

# Pertemuan 8 : Overloading dan Overriding



Some lecture slides obtained from *Pemrograman Berbasis Obyek* by Ali Ridho Barakbah, SKom.



# Pokok Bahasan

- Overloading
- Overriding
- Aturan tentang Overriden method



# Review : Constructor

- Suatu obyek dapat diberi nilai awal atau diinisialisasi dengan menggunakan constructor.
- Jadi constructor dijalankan hanya sekali yaitu saat suatu obyek diciptakan.
- Contoh : memberi nilai awal 1-Mei-2007 pada obyek yang diciptakan dari class Tanggal

```
Tanggal.java
public class Tanggal {
    public int tgl;
    public int bulan;
    public int tahun;
    public Tanggal() {
        tgl=1;
        bulan=5;
        tahun=2007;
    }
}
```



# Ciri Constructor

- Ciri constructor :
  - Mempunyai nama yang sama dengan nama class
  - Tidak mempunyai nilai kembalian (return type)



# Overloading pada Constructor

- Suatu class boleh memiliki lebih dari satu constructor
- Constructor yang terdiri dari dua atau lebih disebut dengan overloading constructor
- Constructor yang terdiri dari dua atau lebih disebut dengan overloading constructor
- Overloading constructor termasuk salah satu dari ciri program OOP yaitu polimorfisme
- Polimorfisme => berbeda-beda bentuk;



## Syarat pada Overloading Constructor

- Syarat pada overloading constructor adalah signature dari constructor-constructor tersebut tidak sama.
- Signature adalah informasi untuk membedakan method seperti nama method, jumlah parameter, tipe data, dan tipe kembalian (return type).



## Percobaan 3

- Contoh Overloading constructor :

```
class Titik3D {
    double x;
    double y;
    double z;
    Titik3D (double ax) {
        x = ax;
        y = 1;
        z = 1;
    }
    Titik3D (double ax, double zy) {
        x = ax;
        y = ay;
        z = 1;
    }
    Titik3D (double ax, double zy, double az) {
        x = ax;
        y = ay;
        z = az;
    }
}
```

```
class ContohTitik3D {
    public static void main(String args[]) {
        Titik3D p = new Titik3D(1.1);
        System.out.println("p.x = " + p.x);
        System.out.println("p.y = " + p.y);
        System.out.println("p.z = " + p.z);
        Titik3D p = new Titik3D(1.1, 3.4);
        System.out.println("p.x = " + p.x);
        System.out.println("p.y = " + p.y);
        System.out.println("p.z = " + p.z);
        Titik3D p = new Titik3D(1.1, 3.4, -2.8);
        System.out.println("p.x = " + p.x);
        System.out.println("p.y = " + p.y);
        System.out.println("p.z = " + p.z);
    }
}
```







# Overloading

- Overloading adalah suatu keadaan dimana beberapa method memiliki nama yang sama tetapi fungsionalitasnya berbeda
- Contoh :
  - titik(x,y);
  - titik(x,y,z);
- Ciri Overloading :
  - Nama method harus sama
  - Daftar parameter harus berbeda
  - Return type boleh sama, boleh berbeda



# Overriding

- Overriding menyatakan suatu keadaan dimana method pada subclass menolak method pada parent class-nya.
- Ciri dari overriding :
  - Nama method harus sama
  - Daftar parameter harus sama
  - Return type harus sama



# Contoh overriding

```
class Parent {  
    public void Info() {  
        System.out.println("ini class parent");  
    }  
class Child extends Parent {  
    public void Info() {  
        System.out.println("ini class child");  
    }  
}
```



## Peraturan pada Overriding

- Method yang terkena Override (overriden method) tidak boleh mempunyai modifier yang lebih luas aksesnya daripada method yang meng-override (overriding method).

# Percobaan





# Melakukan Overloading pada Method

```
import java.awt.Point;
public class Segiempat {
    int x1 = 0;
    int y1 = 0;
    int x2 = 0;
    int y2 = 0;
    public void buatSegiempat(int x1, int y1, int x2, int y2) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
    }
    public void buatSegiempat(Point topLeft, Point bottomRight) {
        x1 = topLeft.x;
        y1 = topLeft.y;
        x2 = bottomRight.x;
        y2 = bottomRight.y;
    }
    public void buatSegiempat(Point topLeft, int w, int h) {
        x1 = topLeft.x;
        y1 = topLeft.y;
        x2 = (x1 + w);
        y2 = (y1 + h);
    }
}
```



# Melakukan Overloading pada Method

```
void cetakSegiempat(){
    System.out.print("Segiempat: <" + x1 + ", " + y1);
    System.out.println(", " + x2 + ", " + y2 + ">");
}
public static void main(String[] arguments) {
    Segiempat rect = new Segiempat();
    System.out.println("Buat segiempat dengan koordinat (25,25)
    dan (50,50)");
    rect.buatSegiempat(25, 25, 50, 50);
    rect.cetakSegiempat();
    System.out.println();
    System.out.println("Buat segiempat dengan point (10,10) dan
    point (20,20):");
    rect.buatSegiempat(new Point(10,10), new Point(20,20));
    rect.cetakSegiempat();
    System.out.println();
    System.out.print("Buat segiempat dengan 1 point (10,10),
    koodinat (50,50)");
    rect.buatSegiempat(new Point(10,10), 50, 50);
    rect.cetakSegiempat();
}
}
```

# Latihan

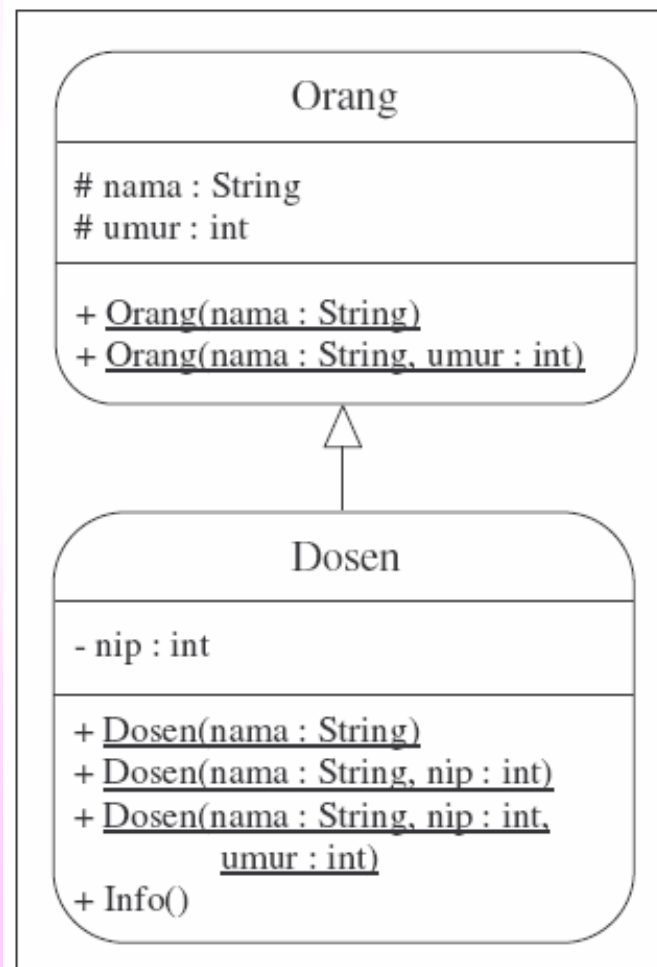






# Mengimplementasikan UML Class Diagram pada Program

- Implementasikan UML Class diagram berikut :





# Menguji UML Class Diagram pada Program

```
public class TesLatihan {  
    public static void main(String args[]) {  
        System.out.println("Masukkan identitas dosen 1 : Agus");  
        Dosen dosen1=new Dosen("Agus");  
        System.out.println("Masukkan identitas dosen 2 : Budi, NIP. 1458");  
        Dosen dosen2=new Dosen("Budi", 1458);  
        System.out.println("Masukkan identitas dosen 3 : Iwan,NIP. 1215, umur 47");  
        Dosen dosen3=new Dosen("Iwan", 1215, 47);  
        System.out.println();  
        dosen1.Info();  
        System.out.println();  
        dosen2.Info();  
        System.out.println();  
        dosen3.Info();  
    }  
}
```