

**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

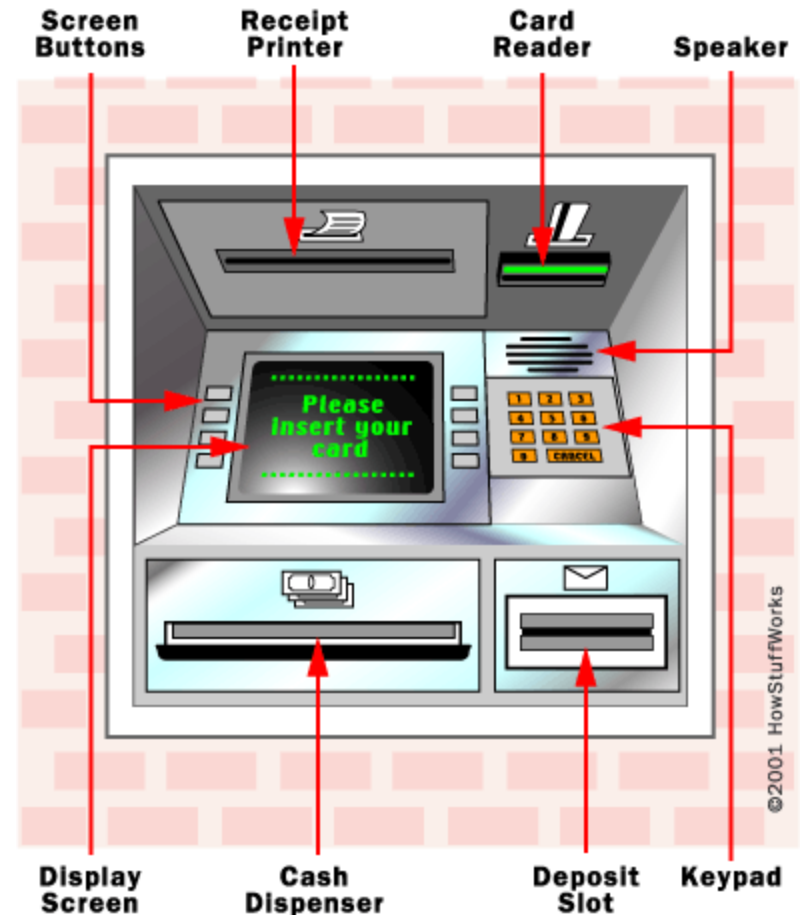
- **Attribut** : ayah, Ibu, Anak[]
- **Method** : TambahAnak,



# CONTOH

IF34348 - Pemrograman Lanjut

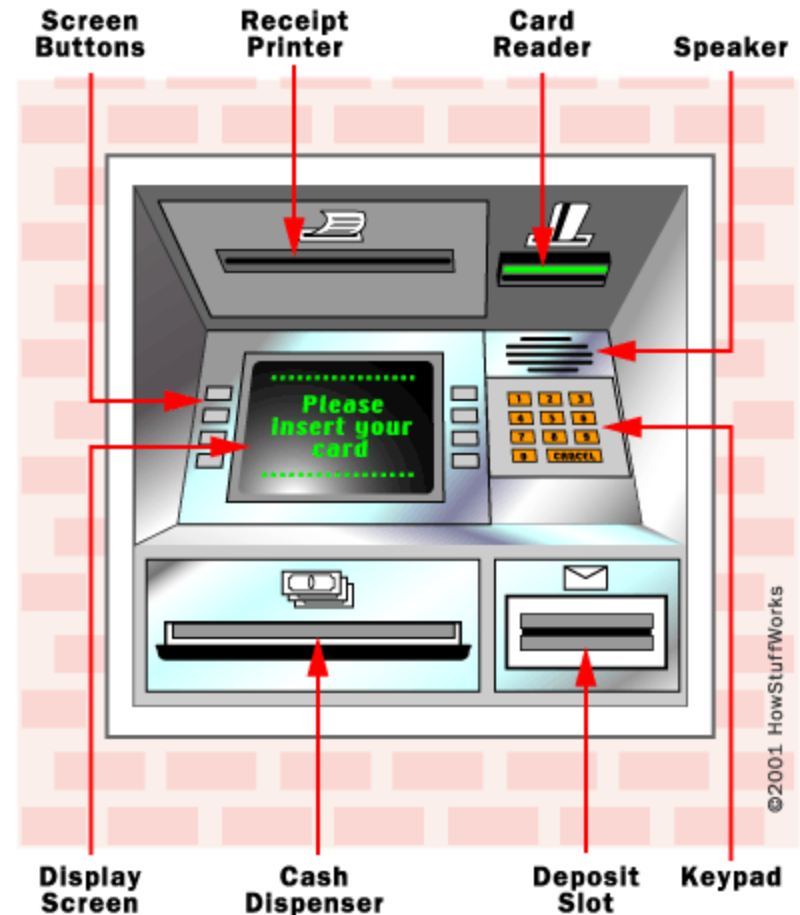
- Perhatikan mesin ATM
- Sebuah mesin ATM terdiri dari elemen-elemen berikut :
  - Display Screen (Layar)
  - Screen Button (Tombol)
  - Receipt Printer
  - Card Reader
  - Speaker
  - Cash Dispenser
  - Deposit Slot
  - Keypad
- Di dalam konsep OOP, setiap setiap elemen tersebut berinteraksi dengan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

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

## ■ 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 :

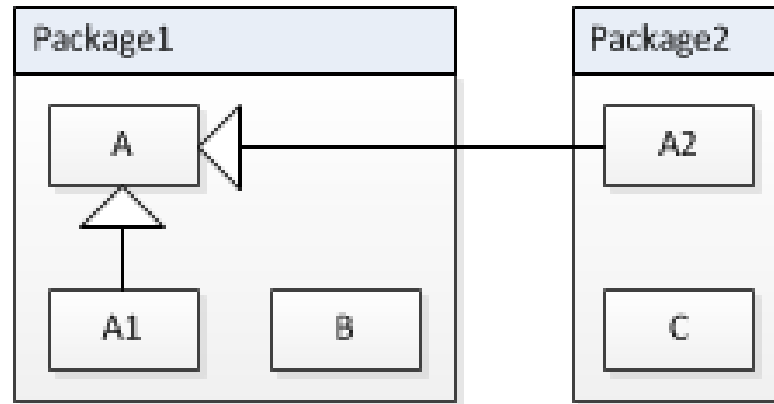
Y : Bisa diakses

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)



# 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.