PEMOGRAMAN JAVA

Yoannita, S.Kom

Overriding Method

Overloading Method

Class(iii) [Superclass and subclass]

[constructor overloading]

Keyword Super

Some materials from Gasim Alkaff

Method overriding

```
class A
{
    void tampil() {
            System.out.println("ini method class A");
    }
}

class B extends A
{
    void tampil() {
            System.out.println("ini method class B");
    }
}

class overriding
{
    public static void main(String args[])
    {
             B orang = new B();
            orang.tampil();
    }
}
```

- Untuk beberapa pertimbangan, terkadang class asal perlu mempunyai implementasi berbeda dari method yang khusus dari *superclass* tersebut.

- Oleh karena itulah, method overriding digunakan. *Subclass* dapat mengesampingkan method yang didefinisikan dalam *superclass* dengan menyediakan implementasi baru dari method tersebut. (subclass dapat meng-override fungsi yang didapat dari superclass)

Method overloading

- Dalam *class* yang kita buat, kadangkala kita menginginkan untuk membuat ***method*** **dengan nama yang sama namun mempunyai fungsi yang berbeda menurut parameter yang digunakan**. Kemampuan ini dimungkinkan dalam pemrograman Java, dan dikenal sebagai *overloading method*.

- *Overloading method* mengijinkan sebuah *method* dengan nama yang sama namun memiliki parameter yang berbeda sehingga mempunyai implementasi dan *return value* yang berbeda pula. Daripada memberikan nama yang berbeda pada setiap pembuatan *method*, *overloading method* dapat digunakan pada operasi yang sama namun berbeda dalam implementasinya.

Method overloading : Contoh

```
public class Bentuk {
    …
    public void Gambar(int t1) {
            …
    }
    public void Gambar(int t1, int t2) {
            …
    }
    public void Gambar(int t1, int t2, int t3) {
            …
    }
    public void Gambar(int t1, int t2, int t3, int t4) {
            …
    }
}
```

Method overloading

return type	nama method	daftar parameter
void	Gambar	(int t1)
void	Gambar	(int t1, int t2)
void	Gambar	(int t1, int t2, int t3)
void	Gambar	(int t1, int t2, int t3, int t4)
↓	↓	↓
sama	sama	berbeda

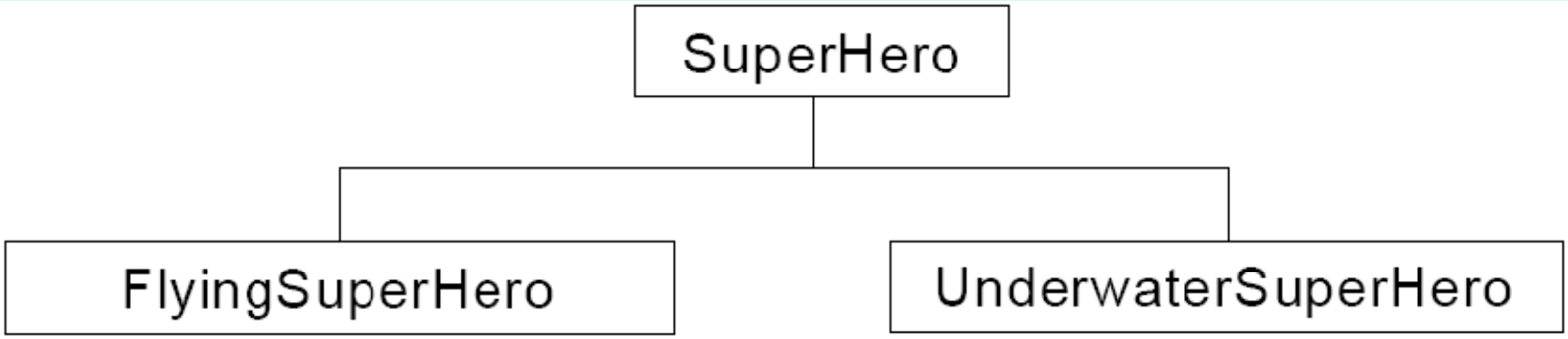
Method overloading

```
class pegawai
{
    void tampil(String nip, String nama, String jabatan) {
      System.out.println("Tampil 3 variabel");
      System.out.println("-----------------");
      System.out.println("NIP     : " + nip);
      System.out.println("Nama    : " + jabatan);
      System.out.println("Jabatan : " + jabatan);
      System.out.println();
    }
    void tampil(String nip, String nama) {
      System.out.println("Tampil 2 variabel");
      System.out.println("-----------------");
      System.out.println("NIP     : " + nip);
      System.out.println("Nama    : " + jabatan);
      System.out.println();
    }
    void tampil(String nip) {
      System.out.println("Tampil 1 variabel");
      System.out.println("-----------------");
      System.out.println("NIP     : " + nip);
      System.out.println();
    }
}
```

```
class overloading {
  public static void main(String args[]) {
            pegawai pgw1, pgw2,pgw3;
            pgw1 = new pegawai();
            pgw2 = new pegawai();
            pgw3 = new pegawai();

            pgw1.tampil("101","Budi","Direktur");
            pgw2.tampil("102");
            pgw3.tampil("103","Desi");
            }
}
```

Pewarisan : [konsep] Superclass dan Subclass



Gambar 1.1: Contoh Pewarisan

- *SuperHero* adalah superclass dari class *FlyingSuperHero* dan *UnderwaterSuperHero*.
- Catatan bahwa *FlyingSuperHero* “is-a” *SuperHero*.
- Sebagaimana juga *UnderwaterSuperHero* “is-a”*SuperHero*

Superclass dan Subclass

```
class orang {
  String nama, alamat;

  orang(String nama, String alamat){
    this.nama = nama;
    this.alamat = alamat;
  }

  orang(){
  }

  void tampil(){
    System.out.println("Nama : " + nama);
    System.out.println("Alamat : " + alamat);
    System.out.println("-----------------------");
   }

}

class dmInherit{
    public static void main(String arg[]){
    pegawai gasim = new pegawai("021013","Gasim AlKaff", "Palembang", "STMIK MDP");
    orang ima = new orang("Fathimah Azzahra", "Palembang");
    gasim.tampil();
    ima.tampil();
  }
}
```

```
class pegawai extends orang{
  String NIP;
  String kantor;
  pegawai(String NIP, String nama, String
alamat,  String kantor){
     this.NIP = NIP;
     this.nama = nama;
     this.alamat=alamat;
     this.kantor = kantor;
   }

  void tampil(){
     System.out.println("NIP : " + NIP);
     System.out.println("Nama : " + nama);
     System.out.println("Alamat : " + alamat);
     System.out.println("Kantor : " + kantor);
   }
}
```

extends merupakan kata kunci yang menyatakan bahwa class ini merupakan *subclass* dari *superclassname.*

Keyword super

- Penggunaan kata kunci *super* berhubungan dengan pewarisan.
- Keyword super digunakan untuk meminta
- constructor superclass (memanggil construktor dan method yang ada pada super-class).
- *Super* juga dapat digunakan seperti kata kunci *this* untuk menunjuk pada anggota dari superclass.

Keyword super

- Ada beberapa hal yang harus diingat ketika menggunakan pemanggil constuktor super:

1. Pemanggil super() HARUS DIJADIKAN PERNYATAAN PERTAMA DALAM constructor.

2. Pemanggil super() hanya dapat digunakan dalam definisi constructor.

3. Termasuk constructor this() dan pemanggil super() TIDAK BOLEH TERJADI DALAM constructor YANG SAMA.

Keyword super

- Pemakaian lain dari super adalah untuk menunjuk anggota dari superclass(seperti reference **this**).
- Sebagai contoh,

```
public class Student extends Person
{
   public Student()          // constructor
   {
      super.name = “somename”;
      super.address = “some address”;
   }
   // some code here
}
```

memanggil variabel name dari class Person

Contoh Keyword **super** dalam Subclass

```
class pegawai extends orang{
  String NIP;
  String kantor;
  pegawai(String NIP, String nama, String alamat,  String kantor){
    super(nama, alamat);
     this.NIP = NIP;
     this.kantor = kantor;
   }

void tampil(){
     System.out.println("NIP : " + NIP);
     System.out.println("Nama : " + nama);
     System.out.println("Alamat : " + alamat);
     System.out.println("Kantor : " + kantor);
  }
}
```

```
class orang {

  String nama, alamat;

  orang(String nama, String alamat){
    this.nama = nama;
    this.alamat = alamat;
  }

  orang(){
  }

}
```

```
class dmInherit{
    public static void main(String arg[]) {
     pegawai gasim = new pegawai("021013","Gasim AlKaff", "Palembang", "STMIK MDP");
     gasim.tampil();
   }
}
```

Contoh Keyword **super** dalam Subclass (ii)

```
public class Superclass {
   public void printMethod() {
      System.out.println("Printed in Superclass.");
   }
}

public class Subclass extends Superclass {
   public void printMethod() {   //overrides printMethod in Superclass
        super.printMethod();
        System.out.println("Printed in Subclass");
   }
   public static void main(String[] args) {
        Subclass s = new Subclass();
        s.printMethod();
   }
}
```

Hasil :
Printed in Superclass.
Printed in Subclass

Keyword ‘super’ dalam Overriding Method (iii)

```
class Y {
	void tampil() {
		System.out.println("ini method class Y");
	}
}

class X extends Y {
	void tampil() {
		super.tampil();
		System.out.println("ini method class X");
	}
}

class SuperDalamOM {
	public static void main(String arg[]) {
		X orang = new X();
		orang.tampil();
	}
}
```

- Hasil program ini adalah.

ini method class Y

ini method class X

Constructor Overloading

- As with methods, constructors can be overloaded.
- Example:

```
public Employee(String name, double salary, Date DoB)
public Employee(String name, double salary)
public Employee(String name, Date DoB)
```

- Argument lists *must* differ.
- You can use the `this` reference at the first line of a constructor to call another constructor.

Constructor Overloading

```
1   public class Employee {
2     private static final double BASE_SALARY = 15000.00;
3     private String name;
4     private double salary;
5     private Date   birthDate;
6
7     public Employee(String name, double salary, Date DoB) {
8       this.name = name;
9       this.salary = salary;
10      this.birthDate = DoB;
11    }
12    public Employee(String name, double salary) {
13      this(name, salary, null);
14    }
15    public Employee(String name, Date DoB) {
16      this(name, BASE_SALARY, DoB);
17    }
18    public Employee(String name) {
19      this(name, BASE_SALARY);
20    }
21    // more Employee code...
22  }
```

Memanggil parent class construktor

```
1   public class Manager extends Employee {
2     private String department;
3
4     public Manager(String name, double salary, String dept) {
5       super(name, salary);
6       department = dept;
7     }
8     public Manager(String n, String dept) {
9       super(name);
10      department = dept;
11    }
12    public Manager(String dept) {
13      department = dept;
14    }
15  }
```