

Object Oriented Programming (OOP)

--Encapsulation--

Saniati

saniati@teknokrat.ac.id

STMIK Teknokrat, Bandar Lampung

Karakteristik OOP

- **Encapsulation**
- Inheritance
- Polimorphisme

Encapsulation

- Encapsulation merupakan kemampuan untuk membuat *user defined data type*, dengan cara membungkus (mengkapsulkan) *method* dan *fields* menjadi sebuah *class*.
- *Class* : Blueprint dari sebuah objek (cetakan dari sebuah objek)
- Contoh *class* : Mobil, Binatang, Manusia.
- *Class* biasanya masih mengacu kepada sebuah *abstract/general entity* (bukan spesifik *entity*).

Mendesain Class

- Amati object yang akan dibuat classnya lalu identifikasi:
 - Dia punya apa? Disebut fields/attribute. Diimplementasikan menjadi variabel.
 - Dia bisa apa? Disebut method/behavior. Diimplementasikan menjadi method/function/procedure.
- Contoh desain class:
 - Nama Class : Mobil
 - Field : mesin, roda, body
 - Method : maju, mundur, belok

Contoh Pembuatan Class

```
public class Mobil {
    int mesin=1;
    int roda=4;
    int body=1;

    void maju(){
        System.out.println("maju");
    }

    void mundur(){
        System.out.println("mundur");
    }

    void belok(){
        System.out.println("belok");
    }
}
```

```
public class Main2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Mobil avanza = new Mobil();
        avanza.maju();
        avanza.mundur();
        avanza.belok();

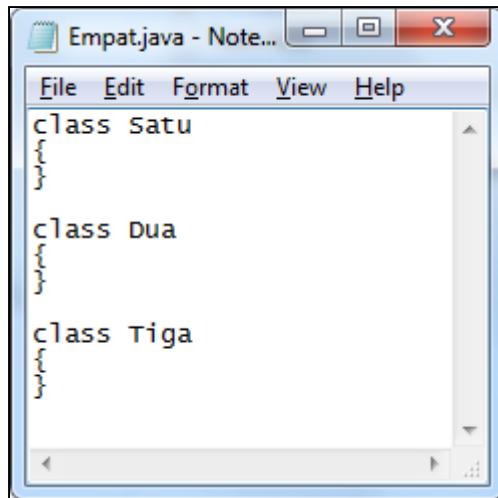
        System.out.println("Mesin : "+avanza.mesin);
        System.out.println("Roda : "+avanza.roda);
        System.out.println("Body : "+avanza.body);
    }
}
```

Aturan penamaan Class dan File

- Jika dalam file.java, tidak berlabel “public”, maka nama file BEBAS
- Jika dalam file .java ada class yang berlabel “public”, maka nama file HARUS sama dengan nama class yang berlabel “public”
- Dalam sebuah file .java, TIDAK BOLEH ada lebih dari 1 class yang berlabel “public”

Contoh Class Bukan “public” /”default”

- Untuk class tanpa label “public”, seperti berikut, nama file BEBAS.
- Ketika di compile, yang diubah menjadi .class adalah class yang kita buat, tidak bergantung pada nama file yang dibuat.



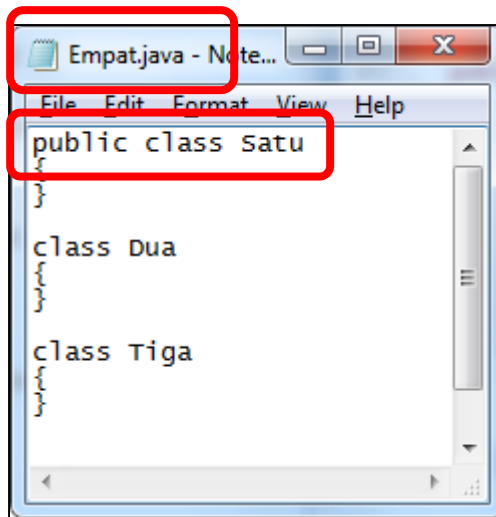
```
File Edit Format View Help
class Satu
}
class Dua
}
class Tiga
}
```

 Dua.class	11/8/2016 2:15 PM	CLASS File	1 KB
 Satu.class	11/8/2016 2:15 PM	CLASS File	1 KB
 Tiga.class	11/8/2016 2:15 PM	CLASS File	1 KB
 Empat.java	11/8/2016 2:14 PM	JAVA File	1 KB

- Contoh berikut menunjukkan bahwa file .class yang terbentuk hanya, Satu.class, Dua.class, Tiga.class.

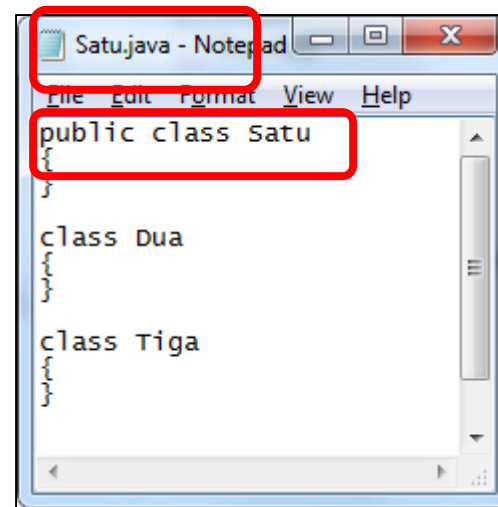
Class dengan label “public”

- Berikut contoh program yang error, karena nama file tidak sama dengan class yg “public” . Harusnya nama file yang benar adalah “Satu.java”, bukan “Empat.java”



```
Empat.java - Note...
File Edit Format View Help
public class satu
{
}
class Dua
{
}
class Tiga
{
}
```

Salah, nama tidak sama,



```
Satu.java - Notepad
File Edit Format View Help
public class satu
{
}
class Dua
{
}
class Tiga
{
}
```

Benar, nama sama

Static Keyword

- Dapat di apply ke atribut ataupun method
- Jika di apply di dalam sebuah atribut(variabel), variable menjadi class variabel
- Jenis variabel didalam class ada 2:
 - Object variabel : variabel tersebut menjadi milik sebuah object. Efeknya, jika variabel tersebut diubah, maka hanya object tersebut yang berubah, yang lain tidak.
 - Class variabel : variabel tersebut menjadi milik kelas. Efeknya, jika variabel tersebut diubah, maka SEMUA OBJECT DARI CLASS TERSEBUT berubah. Atau dapa disebut variabel global.

Object Variabel

Object variabel.
Nilai variable menjadi milik sebuah objek

```
public class Mobil {  
    static int mesin=1;  
    int roda=4;  
    int body=1;  
}
```

```
public class Main2 {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Mobil avanza = new Mobil();  
        Mobil innova = new Mobil();  
  
        System.out.println("roda avanza : "+avanza.roda);  
        System.out.println("roda innova : "+innova.roda);  
        avanza.roda=5;  
        System.out.println("roda avanza : "+avanza.roda);  
        System.out.println("roda innova : "+innova.roda);  
  
        System.out.println("mesin avanza : "+avanza.mesin);  
        System.out.println("mesin innova : "+innova.mesin);  
        avanza.mesin=2;  
        System.out.println("mesin avanza : "+avanza.mesin);  
        System.out.println("mesin innova : "+innova.mesin);  
    }  
}
```

```
roda avanza : 4  
roda innova : 4  
roda avanza : 5  
roda innova : 4  
mesin avanza : 1  
mesin innova : 1  
mesin avanza : 2  
mesin innova : 2
```

Class Variabel

Class variabel.
Nilai variable menjadi
global, milik sebuah class

```
public class Mobil {  
    static int mesin=1;  
    int roda=4;  
    int body=1;  
}
```

```
roda avanza : 4  
roda innova : 4  
roda avanza : 5  
roda innova : 4  
mesin avanza : 1  
mesin innova : 1  
mesin avanza : 2  
mesin innova : 2
```

```
public class Main2 {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Mobil avanza = new Mobil();  
        Mobil innova = new Mobil();  
  
        System.out.println("roda avanza : "+avanza.roda);  
        System.out.println("roda innova : "+innova.roda);  
        avanza.roda=5;  
        System.out.println("roda avanza : "+avanza.roda);  
        System.out.println("roda innova : "+innova.roda);  
  
        System.out.println("mesin avanza : "+avanza.mesin);  
        System.out.println("mesin innova : "+innova.mesin);  
        avanza.mesin=2;  
        System.out.println("mesin avanza : "+avanza.mesin);  
        System.out.println("mesin innova : "+innova.mesin);  
    }  
}
```

Class Variabel

- Class variabel dapat langsung diakses dengan nama kelasnya, tanpa membentuk objek.

```
public class Mobil {  
    static int mesin=1;  
    int roda=4;  
    int body=1;  
}
```

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    Mobil avanza = new Mobil();  
    Mobil innova = new Mobil();  
  
    avanza.body=2;  
    innova.roda=6;  
    Mobil.mesin=2;  
}
```

Analogi Object dan Class Variabel

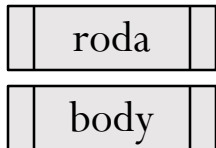
```
public class Mobil {  
    static int mesin=1;  
    int roda=4;  
    int body=1;  
}
```

```
public class Main2 {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Mobil avanza = new Mobil();  
        Mobil innova = new Mobil();  
  
        System.out.println("roda avanza : "+avanza.roda);  
        System.out.println("roda innova : "+innova.roda);  
        avanza.roda=5;  
        System.out.println("roda avanza : "+avanza.roda);  
        System.out.println("roda innova : "+innova.roda);  
  
        System.out.println("mesin avanza : "+avanza.mesin);  
        System.out.println("mesin innova : "+innova.mesin);  
        avanza.mesin=2;  
        System.out.println("mesin avanza : "+avanza.mesin);  
        System.out.println("mesin innova : "+innova.mesin);  
    }  
}
```

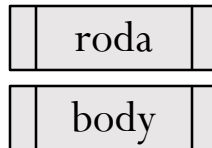
Class / Global / Static Variable



Avanza



Innova



Static/ Class Method

- Jika static di apply di dalam sebuah method, maka akan menjadi class method, bukan object method.
- Artinya method tersebut dapat dipanggil tanpa menggunakan object, langsung menggunakan class
- Aturan static variabel maupun static method:
 - Static memanggil static (bisa)
 - Static memanggil non-static (tidak bisa)
 - Non-static memanggil static (bisa)
 - Non-static memanggil non-static (bisa)

Class Method

```
public class Mobil {  
    static int mesin=1;  
    int roda=4;  
    int body=1;  
  
    static void maju(){  
        System.out.println("maju");  
    }  
  
    void mundur(){  
        System.out.println("mundur");  
    }  
  
    void belok(){  
        System.out.println("belok");  
    }  
}
```

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    Mobil avanza = new Mobil();  
    Mobil innova = new Mobil();  
  
    Mobil.maju();  
}
```

Method static `maju()`,
dapat langsung diakses
dengan class `Mobil`
“`Mobil.maju();`”

Constructor

- Merupakan block kode mirip sub program/method, dg ciri:
 - Nama block persis seperti nama class
 - Dipanggil otomatis ketika object di create dengan operator “new”
 - Digunakan untuk menginisialisasi field
 - Tanpa ada label “void” /return type

Contoh Constructor

```
public class Mobil {  
    int mesin=1;  
    int roda=4;  
    int body=1;  
  
    Mobil(){  
        System.out.println("sebuah object mobil terbentuk di memori...");  
    }  
  
    void maju(){  
        System.out.println("maju");  
    }  
}
```

Konstruktor

```
public class Main2 {  
  
    public static void main(String[] args) {  
        Mobil avanza = new Mobil();  
        Mobil innova = new Mobil();  
    }  
}
```

Pemanggilan
konstruktor saat
pembentukan objek

```
sebuah object mobil terbentuk di memori...  
sebuah object mobil terbentuk di memori...
```

Hasil eksekusi

Constructor Dengan Parameter

```
public class Mobil {  
    int mesin=1;  
    int roda=4;  
    int body=1;  
  
    Mobil(){  
        System.out.println("sebuah object mobil terbentuk di memori...");  
    }  
  
    Mobil(int a, int b, int c){  
        mesin = a;  
        roda = b;  
        body = c;  
    }  
}
```











Konstruktur dengan parameter

```
public class Main2 {  
  
    public static void main(String[] args) {  
        Mobil avanza = new Mobil();  
        Mobil innova = new Mobil(3, 4, 5);  
    }  
}
```

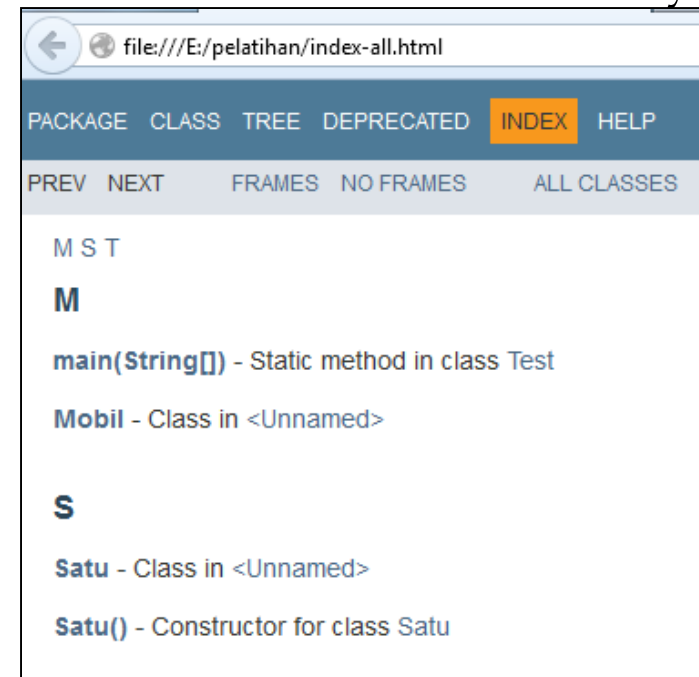
Pemanggilan konstruktur dengan parameter saat pembentukan objek

Java Documentation (1)

- Menggunakan Command Prompt
 - Gunakan perintah “javadoc namaclass.java”
 - Contoh “javadoc Mobil.class”
 - Jika berhasil akan ada beberapa file .html dan .css di folder yang sama. Pilih index-all.html.

	allclasses-frame	11/9/2016 9:06 AM
	allclasses-noframe	11/9/2016 9:06 AM
	constant-values	11/9/2016 9:06 AM
	deprecated-list	11/9/2016 9:06 AM
	help-doc	11/9/2016 9:06 AM
	index	11/9/2016 9:06 AM
	index-all	11/9/2016 9:06 AM
	Mobil	11/9/2016 9:06 AM
	overview-tree	11/9/2016 9:06 AM
	package-frame	11/9/2016 9:06 AM

Type: Firefox HTML Document
Size: 4.85 KB
Date modified: 11/9/2016 9:06 AM



file:///E:/pelatihan/index-all.html

PACKAGE CLASS TREE DEPRECATED **INDEX** HELP

PREV NEXT FRAMES NO FRAMES ALL CLASSES

M S T

M

main(String[]) - Static method in class Test

Mobil - Class in <Unnamed>

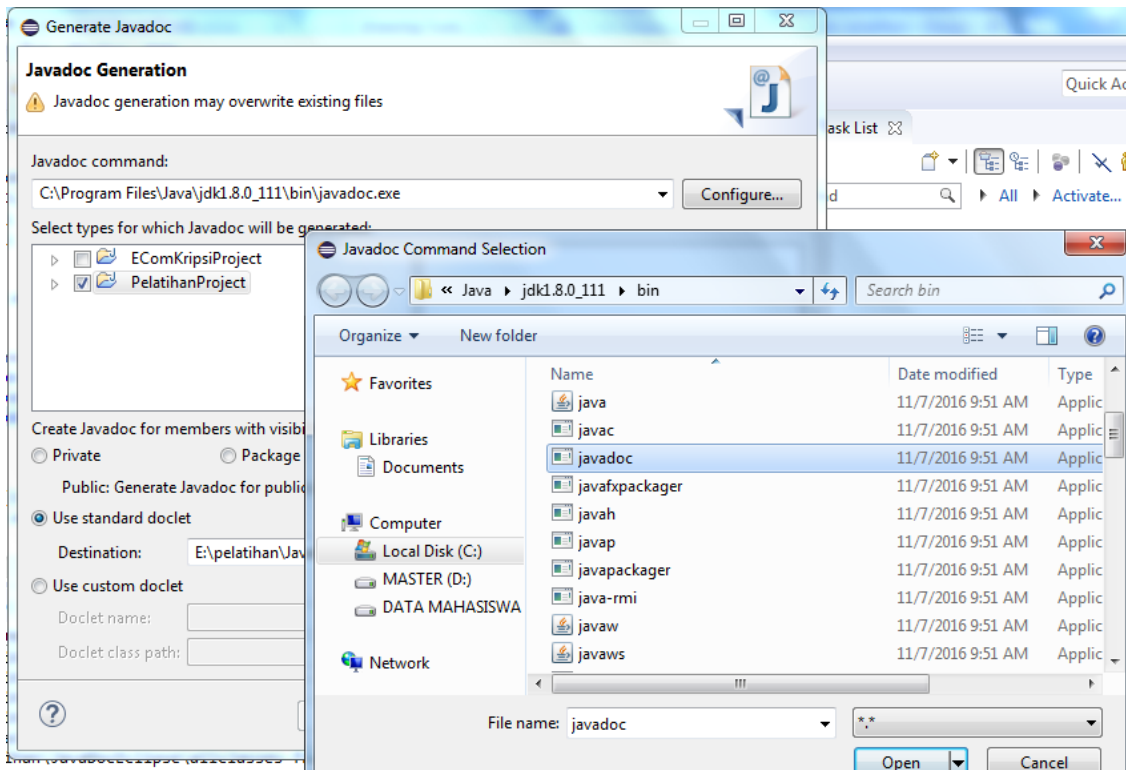
S

Satu - Class in <Unnamed>

Satu() - Constructor for class Satu

Java Documentation (2)

- Menggunakan Eclipse
 - Pilih Project- → Generate Javadoc
 - Pilih configure → browse javadoc.exe pada hasil instalasi jdk → bin



Java Documentation (2)

- Tidak semua variabel, method dan constructor tampil pada hasil javadoc, karena hak akses tidak “PUBLIC”
- Jika telah dipublic, dokumentasi akan lebih lengkap.

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
void	belok()	
void	maju()	
void	mundur()	
Methods inherited from class java.lang.Object		
equals, getClass, hashCode, notify, notifyAll, toString		
Constructor Detail		
Mobil		
public Mobil()		
Mobil		
public Mobil(int a, int b,		

Method Detail

maju

```
public void maju()
```

mundur

```
public void mundur()
```

belok

```
public void belok()
```

“This” Keyword

- Pada konstruktor yang dibuat, ada variabel a, b dan c yang tidak menggambarkan mewakili variabel apa saja.
- Untuk memudahkan variabel a, b, c diubah sesuai dengan nama variabel.

Untuk membedakan antara mesin, roda, body parameter dengan mesin, roda, body milik kelas, digunakan kata kunci **this**.

```
Constructor Detail  
Mobil  
public Mobil()  
  
Mobil  
public Mobil(int a,  
             int b,  
             int c)
```

```
Mobil(int a, int b, int c){  
    mesin = a;  
    roda = b;  
    body = c;  
}
```

```
Constructor Detail  
Mobil  
public Mobil()  
  
Mobil  
public Mobil(int mesin,  
             int roda,  
             int body)
```

```
public Mobil(int mesin, int roda, int body){  
    this.mesin = mesin;  
    this.roda = roda;  
    this.body = body;  
}
```

“This” Keyword

- This artinya yaitu This Class
- Digunakan untuk mengakses field / method milik class
- This bisa juga digunakan untuk memanggil 1 constructor dari constructor lain, syaratnya:
 - Hanya dapat dilakukan dari constructor lain
 - Harus dibaris paling atas
 - Hanya bisa satu kali

HAS-A

- Object Mobil, memiliki Pintu (Mobil HAS-A Pintu)
- Object Pintu, memiliki Jendela (Pintu HAS-A Jendela)
- Maka dapat disimpulkan kita membutuhkan 3 Class untuk Jendela, Pintu dan Mobil.
- Lalu identifikasi atribut dan method yang dapat dilakukan tiap Class

Jendela, Pintu, dan Mobil

```
class Window {
    int kaca;

    void bukaKaca(){
        System.out.println("buka kaca...");
    }

    void tutupKaca(){
        System.out.println("tutup kaca...");
    }
}
```

```
class Door {
    Window jendela;
    Door(Window jendela){
        this.jendela=jendela;
    }
    void bukaJendela(){
        System.out.println("buka jendela");
        jendela.bukaKaca();
    }

    void tutupJendela(){
        System.out.println("tutup jendela");
        jendela.tutupKaca();
    }
}
```

```
public class Car {

    Door pintu;
    Car(Door pintu){
        this.pintu=pintu;
    }

    public void bukaPintu(){
        System.out.println("buka pintu");
    }

    public void bukaJendela(){
        pintu.bukaJendela();
    }
}
```

```
public class Main3 {

    public static void main(String[] args) {
        Window jendela = new Window();
        Door pintu = new Door(jendela);
        Car bmw = new Car(pintu);

        bmw.bukaPintu();
        bmw.bukaJendela();
    }
}
```

Jendela, Pintu, dan Mobil (2)

- Untuk membuat tiap object yang dibuat tidak perlu di ciptakan di Main, maka di Konstruktor tiap kelas langsung diinstansiasi objectnya

```
class Window {
    int kaca;

    void bukaKaca(){
        System.out.println("buka kaca...");
    }

    void tutupKaca(){
        System.out.println("tutup kaca...");
    }
}
```

```
class Door {
    Window jendela;
    Door(){
        jendela = new Window();
    }
    void bukaJendela(){
        System.out.println("buka jendela");
        jendela.bukaKaca();
    }

    void tutupJendela(){
        System.out.println("tutup jendela");
        jendela.tutupKaca();
    }
}
```

```
public class Car {
    Door pintu;
    Car(){
        pintu = new Door();
    }

    public void bukaPintu(){
        System.out.println("buka pintu");
    }

    public void bukaJendela(){
        pintu.bukaJendela();
    }
}
```

```
public class Main3 {
    public static void main(String[] args) {
        Car bmw = new Car();

        bmw.bukaPintu();
        bmw.bukaJendela();
    }
}
```

Reference

- Pemaparan materi TOT Java Fundamental oleh bapak Tri Haryoko (7-11 Nopember 2016, Bandar Lampung)
- <https://docs.oracle.com/javase/tutorial/java>
- “Thinking in Java”, Bruce Eckel