

Modul 5: Constructor, Overloading, inheritance, Polymorphism

Setelah mengikuti mata kuliah ini mahasiswa dapat mendefinisikan constructor, menerapkan konsep *overloading* dan konsep *inheritance* dan *polymorphism*

Pengantar :

- Dalam modul ini akan diuraikan beberapa topik bahasan yaitu :
 - 1) Pengertian constructor
 - 2) Cara mengoverloading constructor
 - 3) Melakukan inheritance, mendefinisikan subclass

1. Constructor

- **Constructor** adalah metod dari suatu class yang memiliki karakteristik:
 - otomatis akan dieksekusi jika suatu objek class tersebut diciptakan
 - didefinisikan dengan modifier public
 - sama dengan nama class
 - bisa mengandung atau tidak parameter
 - dapat didefinisikan lebih dari satu macam definisi (biasanya ini yang terjadi)
- **Jika Constructor tidak didefinisikan :**
 - saat objek dibuat hanya memiliki satu pilihan membuat objek , yaitu :
`namaClass namaObjek= new namaClass();`

Untuk ases selanjutnya terhadap data dan metod digunakan teknik dalam pembatasan (modifier) dari data dan metodnya.

Contoh class tanpa constructor

Contoh class Data didefinisikan tanpa constructor

```
public class Tes
{ public static void main(String args[])
  { Data data1=new Data();
    Data data2=new Data();
    data1.cetak();
    data2.cetak();
  }
}
class Data
{ int P=88; int Q=77;
  public void cetak()
  {System.out.println("P="+P+" dan Q="+Q);}
}
```

Hasil program



```
C:\j2\bin>java Tes
P=88 dan Q=77
P=88 dan Q=77
```

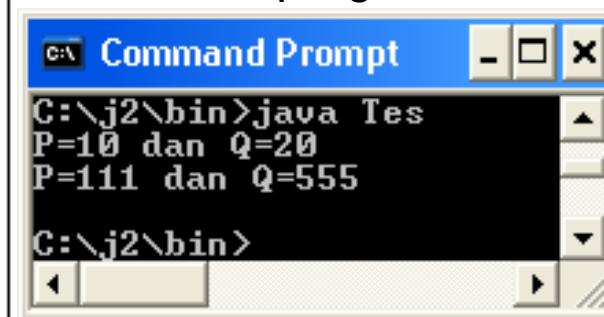
Pertanyaannya : Jika objek dibuat dan sekaligus akan merubah/memberi nilai baru bagi P dan Q caranya bagaimana?
Itulah diperlukannya constructor !!

Class dengan 1 constructor saja

1 constructor : mengisi P dan Q

```
public class Tes
{ public static void main(String args[])
  { Data data1=new Data(10,20);
    Data data2=new Data(111,555);
    data1.cetak();
    data2.cetak();
  }
}
class Data
{ private int P=88; private int Q=77;
  public Data(int p,int q){P=p; Q=q;}
  public void cetak()
  {System.out.println("P="+P+" dan Q="+Q);}
}
```

Hasil program



```
C:\j2\bin>java Tes
P=10 dan Q=20
P=111 dan Q=555
C:\j2\bin>
```

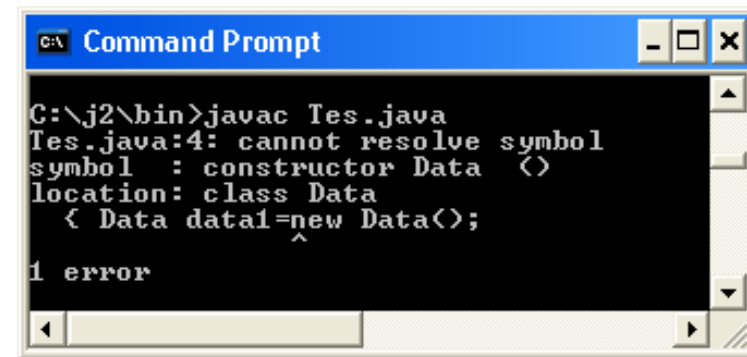
Pertanyaannya :

1. Bagaimana jika kita tidak akan mengubah nilai P dan Q, tetapi menggunakan nilai P=88 dan Q=77 ?
2. Bagaimana pula jika yang akan kita ubah Cuma nilai P saja atau Q saja

Membuat objek tanpa parameter dengan constructor berparameter

- Berikut ini membuat objek data dengan tanpa melewati nilai parameter

```
public class Tes
{ public static void main(String args[])
  { Data data1=new Data();
    data1.cetak();
  }
}
```



```
C:\j2\bin>javac Tes.java
Tes.java:4: cannot resolve symbol
symbol  : constructor Data ()
location: class Data
  < Data data1=new Data();
                    ^
1 error
```

Ternyata ketika telah didefinisikan constructor dengan parameter, kita tidak dapat lagi membuat objek tanpa parameter. Program gagal dicompile.

Solusi dari masalah ini adalah dengan meng-**overloading** constructor

2. Overloading

- **Overloading** adalah mendefinisikan lebih dari satu constructor, dengan parameter yang berbeda-beda sehingga pada saat membuat objek maka akan dapat dipilih constructor yang mana yang akan digunakan.
- Salah satu constructor itu biasanya disebut constructor default (biasanya didefinisikan tanpa parameter)
- Contoh :

Untuk class Data pada program sebelumnya, jika didefinisikan constructor default :

```
public Data(){ };
```

maka ketika membuat objek dengan :

```
Data data=new Data(); tidak akan ada masalah
```

Construktur default digunakan

- Konstruktur default
`public Data(){ }`

```
public class Tes
{ public static void main(String args[])
  { Data data1=new Data();
    data1.cetak();
  }
}
class Data
{ private int P=88; private int Q=77;
  public Data(){ }
  public Data(int p,int q){P=p; Q=q;}
  public void cetak()
  {System.out.println("P="+P+" dan Q="+Q);}
}
```

Hasil program



```
C:\> Command Prompt
C:\j2\bin>javac Tes.java
C:\j2\bin>java Tes
P=88 dan Q=77
C:\j2\bin>
```

Jika constructor default digunakan maka sebagai akibatnya data yang didefinisikan secara default juga akan digunakan.

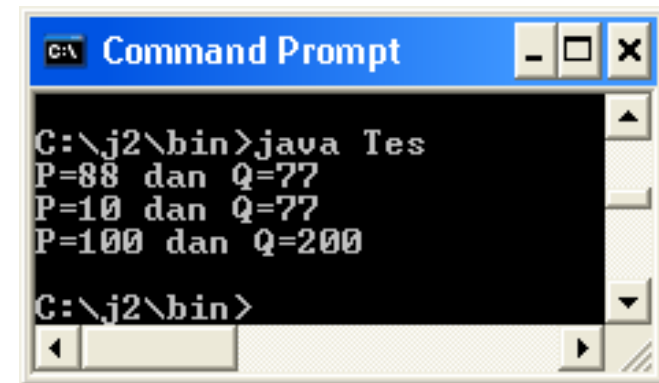
Nilai P=88 dan Q=77 adalah nilai default yang digunakan seperti nampak dalam output program

Banyak constructor

Membuat objek dengan 3 cara

```
public class Tes
{
    public static void main(String args[])
    {
        Data data1=new Data();
        Data data2=new Data(10);
        Data data3=new Data(100,200);
        data1.cetak();
        data2.cetak();
        data3.cetak();
    }
}
class Data
{
    private int P=88; private int Q=77;
    public Data(){ }
    public Data(int p){P=p;}
    public Data(int p,int q){P=p; Q=q;}
    public void cetak()
    {System.out.println("P="+P+" dan Q="+Q);}
}
```

Hasil program



```
C:\> Command Prompt
C:\j2\bin>java Tes
P=88 dan Q=77
P=10 dan Q=77
P=100 dan Q=200
C:\j2\bin>
```

Pada data1 : P, Q default

Pada data2 : Q default

Ternyata program dapat memilih constructor yang mana harus digunakan dengan cara melihat parameter yang dilewatkan/tidak dilewatkan

Contoh class Anjing

```
class Anjing
{ private String Nama="NoName"; // data default
  private int NoAnjing;
  private static int Urut=0;

  // constructor tanpa paramater = default
  public Anjing()
  {NoAnjing=++Urut;};

  // constructor dengan 1 parameter
  public Anjing(String n)
  {this.Nama=n;
   this.NoAnjing=++Urut;};

  // constructor dengan 2 parameter
  public Anjing(String n, int u)
  {this.Nama=n;
   this.NoAnjing=u;};

  public void Gonggong(String S)
  { System.out.println(S+"! " + S + "!! " + S + " !!!");
  }

  public void SayHello()
  { System.out.println("Hello Saya Anjing No:" +NoAnjing+ " Nama saya : "+Nama);
    System.out.println(" ");
  }
}
```

Operator **this** berarti mangacu pada objek yang sedang digunakan

Program utama yang menguji

```
public class TesAnjing
{ public static void main(String args[])
  { // memanggil constructor default
    Anjing AnjingKU=new Anjing();
    Anjing AnjingMU=new Anjing();

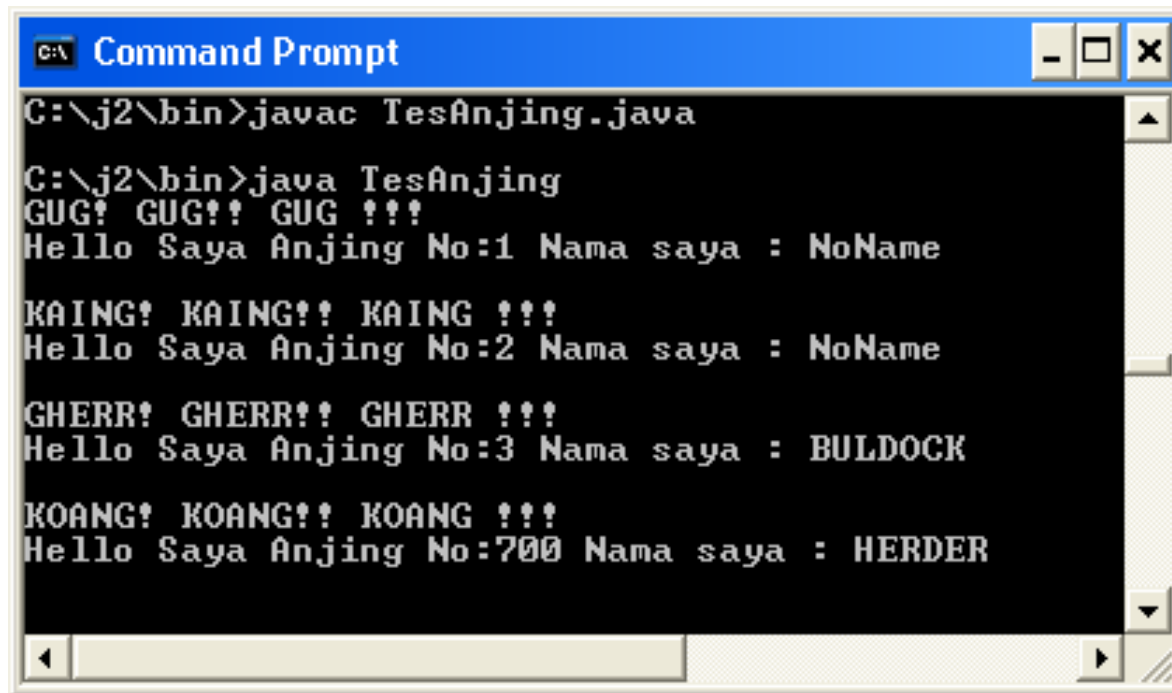
    AnjingKU.Gonggong("GUG");
    AnjingKU.SayHello();

    AnjingMU.Gonggong("KAING");
    AnjingMU.SayHello();

    // memanggil constructor dengan 1 parameter
    Anjing AnjingNYA=new Anjing("BULDOCK");
    AnjingNYA.Gonggong("GHERR");
    AnjingNYA.SayHello();

    // memanggil constructor dengan 2 parameter
    Anjing AnjingKITA=new Anjing("HERDER",700);
    AnjingKITA.Gonggong("KOANG");
    AnjingKITA.SayHello();
  }
}
```

Hasil programnya ...



```
C:\j2\bin>javac TesAnjing.java
C:\j2\bin>java TesAnjing
GUG! GUG!! GUG !!!
Hello Saya Anjing No:1 Nama saya : NoName

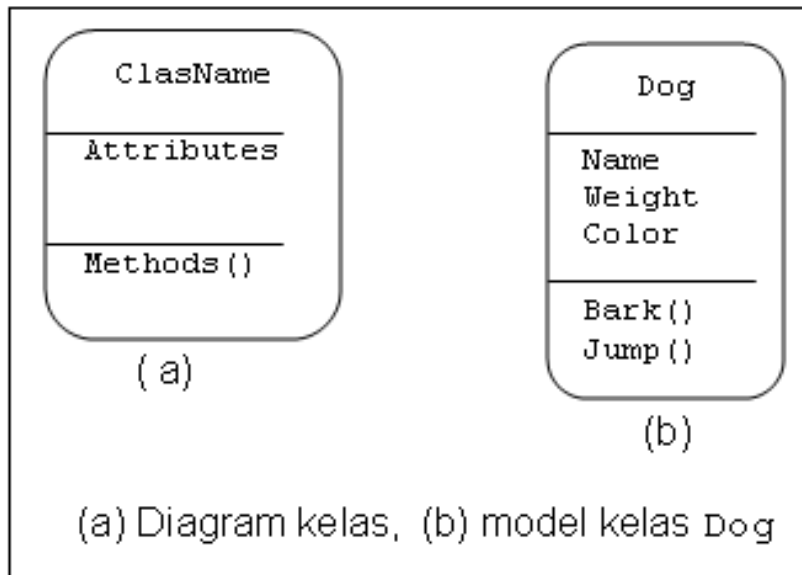
KAING! KAING!! KAING !!!
Hello Saya Anjing No:2 Nama saya : NoName

GHERR! GHERR!! GHERR !!!
Hello Saya Anjing No:3 Nama saya : BULDOCK

KOANG! KOANG!! KOANG !!!
Hello Saya Anjing No:700 Nama saya : HERDER
```

3. Inheritance

- Introduction ke dalam inheritance dimulai dari
- Struktur sebuah kelas dapat digambarkan dalam sekema



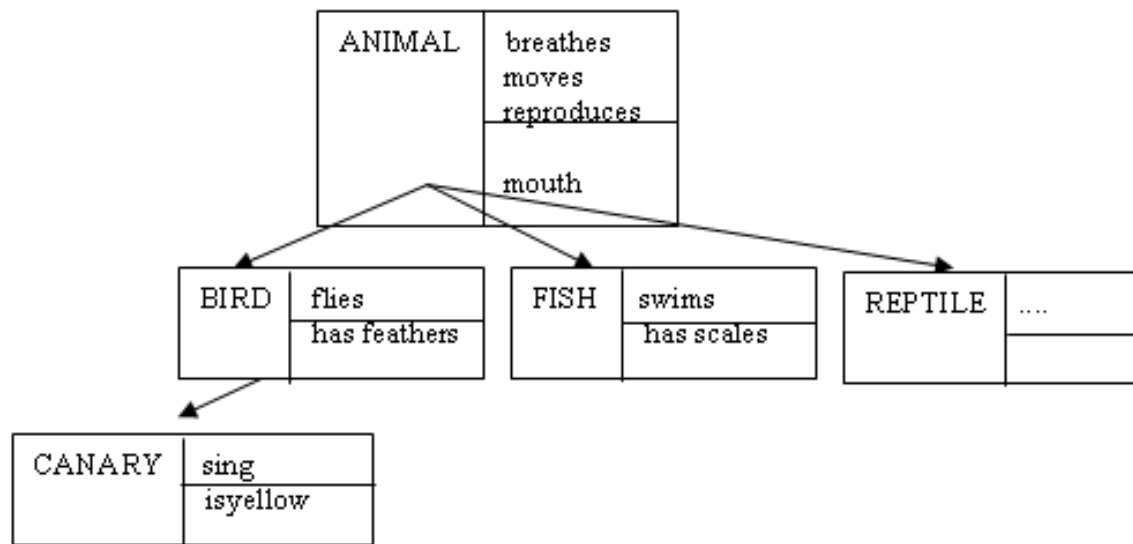
Class memiliki 2 hal ,
yaitu :

Data (Attribut)

Metod

Analogi konsep inheritance

- Secara konsep Inheritance adalah pewarisan sifat induk. Seperti dalam gambaran taksonomi berikut



Pada gambar tersebut class CANARY dapat mewakili data dan metod dari BIRD dan ANIMAL

Sesuai kaidah inheritance

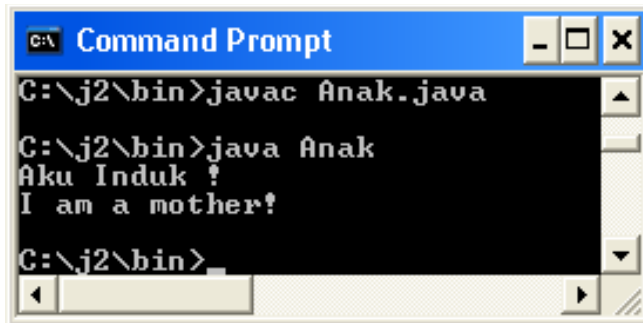
Konsep Inheritance ...

- **Inheritance** adalah konsep penurunan data atau metod (biasanya metod) oleh class yang merupakan class turunan.
- Cara mendefinisikan class turunan memiliki sintak:
class namaClassAnak **extends**
namaClassInduk
- Dengan deklarasi seperi itu maka metod-metod dalam class induk yang bersifat **public** dan **protected** dapat di"warisi" oleh class anakl

Contoh Inheritance

```
public class Anak extends Induk
{public static void main(String args[])
  {Anak a=new Anak();
   a.cetak1();
   a.cetak2();
  }
}

class Induk
{ public void cetak1(){System.out.println("Aku Induk !"); }
  protected void cetak2(){System.out.println("I am a mother!");}
  private void cetak3(){System.out.println("P= "+P);}
  int P=8;
}
```



```
C:\j2\bin>javac Anak.java
C:\j2\bin>java Anak
Aku Induk !
I am a mother!
C:\j2\bin>
```

Terlihat bahwa sebagai objek dari class Anak, objek a dapat memanggil metod dari class induk, yaitu cetak1() dan cetak2()

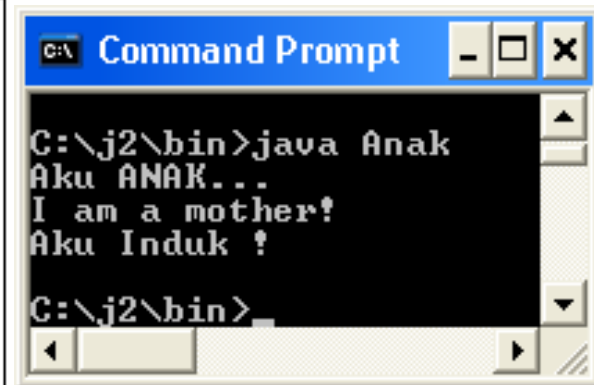
Polymorphism

- **Polymorphism** adalah konsep yang memungkinkan mendefinisikan metod pada class anak (**sub class**) yang memiliki definisi sama dengan metod induk (**super class**)
- Pendefinisian kembali metod dalam class anak dengan nama yang sama dari metod class induk sering disebut melakukan **OverRiding** terhadap metod

Contoh Polymorphism (mengoverride metod cetak1())

```
public class Anak extends Induk
{ public void cetak1(){System.out.println("Aku ANAK..."); }
  public static void main(String args[])
  {Induk d=new Induk();
   Anak a1=new Anak();
   a1.cetak1();
   a1.cetak2();
   d.cetak1();
  }
}

class Induk
{ public void cetak1(){System.out.println("Aku Induk !"); }
  protected void cetak2(){System.out.println("I am a mother!");}
  private void cetak3(){System.out.println("P= "+P);}
  int P=8;
}
```



```
C:\j2\bin>java Anak
Aku ANAK...
I am a mother!
Aku Induk !
C:\j2\bin>
```

Hasil program

Hasil polymorphism

- Metod cetak1() dapat diwariskan
- Metod cetak2() dapat diwariskan
- Metod cetak3() tidak dapat diwariskan

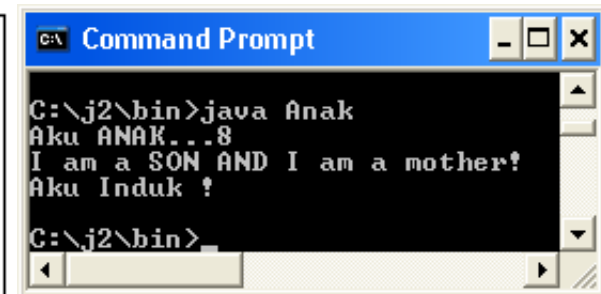
a1.cetak1() : akan gunakan definisi dari class Anak

d.cetak1() : akan gunakan definisi dari induk

Memfaatkan data dan metod induk dengan kata kunci : `super`

- Misalkan dengan tidak merubah definisi pada class Induk, kita lakukan perubahan definisi pada class Anak.
- Untuk akses data induk digunakan `super>NamaData`, dan untuk menggunakan metod induk digunakan `super>NamaMetod`

```
public class Anak extends Induk
{ public void cetak1()
    {System.out.println("Aku ANAK..." +super.P); }
  public void cetak2()
    {System.out.print("I am a SON AND ");
     super.cetak2();
    }
  public static void main(String args[])
  {Induk d=new Induk();
   Anak a1=new Anak();
   a1.cetak1();
   a1.cetak2();
   d.cetak1();
  }
}
```



```
C:\j2\bin>java Anak
Aku ANAK...8
I am a SON AND I am a mother!
Aku Induk ?
C:\j2\bin>
```

Dengan `super.P` class anak dapat mengakses data induk

Dengan `super.cetak2()` metod induk dapat dijalankan dari Anak

Rangkuman

- Constructor adalah metod khusus dengan nama sama dengan nama class dan bertugas membuat objek
- Constructor dapat di**overloading** dengan mendefinisikan beberapa nama constructor
- **Inheritance** merupakan kemampuan java sebagai OOP untuk mewariskan data dan metod dari Induk(superclass) kepada anak (subclass)
- Jika metod dalam class induk didefinisikan ulang oleh class anak maka pendefinisian disebut melakukan **overriding** metod
- Kemampuan java memiliki nama-nama metod yang sama yang melintas pada garis keturuanan disebut **polymorphism**

Latihan

1. Buatlah definisi class Motor, yang memiliki data : nama (String) default "NoName", dan status (boolean) default= false
memiliki metod : tampilkan() : mencetak keterangan nama keadaan motor (status mati /hidup)
memiliki metod : nyalakan() :berfungsi men-set status dari mati (false) menjadi hidup (true). Jika status sudah hidup dan dipanggil metod nyalakan(), maka akan ada komentar "Mesin sudah hidup", jika mesin belum hidup dan meetod nyalakan() dipanggil maka status diubah menjadi true.
Definisikan Constructor dengan tiga macam
Motor()
Motor (boolean status) untujk mendefinisikan status
Motor (String nama) untuk mendefinisikan nama motor

Latihan ... Ijt

2. Buatlah definisi class TesMotor , untuk membuat objek motor :
objek :motorku , nama motor HONDA kondisi mati, panggil metod
tampilkan(), nama motor dedinisikan lewat constructor

objek motormu, nama motor YAMAHA , panggil metod nyalakan()
dan tampilkan() , motor saat objek dibuat kondisi hidup (true)
melalui konstruktor

objek motornya, nama motor SUZUKU , panggil metod nyalakan()
dan tampilkan() , gunakan konstruktor default

3 ... lanjutan

Lanjutan..

3. Definisikan suatu kelas Lingkaran dengan atribut R dan Luas serta method Cetak() yang menghasilkan output :

LINGKARAN DENGAN JEJARI ... R LUASNYA ADALAH

Definisikan kelas turunan dari kelas Lingkaran yaitu kelas Tabung dengan atribut tambahan TinggiTabung dan VolumeTabung serta *overriding* method Cetak() dengan menghasilkan output :

“TABUNG DENGAN LUAS ALAS ... DAN TINGGI ... VOLUMENYA ADALAH ...

Testlah kemampuan kelas tersebut dengan class TesTab simpan dalam file TesTab.java

Deklarasi objek lingkaran dan objek tabung adalah :

```
Lingkaran L1=new Lingkaran(2); /// lingkaran jejari 2
```

```
Tabung T1 = new tabung(2,4); // tabung Jejari alas 2 tinggi 4
```

Untuk memanggil method Cetak()

```
L1.Cetak()
```

```
T1.Cetak()
```