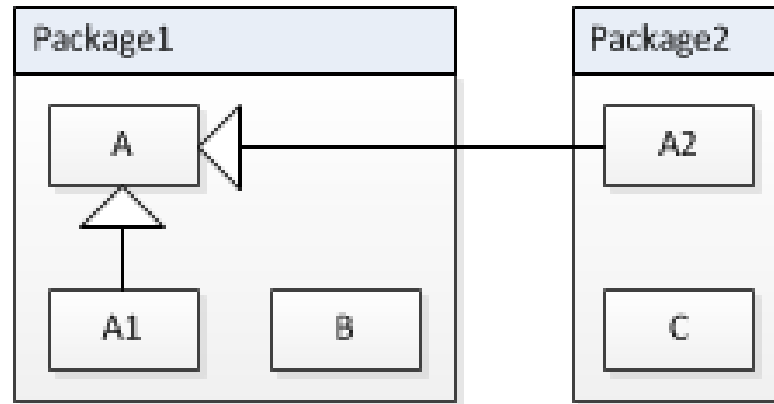
Y : Bisa diakses

T : Tidak bisa diakses



PENGATURAN AKSES DI LEVEL MEMBER

IF34348 - Pemrograman Lanjut



- Visibilitas member-member class A dari class lain.

Modifier Member A	A	A1	B	A2	C
public	Y	Y	Y	Y	Y
protected	Y	Y	Y	Y	T
tanpa modifier	Y	Y	Y	T	T
private	Y	T	T	T	T

ACCESS MODIFIER

IF34348 - Pemrograman Lanjut

Kenapa harus memikirkan Access Modifier?

Alasannya adalah

- Membatasi hak akses. Ini dilakukan agar sebuah member tidak sembarangan dibaca atau diisi
- Menyembunyikan informasi. Pengguna class tidak harus tahu apa yang ada/terjadi di dalam class (Information Hiding)



Oleh : Andri Heryandi, M.T.

CONTOH KASUS ACCESS MODIFIER

IF34348 - Pemrograman Lanjut

- Buatlah sebuah class bernama Nilai yang akan menampung nilai Quis, UTS, UAS. Nilai yang dapat diterima adalah antara 0 sampai 100. Setiap nilai dapat diubah nilainya. Class ini harus dapat diakses dari class mana pun.

```
public class Nilai {  
    public double Quis;  
    public double UTS;  
    public double UAS;  
    public double getNA() {  
        return 0.20*Quis+0.30*UTS+0.5*UAS;  
    }  
}
```

CONTOH KASUS ACCESS MODIFIER

IF34348 - Pemrograman Lanjut

■ Contoh pemanggilan class Nilai

```
public class NilaiTester {  
    public static void main(String[] args) {  
        Nilai n=new Nilai();  
        n.Quis=90;  
        n.UTS=70;  
        n.UAS=150;  
        System.out.println("NA : "+n.getNA());  
    }  
}
```

- Jika aplikasi di atas dieksekusi maka menghasilkan nilai akhir sebesar 114.0. Apakah ini legal??????. Hal ini terjadi karena atribut UAS bersifat public sehingga bisa diisi secara bebas dari luar class.



CONTOH KASUS ACCESS MODIFIER

IF34348 - Pemrograman Lanjut

- Solusinya adalah dengan membuat atribut Quis, UTS dan UAS tidak bisa diisi secara langsung (misalnya dengan private atau protected).

```
public class Nilai {  
    private double Quis;  
    private double UTS;  
    private double UAS;  
    public double getNA(){  
        return 0.20*Quis+0.30*UTS+0.5*UAS;  
    }  
}
```

Tidak, dengan hanya mengubah public menjadi private, maka pengisian nilai Quis, UTS dan UAS tidak bisa dilakukan. Solusinya adalah dengan

CONTOH KASUS ACCESS MODIFIER

IF34348 - Pemrograman Lanjut

- Solusinya adalah dengan membuat atribut Quis, UTS dan UAS tidak bisa diisi secara langsung (misalnya dengan private atau protected).

```
public class Nilai {  
    private double Quis;  
    private double UTS;  
    private double UAS;  
    public double getNA() {  
        return 0.20*Quis+0.30*UTS+0.5*UAS;  
    }  
}
```

- Cukup? Tidak. Karena atribut Quis, UTS dan UAS tidak bisa diakses dari mana pun.



CONTOH KASUS ACCESS MODIFIER (METHOD SETTER DAN GETTER)

IF34348 - Pemrograman Lanjut

- Karena atribut Quis, UTS, dan UAS mempunyai akses private, maka class harus menyediakan suatu cara agar bisa mengakses (mengisi/membaca) nilai atribut tersebut.
- Solusinya adalah method setter dan getter.
 - Method getter adalah method yang digunakan sebagai perantara untuk **mengambil** nilai atribut yang tidak bisa diakses (karena private atau protected). Method getter biasanya berupa function yang mereturnkan tipe data sesuai atribut yang diambil.
 - Method setter adalah method yang digunakan sebagai perantara untuk **mengisi** nilai atribut yang tidak bisa diakses (karena private atau protected). Method setter biasanya berupa procedure (void function) yang mempunyai parameter input yang bertipe data sama dengan tipe data atributnya.
- Method setter dan getter seharusnya mempunyai access modifier public (karena harus bisa diakses oleh semua class).



CONTOH KASUS ACCESS MODIFIER (METHOD SETTER DAN GETTER)

IF34348 - Pemrograman Lanjut

■ Method setter dan getter untuk Quis, UTS, UAS.

```
public class Nilai {  
    private double Quis;  
    private double UTS;  
    private double UAS;  
    public void setQuis(double x){  
        Quis=x;  
    }  
    public void setUTS(double x){  
        UTS=x;  
    }  
    public void setUAS(double x){  
        UAS=x;  
    }  
}
```

Function
setter



CONTOH KASUS ACCESS MODIFIER (METHOD SETTER DAN GETTER)

IF34348 - Pemrograman Lanjut

■ Method setter dan getter untuk Quis, UTS, UAS.

```
public double getQuis() {  
    return Quis;  
}  
public double getUTS() {  
    return UTS;  
}  
public double getUAS() {  
    return UAS;  
}  
public double getNA() {  
    return 0.20*Quis+0.30*UTS+0.5*UAS;  
}  
}
```

Function
getter



CONTOH KASUS ACCESS MODIFIER (METHOD SETTER DAN GETTER)

IF34348 - Pemrograman Lanjut

- Contoh pemanggilan class Nilai berubah, karena pengisian atribut dilakukan melalui function.

```
public class NilaiTester {  
    public static void main(String[] args) {  
        Nilai n=new Nilai();  
        n.setQuis(90);  
        n.setUTS(70);  
        n.setUAS(150);  
        System.out.println("NA : "+n.getNA());  
    }  
}
```

- Beres? Belum, ternyata walau pun atribut telah diberi akses private, ternyata nilai atribut bisa diberi nilai yang tidak seharusnya. Padahal nilai hanya boleh antara 0 s.d 100.

CONTOH KASUS ACCESS MODIFIER (METHOD SETTER DAN GETTER)

IF34348 - Pemrograman Lanjut

- Solusinya adalah dengan membuat validasi pada method getter. Nilai atribut hanya akan berubah jika nilai yang diinput valid.
- Perubahan method getter.

```
public void setQuis(double x) {  
    if(x>=0 && x<=100)  
        Quis=x;  
}  
public void setUTS(double x) {  
    if(x>=0 && x<=100)  
        UTS=x;  
}  
public void setUAS(double x) {  
    if(x>=0 && x<=100)  
        UAS=x;  
}
```



CONTOH KASUS ACCESS MODIFIER (METHOD SETTER DAN GETTER)

IF34348 - Pemrograman Lanjut

- Contoh pemanggilan class Nilai setelah ada validasi di method getter

```
public class NilaiTester {  
    public static void main(String[] args) {  
        Nilai n=new Nilai();  
        n.setQuis(90);  
        n.setUTS(70);  
        n.setUAS(150);  
        System.out.println("NA : "+n.getNA());  
    }  
}
```

- Pada perintah `n.setUAS(150)`, pengisian UAS menjadi 150 tidak dilakukan karena nilainya tidak valid.
- Jika dieksekusi akan menghasilkan NA : 39.0 karena atribut UAS bernilai 0



LATIHAN

IF34348 - Pemrograman Lanjut

- **Buatlah suatu class bernama Waktu dengan spesifikasi class adalah :**

- **Attribut**

- menitWaktu bertipe integer. menitWaktu akan berisi jumlah menit sejak jam 00.00.

- **Method**

- **getJam()** : Mengambil bagian jam dari menitWaktu.
- **getMenit()** : Mengambil bagian menit dari menitWaktu.
- **setJam(int j)** : Menambahkan $j \times 60$ menit ke menitWaktu tanpa mempengaruhi bagian menitnya.
- **setMenit(int m)** : Menambahkan m menit ke menitWaktu tanpa mempengaruhi bagian jamnya.
- **getTotalMenit()** : Mengambil nilai dari menitWaktu



LATIHAN

IF34348 - Pemrograman Lanjut

■ Method (lanjutan)

- tambahJam(int j) : Menambahkan $j*60$ ke menitWaktu
- tambahMenit(int m) : Menambahkan m ke menitWaktu
- tambahWaktu(int j, int m) : Menambahkan $j*60+m$ ke menitWaktu
- tampilWaktu() : Menampilkan jam dan menit.



Oleh : Andri Heryandi, M.T.

CLASS WAKTUTESTER

IF34348 - Pemrograman Lanjut

```
public class WaktuTester {  
    public static void main(String[] args) {  
        Waktu w=new Waktu();  
        System.out.println("1. Total Menit : "+w.getTotalMenit());  
        w.setJam(10);  
        System.out.println("2. Total Menit : "+w.getTotalMenit());  
        w.setMenit(57);  
        System.out.println("3. Total Menit : "+w.getTotalMenit());  
        w.tampilWaktu();  
        w.tambahJam(7);  
        System.out.println("4. JAM : "+w.getJam()+" MENIT : "+w.getMenit());  
        w.tambahMenit(40);  
        System.out.println("5. JAM : "+w.getJam()+" MENIT : "+w.getMenit());  
    }  
}
```

```
1. Total Menit : 0  
2. Total Menit : 600  
3. Total Menit : 657  
Waktu : 10:57  
4. JAM : 17 MENIT : 57  
5. JAM : 18 MENIT : 37
```



Oleh : Andri Heryandi, M.T.

PART III

06

INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

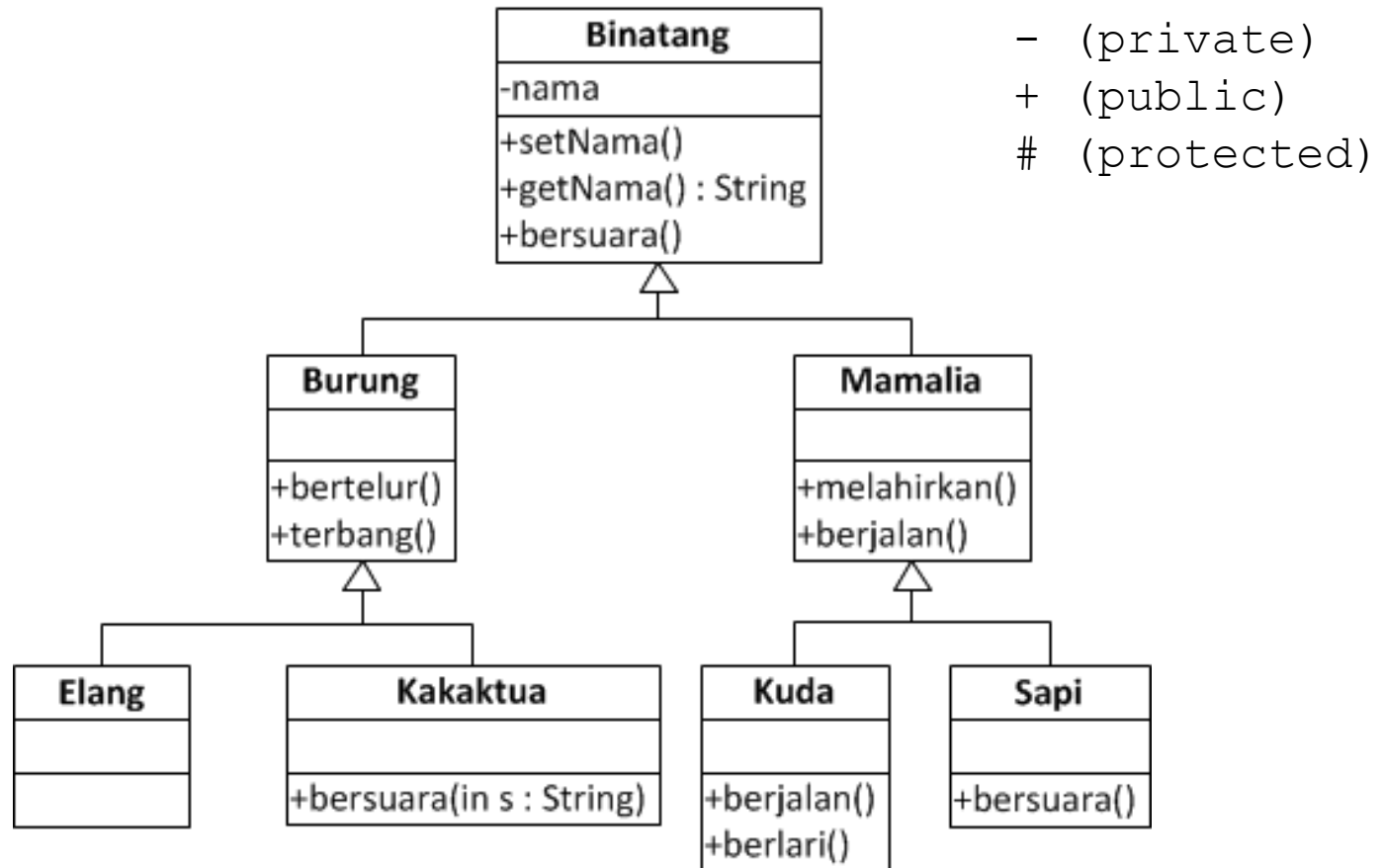
- Inheritance adalah salah satu fitur utama dari OOP.
- Inheritance adalah proses dimana sebuah class mendapatkan properti (atribut + method) dari class lain.
- Dengan inheritance, sebuah class dapat dikembangkan menjadi class baru yang lebih lengkap/baik.
- Class yang diturunkan disebut sebagai superclass (*base class* atau *parent class*) dan class yang menjadi turunan disebut sebagai subclass(*derived class*, *extended class*, atau *child class*).
- Inheritance dilakukan dengan keyword **extends**.
- Class paling atas adalah class bernama **Object**.



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Contoh :



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class Binatang

```
package pewarisan;

public class Binatang {
    private String nama;
    public void setName(String n){
        nama=n;
    }
    public String getName(){
        return nama;
    }
    public void bersuara(){
        System.out.println(getName()+" sedang bersuara.");
    }
}
```

Setiap Binatang mempunyai **kesamaan**, yaitu mempunyai nama dan dapat bersuara.



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class BinatangTester

```
package pewarisan;
```

```
public class BinatangTester {  
    public static void main(String[] args) {  
        Binatang a;  
        a=new Binatang();  
        a.setNama("kuda");  
        a.bersuara();  
    }  
}
```

Tampilan hasil run :
kuda sedang bersuara.

INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class Burung

```
package pewarisan;
```

```
public class Burung extends Binatang {  
    public void bertelur(){  
        System.out.println(getNama()+" sedang bertelur");  
    }  
    public void terbang(){  
        System.out.println(getNama()+" sedang terbang");  
    }  
}
```

Burung adalah Binatang (mempunyai nama dan bersuara) yang dapat bertelur dan terbang. Method bertelur dan terbang adalah method tambahan yang dapat dilakukan oleh class Burung.



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class BurungTester

```
package pewarisan;

public class BurungTester {
    public static void main(String[] args) {
        Burung b;
        b=new Burung();
        b.setNama("Pipit");// diturunkan dari Binatang
        b.bersuara();        // diturunkan dari Binatang
        b.bertelur();        // ditambahkan di Burung
        b.terbang();        // ditambahkan di Burung
    }
}
```

Hasil Run :

Pipit sedang bersuara.

Pipit sedang bertelur

Pipit sedang terbang



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class Mamalia

```
package pewarisan;
```

```
public class Mamalia extends Binatang {  
    public void melahirkan() {  
        System.out.println(getNama()+" sedang melahirkan.");  
    }  
    public void berjalan() {  
        System.out.println(getNama()+" sedang berjalan");  
    }  
}
```

Mamalia adalah Binatang (mempunyai nama dan bersuara) yang dapat melahirkan dan berjalan. Method melahirkan dan berjalan adalah method tambahan yang dapat dilakukan oleh class Mamalia.



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class MamaliaTester

```
package pewarisan;

public class MamaliaTester {
    public static void main(String[] args) {
        Mamalia m;
        m=new Mamalia();
        m.setNama("kuda"); // diturunkan dari Binatang
        m.bersuara();       // diturunkan dari Binatang
        m.berjalan();       // ditambahkan di Mamalia
        m.melahirkan();     // ditambahkan di Mamalia
    }
}
```

Hasil Run :

kuda sedang bersuara.

kuda sedang berjalan

kuda sedang melahirkan.



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class Elang

```
package pewarisan;
```

```
public class Elang extends Burung {  
  
}
```

Elang adalah Burung. Asumsikan bahwa Elang tidak mempunyai perbedaan dari burung secara umum (general) [hanya punya nama dan bersuara].

INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class ElangTester

```
package pewarisan;

public class ElangTester {
    public static void main(String[] args) {
        Elang e=new Elang();
        e.setNama("Elang");
        e.bersuara(); // Diturunkan dari Binatang
        e.bertelur(); // Diturunkan dari Burung
        e.terbang(); // Diturunkan dari Burung
    }
}
```

Hasil Run :

Elang sedang bersuara.

Elang sedang bertelur

Elang sedang terbang



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class Kakaktua

```
package pewarisan;
```

```
public class Kakaktua extends Burung {  
    public void bersuara(String s){  
        System.out.println(getNama()+" bersuara : "+s);  
    }  
}
```

Kakaktua adalah Burung. Tetapi cara bersuara burung kakaktua dapat meniru ucapan manusia. Oleh karena itu method bersuara yang diturunkan dari superclassnya harus dioverride(diganti) sehingga menghasilkan operasi/perilaku yang berbeda dari superclassnya.



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class KakaktuaTester

```
package pewarisan;

public class KakaktuaTester {
    public static void main(String[] args) {
        Kakaktua k=new Kakaktua();
        k.setNama("Kakaktua"); // diturunkan dari Binatang
        k.bertelur();           // diturunkan dari Burung
        k.terbang();            // diturunkan dari Burung
        k.bersuara();           // diturunkan dari Binatang
        k.bersuara("Selamat Datang "); // override di Kakaktua
    }
}
```

Hasil Run :

Kakaktua sedang bertelur

Kakaktua sedang terbang

Kakaktua sedang bersuara.

Kakaktua bersuara : Selamat Datang



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class Kuda

```
package pewarisan;

public class Kuda extends Mamalia {
    public void berjalan(){
        System.out.println(getNama()+
            " berjalan : Tuk tik tak tik tuk tik tak");
    }
    public void berlari(){
        System.out.println(getNama()+" sedang berlari ");
    }
}
```

Kuda adalah Mamalia yang dapat berjalan dan berlari. Bedanya ketika kuda berjalan mengeluarkan suara “Tuk tik tak tik tuk tik tak”, dan kelebihan lain dari kuda adalah dapat berlari.



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class KudaTester

```
package pewarisan;
```

```
public class KudaTester {  
    public static void main(String[] args) {  
        Kuda k=new Kuda();  
        k.setNama("Si Hitam");// diturunkan dari Binatang  
        k.bersuara(); // diturunkan dari Binatang  
        k.melahirkan(); // diturunkan dari Mamalia  
        k.berjalan(); // override di Kuda  
        k.berlari(); // tambahan di Kuda  
    }  
}
```

Hasil Run :

Si Hitam sedang bersuara.

Si Hitam sedang melahirkan.

Si Hitam berjalan : Tuk tik tak tik tuk tik tak

Si Hitam sedang berlari



INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class Sapi

```
package pewarisan;
```

```
public class Sapi extends Mamalia {  
    public void bersuara() {  
        System.out.println(getNama()+" bersuara : Mooooooooooooo");  
    }  
}
```

Sapi adalah Mamalia (biasa) yang kalau bersuara mengeluarkan suara “Mooooooooooooo”. Karena itu maka method bersuara di class Sapi akan mengoverride fungsi bersuara dari class Binatang.

INHERITANCE (PEWARISAN)

IF34348 - Pemrograman Lanjut

■ Class SapiTester

```
package pewarisan;

public class SapiTester {
    public static void main(String[] args) {
        Sapi s=new Sapi();
        s.setNama("Si Moo");
        s.berjalan();
        s.melahirkan();
        s.bersuara();
    }
}
```

Hasil Run :

Si Moo sedang berjalan

Si Moo sedang melahirkan.

Si Moo bersuara : Mooooooooooooo



OVERRIDE CONSTRUCTOR

IF34348 - Pemrograman Lanjut

- Constructor tidak diturunkan ke subclass, tetapi constructor dapat dipanggil dari subclass dengan menggunakan keyword **super**.
- Keyword **super** tidak hanya digunakan untuk memanggil constructor tetapi dapat juga digunakan untuk memanggil method yang ada di superclass.
- Cara pemanggilannya
 - Untuk constructor : `super()`, atau `super(daftar_parameter)`
 - Untuk method : `super.namamethhod(daftar_parameter)`

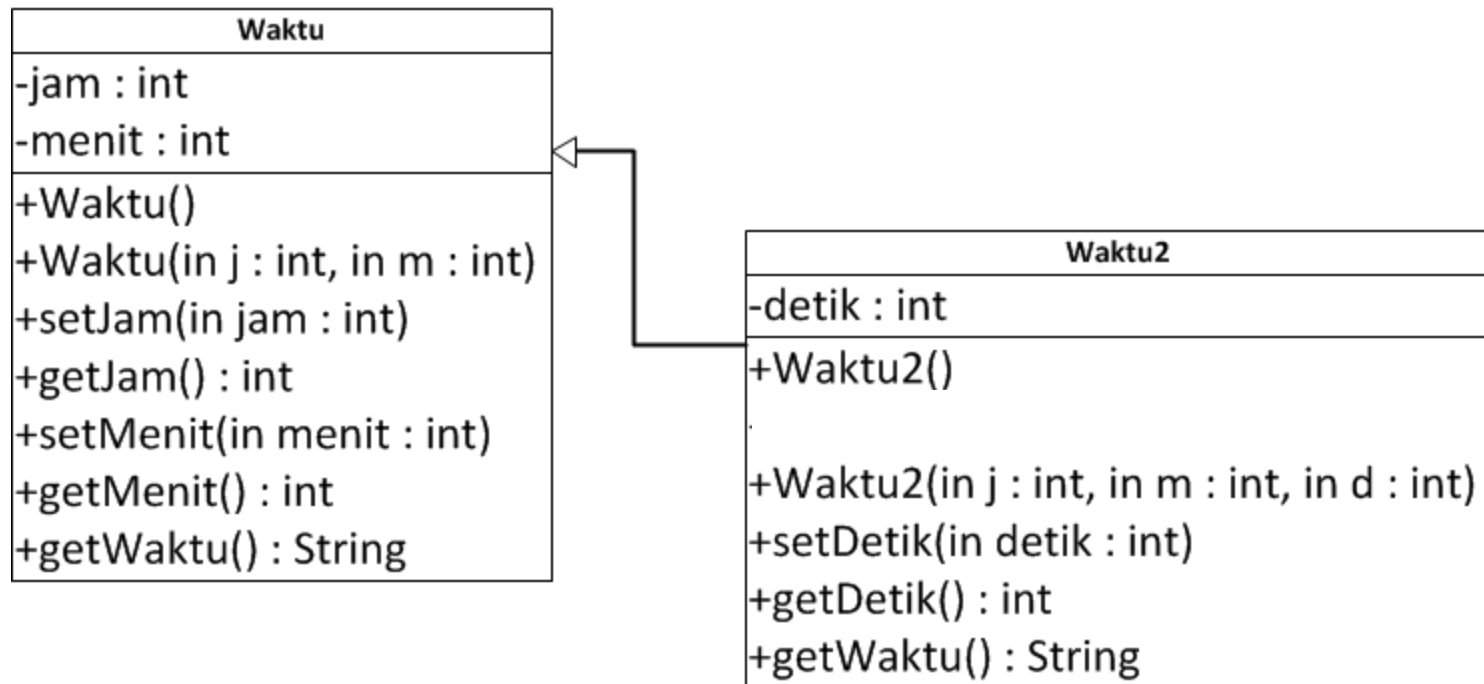


OVERRIDE CONSTRUCTOR

IF34348 - Pemrograman Lanjut

■ Contoh :

- Class Waktu2 merupakan turunan dari class Waktu yang mempunyai atribut baru bernama detik.



OVERRIDE CONSTRUCTOR

IF34348 - Pemrograman Lanjut

■ Class Waktu

```
package pewarisan;
```

```
public class Waktu {  
    private int jam;  
    private int menit;  
  
    public Waktu() {  
        jam=0;  
        menit=0;  
    }  
    public Waktu(int j, int m) {  
        jam=j;  
        menit=m;  
    }  
}
```

Class Waktu mempunyai 2 buah constructor yaitu :

1. Constructor tanpa parameter
2. Constructor dengan parameter j (jam) dan m (menit).



OVERRIDE CONSTRUCTOR

IF34348 - Pemrograman Lanjut

■ Class Waktu

```
public void setJam(int jam){
    this.jam=jam;
}
public int getJam(){
    return jam;
}
public void setMenit(int menit){
    this.menit=menit;
}
public String getWaktu(){
    return jam+": "+menit;
}
}
```

Perhatikan pada method setJam. Method ini memiliki parameter input bernama jam. Class waktu juga mempunyai atribut bernama jam. Untuk membedakan jam dari parameter dan jam milik atribut class, gunakan keyword this. Keyword this menyatakan “milik class ini”.

Pernyataan “**this.jam=jam**” berarti atribut jam milik class ini diisi dengan jam dari parameter.

Begitu pula untuk method setMenit.



OVERRIDE CONSTRUCTOR

IF34348 - Pemrograman Lanjut

■ Class WaktuTester

```
package pewarisan;

public class WaktuTester {
    public static void main(String[] args) {
        Waktu w1, w2;
        w1=new Waktu(); // memanggil constructor tanpa parameter
        System.out.println("W1 : "+w1.getWaktu());
        w2=new Waktu(5,7); // memanggil constructor dengan parameter
        System.out.println("W2 : "+w2.getWaktu());
    }
}
```

Hasil Run :

W1 : 0:0

W2 : 5:7



OVERRIDE CONSTRUCTOR

IF34348 - Pemrograman Lanjut

■ Class Waktu2

```
package pewarisan;
```

```
public class Waktu2 extends Waktu {  
    private int detik;  
    public Waktu2(){  
        super(); // memanggil constructor class super tanpa parameter  
        detik=0;  
    }  
    public Waktu2(int j, int m, int d){  
        super(j,m); // memanggil konstruktor class super yang menggunakan  
parameter  
        detik=d;  
    }  
}
```



OVERRIDE CONSTRUCTOR

IF34348 - Pemrograman Lanjut

■ Class Waktu2Tester

```
public void setDetik(int detik){
    this.detik=detik;
}
public int getDetik(){
    return this.detik; // sama saja dengan return detik;
}
public String getWaktu(){
    return super.getWaktu()+":"+detik;
    // atau return getJam()+":"+getMenit()+":"+detik;
}
}
```



OVERRIDE CONSTRUCTOR

IF34348 - Pemrograman Lanjut

■ Class Waktu2

```
package pewarisan;
```

```
public class Waktu2Tester {
```

```
    public static void main(String[] args) {  
        Waktu2 w1,w2;  
        w1=new Waktu2();  
        System.out.println("W1 : "+w1.getWaktu());  
        w2=new Waktu2(10,12,20);  
        System.out.println("W2 : "+w2.getWaktu());  
    }  
}
```

Hasil Run :

W1 : 0:0:0

W2 : 10:12:20



POLYMORPH

IF34348 - Pemrograman Lanjut

- Polymorph (mempunyai banyak bentuk).
- Jenis-jenis polymorph bisa terjadi adalah :
 - Polymorph Ad Hoc
 - Polymorph Subtyping



Oleh : Andri Heryandi, M.T.

POLYMORPH

IF34348 - Pemrograman Lanjut

Polymorph Ad Hoc terjadi ketika memiliki sebuah method yang akan mempunyai operasi yang berbeda ketika dipanggil dengan tipe parameter yang berbeda.

```
package pewarisan;

public class Polymorph {
    public int tambah(int x, int y){
        return x+y;
    }
    public String tambah(String x, String y){
        return x+" "+y;
    }
}
```

Class Polymorph mempunyai 2 buah method yang namanya sama yaitu “tambah”, tetapi dibedakan dengan parameternya.



POLYMORPH

IF34348 - Pemrograman Lanjut

■ Class PolymorphTester

```
package pewarisan;
```

```
public class PolymorphTester {
```

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

```
        Polymorph p=new Polymorph();
```

```
        System.out.println("2 + 3 = "+p.tambah(2, 3));
```

```
        System.out.println("\"2\" + \"3\" = "+p.tambah("2", "3"));
```

```
    }
```

```
}
```

Hasil Run :

2 + 3 = 5

"2" + "3" = 2 3



POLYMORPH

IF34348 - Pemrograman Lanjut

Polymorph Subtyping terjadi sebuah referensi dari sebuah class diakses dengan menggunakan referensi subclassnya.

```
package pewarisan;
```

```
public class PolySubtype {  
    public static void main(String[] args) {  
        Binatang elang,kakaktua,kuda,sapi;  
        elang=new Elang();elang.setNama("Elang");  
        kakaktua=new Kakaktua();kakaktua.setNama("Kakaktua");  
        kuda=new Kuda();kuda.setNama("Si Pony");  
        sapi=new Sapi();sapi.setNama("Si Moo");  
        elang.bersuara();  
        kakaktua.bersuara();  
        kuda.bersuara();  
        sapi.bersuara();  
    }  
}
```

Seluruh objek menggunakan superclass. Tetapi tetap akan mengeksekusi method masing-masing class.

Hasil Run :

Elang sedang bersuara.

Kakaktua sedang bersuara.

Si Pony sedang bersuara.

Si Moo bersuara : Moooooooooooo



POLYMORPH

IF34348 - Pemrograman Lanjut

Contoh Lain :

```
package pewarisan;
public class PolySubtype2 {
    static void bicara(Binatang b){
        b.bersuara();
    }
    public static void main(String[] args) {
        Elang e=new Elang();e.setNama("Hunter");
        Kakaktua k=new Kakaktua();k.setNama("Oces");
        Kuda kd=new Kuda();kd.setNama("Blacky");
        Sapi s=new Sapi();s.setNama("Moooo");
        bicara(e);
        bicara(k);
        bicara(kd);
        bicara(s);
    }
}
```

Pemanggilan
method Bicara
menggunakan
parameter
subclass dari
Binatang.

Method bicara membutuhkan
parameter dengan class
Binatang.

Hasil Run :
Hunter sedang bersuara.
Oces sedang bersuara.
Blacky sedang bersuara.
Moooo bersuara : Moooooooooooo



Oleh : Andri Heryandi, M.T.

PART IV (ABSTRACT)

06

ABSTRACT

IF34348 - Pemrograman Lanjut

- Class abstract adalah class yang tidak dapat diinstansiasikan.
- Semua member dari class tetap ada tetapi tidak bisa diinstansikan.
- Jika class abstract ingin diinstansiasikan, maka harus diturunkan terlebih dahulu ke subclassnya.
- Abstract dapat digunakan pada level Class atau Method.
- Untuk menyatakan sebuah class adalah class abstract, tambahkan keyword abstract ketika mendefinisikan class.
- Class abstract dikembangkan dengan keyword extends.
- Abstract terjadi ketika seorang pendesain sistem baru memikirkan rencana method yang ada tetapi belum terlalu memperdulikan bagaimana implementasinya.



CLASS ABSTRACT

IF34348 - Pemrograman Lanjut

■ Contoh Class biasa

```
public class Pegawai {  
    private String NIP;  
    private String nama;  
    public Pegawai(String nama, String NIP){  
        this.nama=nama;  
        this.NIP=NIP;  
    }  
    public String getName(){  
        return nama;  
    }  
    public String getNIP(){  
        return NIP;  
    }  
    public void kirimEmail(String to, String subjek, String isi){  
        System.out.println(getName()+" Kirim email ke : "+to+"\n"+  
                             "Dengan Subjek : "+subjek+"\n"+  
                             "Dengan Isi      : "+isi);  
    }  
}
```



Oleh : Andri Heryandi, M.T.

CLASS ABSTRACT

IF34348 - Pemrograman Lanjut

■ Contoh Class Abstract

```
public abstract class Pegawai {  
    private String NIP;  
    private String nama;  
    public Pegawai(String nama, String NIP){  
        this.nama=nama;  
        this.NIP=NIP;  
    }  
    public String getNama(){  
        return nama;  
    }  
    public String getNIP(){  
        return NIP;  
    }  
    public void kirimEmail(String to, String subjek, String isi){  
        System.out.println(getNama()+" Kirim email ke : "+to+"\n"+  
                             "Dengan Subjek : "+subjek+"\n"+  
                             "Dengan Isi      : "+isi);  
    }  
}
```



Oleh : Andri Heryandi, M.T.

CLASS ABSTRACT

IF34348 - Pemrograman Lanjut

■ Contoh Class Tester

```
public class PegawaiTester {  
  
    public static void main(String[] args) {  
        Pegawai p=new Pegawai("Januar","1234");  
        p.kirimEmail("a@test.com","judul","isi email");  
    }  
  
}
```

■ Jika di-run akan menghasilkan error:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Cannot instantiate the type Pegawai

at PegawaiTester.main(PegawaiTester.java:5)



CLASS ABSTRACT

IF34348 - Pemrograman Lanjut

■ Menurunkan Class Abstract

```
public class Staff extends Pegawai {  
    private String bagian;  
    public Staff(String nama, String NIP, String bagian){  
        super(nama,NIP);  
        setBagian(bagian);  
    }  
    public void setBagian(String namabagian){  
        bagian=namabagian;  
    }  
    public String getBagian(){  
        return bagian;  
    }  
}
```



CLASS ABSTRACT

IF34348 - Pemrograman Lanjut

■ Test kelas turunan

```
public class StaffTester {  
    public static void main(String[] args) {  
        Staff s=new Staff("Januar","1234","Keuangan");  
        s.kirimEmail("a@test.com", "info test", "isi email");  
        System.out.println("NIP      : "+s.getNIP()+"\n"+  
                           "Nama      : "+s.getNama()+"\n"+  
                           "Bagian    : "+s.getBagian());  
    }  
}
```

■ Tampilan Output :

```
Januar Kirim email ke : a@test.com  
Dengan Subjek : info test  
Dengan Isi      : isi email  
NIP      : 1234  
Nama      : Januar  
Bagian    : Keuangan
```



METHOD ABSTRACT

IF34348 - Pemrograman Lanjut

- Method abstract adalah suatu metode di dalam class abstract yang belum ditentukan isinya (hanya nama method dan parameternya saja).
- Method abstract harus diimplementasikan (ditulis isi methodnya) di subclassnya.



METHOD ABSTRACT

IF34348 - Pemrograman Lanjut

■ Class Pegawai

```
public abstract class Pegawai {  
    private String NIP;  
    private String nama;  
    public Pegawai(String nama, String NIP){  
        this.nama=nama;  
        this.NIP=NIP;  
    }  
    public String getNama(){  
        return nama;  
    }  
    public String getNIP(){  
        return NIP;  
    }  
    public abstract void kirimEmail(String to, String subjek, String isi);  
}
```

Hanya mendeklarasikan nama method dan parameter (tanpa isi method)

METHOD ABSTRACT

IF34348 - Pemrograman Lanjut

■ Isi method abstract dikembangkan di sub class

```
public class Staff extends Pegawai {  
    private String bagian;  
    public Staff(String nama, String NIP, String bagian){  
        super(nama,NIP);  
        setBagian(bagian);  
    }  
    public void setBagian(String namabagian){  
        bagian=namabagian;  
    }  
    public String getBagian(){  
        return bagian;  
    }  
    public void kirimEmail(String to, String subjek, String isi){  
        System.out.println("Kirim ke "+to+" dengan judul "+  
            subjek+" dengan isi : "+isi);  
    }  
}
```

Isi method dikembangkan di subclass.



PART IV (INTERFACE)

06

INTERFACE

IF34348 - Pemrograman Lanjut

- Interface adalah sekumpulan method abstract.
- Interface adalah kontrak interaksi objek dengan dunia luar.
- Class dapat mengembangkan sebuah interface dengan cara mengimplementasikannya.
- Cara menuliskan interface mirip dengan cara menuliskan class, tetapi interface bukanlah class.
- Perbedaan interface dengan class:
 - Interface dikembangkan dengan keyword implements
 - Class dikembangkan/diturunkan dengan keyword extends
 - Semua method yang ada di interface **WAJIB** diimplementasikan di class yang mengimplementasikannya.
 - Setiap method yang ada di interface harus diimplementasikan, sedangkan pada class tidak semua method harus dioverride.



CONTOH KASUS INTERFACE

IF34348 - Pemrograman Lanjut

- Kita ingin membuat sebuah aplikasi untuk menghitung luas dan keliling dari berbagai bidang 2 Dimensi. Di setiap bidang 2 Dimensi pasti memiliki 2 method yang sama yaitu getLuas dan getKeliling. Tetapi implementasi/isi dari kedua method tersebut berbeda-beda untuk setiap jenis bidangnya.

- Persegipanjang

- Atribut : panjang, lebar
- Luas : panjang * lebar
- Keliling : $2 * (\text{panjang} + \text{lebar})$

- Bujursangkar

- Atribut : sisi
- Luas : sisi * sisi
- Keliling : $4 * \text{sisi}$

- Lingkaran

- Atribut : radius
- Luas : $\pi * \text{radius} * \text{radius}$
- Keliling : $2 * \pi * \text{radius}$



Oleh : Andri Heryandi, M.T.

CONTOH INTERFACE

IF34348 - Pemrograman Lanjut

■ Interface Bidang2D

```
public interface Bidang2D {  
    double getKeliling();  
    double getLuas();  
}
```



CONTOH INTERFACE

IF34348 - Pemrograman Lanjut

■ Interface Bidang2D

```
public interface Bidang2D {  
    double getKeliling();  
    double getLuas();  
}
```



CONTOH INTERFACE

IF34348 - Pemrograman Lanjut

- Mengimplementasikan Interface Bidang2D sebagai Persegipanjang

```
public class Persegipanjang implements Bidang2D {  
    public double panjang;  
    public double lebar;  
    public double getKeliling() {  
        return 2*(panjang+lebar);  
    }  
    public double getLuas() {  
        return panjang*lebar;  
    }  
}
```



CONTOH INTERFACE

IF34348 - Pemrograman Lanjut

- Mengimplementasikan Interface Bidang2D sebagai Bujursangkar

```
public class Bujursangkar implements Bidang2D {  
    public double sisi;  
    public double getKeliling() {  
        return 4*sisi;  
    }  
    public double getLuas() {  
        return sisi*sisi;  
    }  
}
```



CONTOH INTERFACE

IF34348 - Pemrograman Lanjut

- Mengimplementasikan Interface Bidang2D sebagai Lingkaran

```
public class Lingkaran implements Bidang2D {  
    public double radius;  
    public double getKeliling() {  
        return 2*Math.PI*radius;  
    }  
    public double getLuas() {  
        return Math.PI*radius*radius;  
    }  
}
```



CONTOH INTERFACE

IF34348 - Pemrograman Lanjut

■ Class Tester : Bidang2DTester

```
public class Bidang2DTester {
    public static void main(String[] s){
        Lingkaran l=new Lingkaran();
        Bujursangkar b=new Bujursangkar();
        Persegipanjang p=new Persegipanjang();
        l.radius=10;
        System.out.println("Lingkaran dengan radius "+l.radius);
        System.out.println("Luas : "+l.getLuas()+" Keliling : "+
l.getKeliling());
        b.sisi=15;
        System.out.println("Bujursangkar dengan sisi "+b.sisi);
        System.out.println("Luas : "+b.getLuas()+" Keliling : "+
b.getKeliling());
        p.panjang=5;
        p.lebar=6;
        System.out.println("Persegi dengan panjang : "+p.panjang+
" lebar : "+p.lebar);
        System.out.println("Luas : "+p.getLuas()+" Keliling : "+
p.getKeliling());
    }
}
```

Oleh : Andri Heryandi, M.T.



CONTOH INTERFACE

IF34348 - Pemrograman Lanjut

■ Hasil run Bidang2DTester :

Lingkaran dengan radius 10.0

Luas : 314.1592653589793 Keliling : 62.83185307179586

Bujursangkar dengan sisi 15.0

Luas : 225.0 Keliling : 60.0

Persegi dengan panjang : 5.0 lebar : 6.0

Luas : 30.0 Keliling : 22.0

CONTOH INTERFACE (POLYMORPH)

IF34348 - Pemrograman Lanjut

■ Class Tester : Bidang2DTesterPoly

```
public class Bidang2DTesterPoly {  
    public static void main(String[] args) {  
        Bidang2D bidang;  
        Lingkaran l=new Lingkaran();l.radius=10;  
        Bujursangkar b=new Bujursangkar();b.sisi=15;  
        bidang=l; // bidang2d sebagai lingkaran  
        System.out.println("Luas : "+bidang.getLuas()+" Keliling : "+  
bidang.getKeliling());  
        bidang=b; // bidang2d sebagai bujursangkar  
        System.out.println("Luas : "+bidang.getLuas()+" Keliling : "+  
bidang.getKeliling());  
    }  
}
```

Hasil Run:

Luas : 314.1592653589793

Keliling : 62.83185307179586

Luas : 225.0 Keliling : 60.0



IMPLEMENTASI MULTIPLE INTERFACE

IF34348 - Pemrograman Lanjut

- Di Java, sebuah kelas hanya boleh menjadi turunan dari 1 superclass. Tidak ada multiple-inheritance di java. Tapi Java memperbolehkan sebuah class mengimplementasikan lebih dari interface.
- Contoh :
 - Class Mobil mempunyai method berjalan();
 - Class Helikopter mempunyai method Terbang();
 - Class Mobikopter (gabungan mobil dan helikopter) seharusnya memiliki kedua method tersebut.



IMPLEMENTASI MULTIPLE INTERFACE

IF34348 - Pemrograman Lanjut

■ Interface Mobil

```
public interface Mobil {  
    void berjalan();  
    // dan method-method lainnya  
}
```

■ Interface Helikopter

```
public interface Helikopter {  
    void terbang();  
    // dan method-method lainnya  
}
```



IMPLEMENTASI MULTIPLE INTERFACE

IF34348 - Pemrograman Lanjut

■ Class Mobikopter

```
public class Mobikopter implements Mobil, Helikopter{  
    private String nama;  
    Mobikopter(String nama){  
        this.nama=nama;  
    }  
    public void berjalan(){  
        System.out.println(nama+" sedang berjalan");  
    }  
    public void terbang(){  
        System.out.println(nama+" sedang terbang");  
    }  
}
```



IMPLEMENTASI MULTIPLE INTERFACE

IF34348 - Pemrograman Lanjut

■ Class MobikopterTester

```
public class MobilkopterTester {  
    public static void main(String[] args) {  
        Mobikopter m=new Mobikopter("Moko");  
        m.berjalan();  
        m.terbang();  
    }  
}
```

Hasil Run:

Moko sedang berjalan

Moko sedang terbang



PART IV (FINAL)

06

FINAL

IF34348 - Pemrograman Lanjut

- **Keyword final digunakan untuk menyatakan bahwa sebuah class/atribut/method tidak boleh lagi diubah-ubah (termasuk di subclassnya).**
- **Keyword final dapat digunakan dalam**
 - **Atribut**

Jika atribut suatu class beri keyword final, maka atribut tersebut akan berperan sebagai konstanta (dimana nilainya tidak boleh diubah-ubah lagi).
 - **Method**

Jika sebuah method yang ada di suatu class diberi keyword final, maka method tersebut tidak bisa dioverride di subclassnya.
 - **Class**

Jika sebuah class diberi keyword final maka class tersebut tidak bisa lagi diturunkan.



FINAL ATRIBUT/METHOD

IF34348 - Pemrograman Lanjut

■ Class MyMath

```
public class MyMath {  
    public final double PI=3.1416;// empat digit pecahan  
    public final double luasLingkaran(double radius){  
        return PI*radius*radius;  
    }  
    public final double kelilingLingkaran(double radius){  
        return 2*PI*radius;  
    }  
    public final double sin(double derajat){  
        return Math.sin(Math.toRadians(derajat));  
    }  
    public final double cos(double derajat){  
        return Math.cos(Math.toRadians(derajat));  
    }  
    public final double tan(double derajat){  
        return Math.tan(Math.toRadians(derajat));  
    }  
    public final double pangkat(double x, double y){  
        return Math.pow(x, y);  
    }  
}
```

Atribut dari class MyMath tidak bisa diubah dari mana pun termasuk dari class dirinya sendiri

Method dari class MyMath tidak bisa dioverride di subclass walau pun classnya boleh di-extends (diturunkan)



FINAL CLASS

IF34348 - Pemrograman Lanjut

■ Class MyMath

```
public final class MyMath {  
    public final double PI=3.1416;// empat digit pecahan  
    public final double luasLingkaran(double radius){  
        return PI*radius*radius;  
    }  
    public final double kelilingLingkaran(double radius){  
        return 2*PI*radius;  
    }  
    public final double sin(double derajat){  
        return Math.sin(Math.toRadians(derajat));  
    }  
    public final double cos(double derajat){  
        return Math.cos(Math.toRadians(derajat));  
    }  
    public final double tan(double derajat){  
        return Math.tan(Math.toRadians(derajat));  
    }  
    public final double pangkat(double x, double y){  
        return Math.pow(x, y);  
    }  
}
```

Class MyMath tidak bisa di-extends ke subclass.



FINAL ATRIBUT/METHOD

IF34348 - Pemrograman Lanjut

■ Class MyMathTester

```
public class MyMathTester {  
    public static void main(String[] args) {  
        MyMath m=new MyMath();  
        System.out.println("PI : "+m.PI);  
        // m.PI=25.0; // pasti error  
        System.out.println("Luas Lingkaran : "+m.luasLingkaran(10));  
        System.out.println("Keliling Lingkaran : "+  
m.kelilingLingkaran(10));  
        System.out.println("Sin(30) : "+m.sin(30));  
        System.out.println("Cos(30) : "+m.cos(30));  
        System.out.println("Tan(30) : "+m.tan(30));  
        System.out.println("2^8 : "+m.pangkat(2,8));  
    }  
}
```

Hasil Run:

```
PI : 3.1416  
Luas Lingkaran : 314.16  
Keliling Lingkaran : 62.832  
Sin(30) : 0.49999999999999994  
Cos(30) : 0.8660254037844387  
Tan(30) : 0.5773502691896257  
2^8 : 256.0
```



PART IV (STATIC)

06

STATIC

IF34348 - Pemrograman Lanjut

- Keyword static digunakan untuk menyatakan bahwa sebuah member (atribut/method) akan ada/aktif secara independen walaupun classnya tidak diinstansiasi.
- Contoh atribut/method static adalah atribut/method yang ada di class Math. Kita tidak harus menginstansiasi class ini untuk menjadi objek, tetapi kita bisa gunakan atribut/methodnya (Math.PI, Math.sin(), Math.cos() dll).
- Atribut/Method static tidak dimiliki oleh object, tetapi oleh classnya.
- Keyword static dapat digunakan dalam
 - Atribut

Jika atribut suatu class diberi keyword static, maka atribut tersebut akan dapat diakses dari mana pun tanpa harus diinstansiasikan.
 - Method

Jika sebuah method yang ada di suatu class diberi keyword static maka method tersebut akan dapat diakses dari mana pun tanpa harus diinstansiasikan classnya.



STATIC

IF34348 - Pemrograman Lanjut

- **Atribut/Method static** dipanggil dengan menyebutkan nama classnya diikuti dengan nama atribut/methodnya.

Contoh :

NamaClass>NamaAtribut

NamaClass>NamaMethod()

Contoh :

Math.PI

Math.sin()

Math.cos()



STATIC

IF34348 - Pemrograman Lanjut

■ Class MyMath

```
public final class MyMath {  
    public static final double PI=3.1416;  
    public static double luasLingkaran(double radius){  
        return PI*radius*radius;  
    }  
    public static double kelilingLingkaran(double radius){  
        return 2*PI*radius;  
    }  
    public static double sin(double derajat){  
        return Math.sin(Math.toRadians(derajat));  
    }  
    public static double cos(double derajat){  
        return Math.cos(Math.toRadians(derajat));  
    }  
    public static double tan(double derajat){  
        return Math.tan(Math.toRadians(derajat));  
    }  
    public static double pangkat(double x, double y){  
        return Math.pow(x, y);  
    }  
}
```



Oleh : Andri Heryandi, M.T.

STATIC

IF34348 - Pemrograman Lanjut

■ Contoh penggunaan Atribut/Method Static

```
public class MyMathTester {  
    public static void main(String[] args) {  
        System.out.println("PI : "+MyMath.PI);  
        System.out.println("Luas Lingkaran : "+MyMath.luasLingkaran(10));  
        System.out.println("Keliling Lingkaran : "+MyMath.kelilingLingkaran(10));  
        System.out.println("Sin(30) : "+MyMath.sin(30));  
        System.out.println("Cos(30) : "+MyMath.cos(30));  
        System.out.println("Tan(30) : "+MyMath.tan(30));  
        System.out.println("2^8 : "+MyMath.pangkat(2,8));  
    }  
}
```

Hasil Run:

```
PI : 3.1416  
Luas Lingkaran : 314.16  
Keliling Lingkaran : 62.832  
Sin(30) : 0.49999999999999994  
Cos(30) : 0.8660254037844387  
Tan(30) : 0.5773502691896257  
2^8 : 256.0
```



JENIS – JENIS VARIABLE

IF34348 - Pemrograman Lanjut

■ Jenis-jenis variable di Java terbagi 3 yaitu

■ Variable local

Variable yang ada di dalam constructor atau method. Hanya dikenal secara local di method tempat variable tersebut dideklarasikan.

■ Variable instance

Variable yang menjadi dideklarasikan di dalam kelas tetapi berada di luar pendefinisian method/constructor. Variable ini dimiliki oleh setiap objek (menjadi atribut dari objek). Beda objek bisa memiliki nilai variable yang berbeda.

■ Variable class/static

Variable class/static adalah variable yang dideklarasikan dengan keyword static. Variable ini akan dapat diakses walaupun classnya tidak diinstansiasikan.



**JANGAN
TERTUKAR!!!!!!**

JENIS – JENIS VARIABLE

IF34348 - Pemrograman Lanjut

■ Contoh Class :

```
public class MyVar {  
    public static int myStatic;  
    public int myInstance;  
    private String myName;  
    public void test() {  
        int i;  
        i=0;  
        for(int j=1; j<=10; j++){  
            i+=2;  
        }  
        myInstance=i;  
    }  
}
```

Diagram illustrating variable types in the provided Java code:

- Variable Static:** Points to the declaration of `myStatic`.
- Variable Instance:** Points to the declaration of `myInstance` and `myName`.
- Variable Local:** Points to the declaration of `i` and the loop variable `j` (highlighted with a red circle).