

## Praktikum 6

### Abstract, Interface, Overloading, Overriding, dan Package

Dosen : Ir. Nanang Syahroni M.Kom

#### Pokok Bahasan

- Konsep Abstract, Interface, Overloading, Overriding, dan Package dalam bhs Java
- Deklarasi Abstract, Interface, Overloading, Overriding, dan Package dalam bhs Java
- Penggunaan konsep Abstract, Interface, Overloading, Overriding, dan Package.

#### Tujuan Belajar

- Mengenalkan konsep Abstract, Interface, Overloading, Overriding dan Package pada bahasa pemrograman java
- Mempraktekkan konsep pemrograman berorientasi obyek menggunakan Abstract, Interface, Overloading, Overriding, dan Package.

#### Latar Bekakang

Tabel 1: Kontrol Pengaksesan

Modifier	class yang sama	package yang sama	subclass package lain	class manapun
private	√			
default	√	√		
protected	√	√	√	
public	√	√	√	√

#### A. Abstract

Class Abstrak tidak berbeda dengan class lainnya yaitu memiliki class members (method dan variabel). Sebuah class adalah abstrak jika salah satu methodnya dideklarasikan abstrak. Method abstrak adalah method yang tidak memiliki implementasi.

Contoh deklarasi method abstrak:

```
abstract public void cetak();
```

Beberapa hal yang perlu diperhatikan adalah sebagai berikut:

1. Class abstrak tidak dapat dibuatkan instan atau objeknya menggunakan keyword new.
2. Sebuah class dapat dideklarasikan sebagai class abstrak walaupun tidak memiliki method abstrak.
3. Variabel dengan tipe class abstrak tetap bisa diciptakan, tetapi harus refer ke subclass dari class abstrak tersebut yang tentunya tidak abstrak.

Praktek 1 :

Buatlah program menggunakan abstract seperti pada contoh dibawah:

```
abstract class ClassA {
    int x, y;
    void cetak1(int X, int Y){
    }
    abstract void cetak2();
    abstract void cetak3();
}

class ClassB extends ClassA{
    void cetak2() {
        System.out.println("this is method 2 of class B");
    }
    void cetak3() {
        System.out.println("this is method 3 of class B");
    }
}

class ClassC extends ClassA {
    void cetak2() {
        System.out.println("this is method 2 of class C");
    }
    void cetak3() {
        System.out.println("this is method 3 of class C");
    }
}

public class Latabstract{
    public static void main(String[] args) {
        ClassA obj1= new ClassB();
        obj1.cetak2();
        obj1.cetak3();
        ClassA obj2= new ClassC();
        obj2.cetak2();
        obj2.cetak3();
    }
}
```

## B. Interface

Interface adalah class yang hanya mengandung deklarasi method tanpa memiliki implementasi dan semua property yang dimilikinya bersifat final. Interface mirip dengan class abstrak, tetapi interface tidak terikat dengan class hierarki.

Berikut ini adalah aturan yang harus kita ingat tentang pendeklarasian interface:

1. Modifier yang digunakan hanya public atau tidak sama sekali. Jika tidak menggunakan modifier maka interface tersebut hanya dapat diakses dalam package yang sama.
2. Semua variabel yang dideklarasikan dalam interface secara otomatis adalah static final. Karena itu waktu pendeklarasian harus diberikan nilai.
3. Semua method adalah abstrak. Bedanya dengan class abstrak adalah kita tidak perlu menuliskan keyword abstract pada saat mendeklarasikan method dalam interface.
4. Kita dapat mengimplementasikan lebih dari satu interface (multiple inheritance) dengan memisahkan nama dari setiap interface dengan tanda koma.
5. Dapat terjadi saat kita mengimplementasikan lebih dari satu interface ternyata interface -interface tersebut memiliki method yang sama. Dalam hal ini method yang akan diimplementasi adalah method yang berada pada posisi pertama.
6. Semua method yang diimplemetasikan harus public.
7. Jika kita tidak mengimplementasikan semua method yang ada pada interface, maka class tersebut harus dideklarasikan sebagai abstract class.

Praktek 2 :

Buatlah program menggunakan interface seperti pada contoh dibawah:

```
import java.util.Scanner;

interface Control {
    public void pindahChannel(int channel);
    public void perbesarVolume(int intensitas);
    public void perkecilVolume(int intensitas);
}

class TVPolitron implements Control{
    String[] channel = {"RCTI", "SCTV", "INDOSIAR", "ANTV", "TV7"};
    public void pindahChannel(int channel) {
        System.out.println("TV Politron pindah channel ke "+
            this.channel[channel]);
    }

    public void perbesarVolume(int intensitas) {
        System.out.println("TV Politron perbesar volume ke "+
            intensitas);
    }

    public void perkecilVolume(int intensitas) {
        System.out.println("TV Politron perkecil volume ke "+
            intensitas);
    }
}

class TVSamsung implements Control{
```

```

String[] channel = {"RCTI", "SCTV", "INDOSIAR", "ANTV", "TV7"};

public void pindahChannel(int channel) {
    System.out.println("TV Samsung pindah channel ke "+
this.channel[channel]);
}
public void perbesarVolume(int intensitas) {
    System.out.println("TV Samsung perbesar volume ke "+
intensitas);
}
public void perkecilVolume(int intensitas) {
    System.out.println("TV Samsung perkecil volume ke "+
intensitas);
}
}

class RemoteControl {
public void kirimPerintahKeTv(int aksi,Control tv,int tombol){
    switch(aksi){
        case 1:
            tv.pindahChannel(tombol);
            break;
        case 2:
            tv.perbesarVolume(tombol);
            break;
        case 3:
            tv.perkecilVolume(tombol);
            break;
    }
}
}

public class Latinterface {
public static void main(String[] args){
    TVPolitron tvp = new TVPolitron();
    TVSamsung tvs = new TVSamsung();
    RemoteControl rc = new RemoteControl();

    Scanner scanner = new Scanner(System.in);
    System.out.print("Pilihan Kanal[1], Volume [2-3] : ");
    int ch=scanner.nextInt();
    System.out.print("Jenis TV [1-2] : ");
    int mtv = scanner.nextInt();
    System.out.print("Nomer Kanal/Volume [0-4] : ");
    int vl=scanner.nextInt();

    switch(mtv) {
        case 1:
            rc.kirimPerintahKeTv(ch, tvp, vl);
            break;
        case 2:
            rc.kirimPerintahKeTv(ch, tvs, vl);
            break;
    }
}
}
}

```

### C. Overloading

- Overloading merupakan konsep dalam pemrograman berorientasi objek yang memungkinkan untuk membuat suatu kelas memiliki beberapa method dengan nama sama tetapi memiliki implementasi atau argumen yang berbeda.
- Dalam bahasa pemrograman Java diperkenankan untuk mendefinisikan dua method atau lebih dengan nama yang sama di dalam satu kelas sepanjang deklarasi dan parameternya berbeda, atau disebut *signature*-nya berbeda.
- Hal ini dimungkinkan asalkan deklarasi method membuat penanda berbeda di satu kelas. Penanda adalah kombinasi nama fungsi/method ditambah daftar parameter. Dengan penanda berbeda, Java mampu membedakan metode mana yang perlu dieksekusi dengan mengenali tipe parameter-parameter yang dilewatkan.

Praktek 3: Buatlah program untuk menerapkan konsep Overloading seperti berikut:

```
package perkalian;

public class kalikan {

    public double kali(int a, int b) {
        double hasil = a*b;
        return hasil;
    }

    public double kali(double a, int b, int c) {
        double hasil = a*b+c;
        return hasil;
    }

    public double kali(int a, double b, int c, double d)
    {
        double hasil = a*b+c*d;
        return hasil;
    }

}
```

B

erikut ini adalah program utamanya:

```
public class Perkalian {

    public static void main(String[] args) {

        kalikan kl = new kalikan();
        System.out.println(kl.kali(2,3));
        System.out.println(kl.kali(2,3,4));
        System.out.println(kl.kali(2,3,4,5));

    }

}
```

#### D. Overloading

- Overriding artinya menimpa sebuah properti atau method dengan properti atau method yang sama namun dengan implementasi yang berbeda.
- Overriding sangat terkait dengan pewarisan (inheritance), yaitu kemampuan untuk menurunkan kelas parent kepada sub-class dan menentukan perilaku yang khusus pada sub-class, sub-class dapat juga menerapkan metode kelas parent.

Praktek 4: Buatlah program untuk menerapkan konsep Overriding seperti berikut:

```
package binatang;  
  
class Hewan {  
    public Hewan() {  
        System.out.println("Hewan Bisa Berjalan");  
    }  
}  
  
class Kucing extends Hewan{  
    public Kucing() {  
        System.out.println("Kucing Bisa Memanjat");  
    }  
}  
  
class Macan extends Kucing{  
    public Macan() {  
        System.out.println("Macan Bisa Berenang");  
    }  
}
```

```
public class Binatang {  
  
    public static void main(String[] args) {  
        Hewan a = new Macan();  
    }  
}
```

## E. Package

Package adalah cara untuk mengelompokkan class dan interface yang ada ke dalam kelompoknya (name space) sehingga lebih mudah diatur.

Pendefinisian nama package harus terletak di bagian paling atas dari source program kita. Untuk mendefinisikan suatu package digunakan keyword *package*. Sintaks pendefinisian nama package adalah sebagai berikut:

```
package namaPackage;
```

Contoh:

```
package siswa;
```

Java menggunakan *package* seperti struktur direktori. Oleh karena itu semua class atau interface yang memiliki definisi package seperti contoh di atas, harus disimpan pada direktori bernama *siswa*.

Kita juga dapat membuat package secara hierarki layaknya struktur direktori.

Contoh:

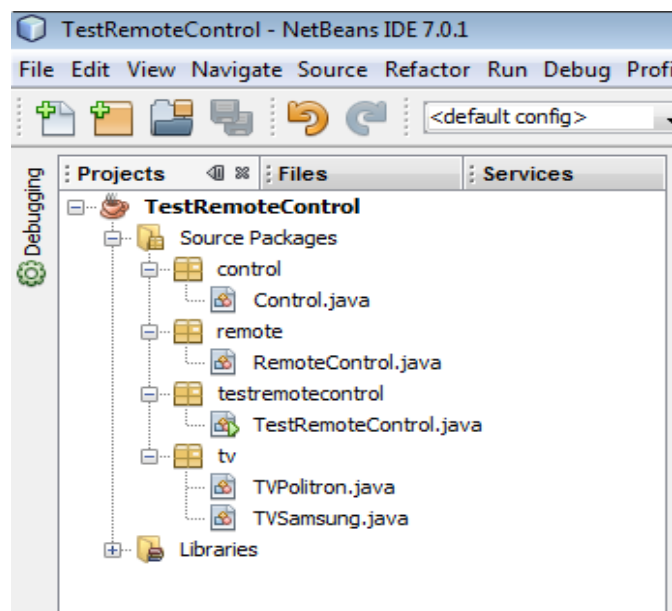
```
package control.contoh;
```

Pada contoh di atas menunjukkan bahwa semua class atau interface yang menggunakan deklarasi package ini harus disimpan pada direktori control → contoh.

Apabila program kita akan menggunakan sebuah class yang terletak pada package yang berbeda, maka kita harus mengimportnya agar dapat digunakan.

### Praktek 5 :

Buatlah program menggunakan package dengan hirarki layaknya struktur direktori seperti pada gambar dibawah:



Sedangkan source programnya adalah sebagai berikut:

Main Program : *TestRemoteControl*

```
package testremotecontrol;
import java.util.Scanner;
import remote.RemoteControl;
import tv.*;
import control.Control;
public class TestRemoteControl {
    public static void main(String[] args){
        TVPolitron tvp = new TVPolitron();
        TVSamsung tvs = new TVSamsung();
        RemoteControl rc = new RemoteControl();
        Scanner scanner = new Scanner(System.in);
        System.out.print("Pilihan Kanal[1], Volume [2-3] : ");
        int ch=scanner.nextInt();
        System.out.print("Jenis TV [1-2] : ");
        int mtv = scanner.nextInt();
        System.out.print("Nomer Kanal/Volume [0-4] : ");
        int vl=scanner.nextInt();

        switch(mtv){
            case 1:
                rc.kirimPerintahKeTv(ch, tvp, vl);
                break;
            case 2:
                rc.kirimPerintahKeTv(ch, tvs, vl);
                break;
        }
    }
}
```

Class : *RemoteControl*

```
package remote;
import control.Control;
import tv.*;
public class RemoteControl {
    public void kirimPerintahKeTv(int aksi,Control tv,int tombol){
        switch(aksi){
            case 1:
                tv.pindahChannel(tombol);
                break;
            case 2:
                tv.perbesarVolume(tombol);
                break;
            case 3:
                tv.perkecilVolume(tombol);
                break;
        }
    }
}
```



Class : *TV Politron*

```
package tv;

import control.Control;
public class TVPolitron implements Control{
    String[] channel =
{"RCTI", "SCTV", "INDOSIAR", "ANTV", "TV7"};
    public void pindahChannel(int channel) {
        System.out.println("TV Politron pindah channel ke "+
this.channel[channel]);
    }
    public void perbesarVolume(int intensitas) {
        System.out.println("TV Politron perbesar volume ke "+
intensitas);
    }
    public void perkecilVolume(int intensitas) {
        System.out.println("TV Politron perkecil volume ke "+
intensitas);
    }
}
}
```

Class : *TV Samsung*

```
package tv;

import control.Control;
public class TVSamsung implements Control{
    String[] channel =
{"RCTI", "SCTV", "INDOSIAR", "ANTV", "TV7"};
    public void pindahChannel(int channel) {
        System.out.println("TV Samsung pindah channel ke "+
this.channel[channel]);
    }
    public void perbesarVolume(int intensitas) {
        System.out.println("TV Samsung perbesar volume ke "+
intensitas);
    }
    public void perkecilVolume(int intensitas) {
        System.out.println("TV Samsung perkecil volume ke "+
intensitas);
    }
}
}
```

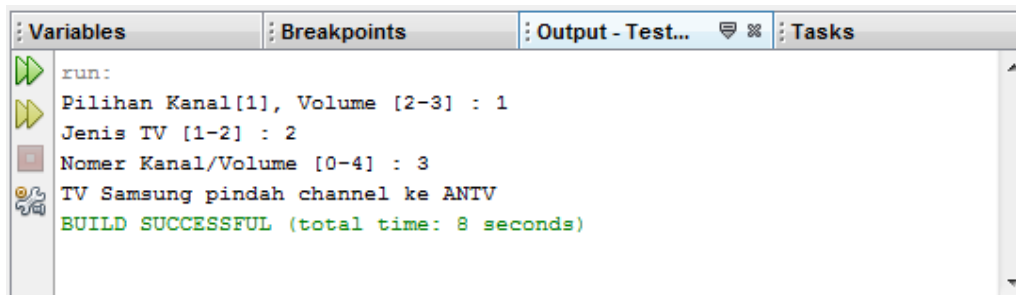
Program interface berikut ini berada dalam direktori *control*.

Interface : *control*

```
package control;

public interface Control {
    public void pindahChannel(int channel);
    public void perbesarVolume(int intensitas);
    public void perkecilVolume(int intensitas);
}
```

Jika telah selesai dibuat, maka apabila program dijalankan dengan sukses akan menghasilkan tampilan sebagai berikut:



```
run:
Pilihan Kanal[1], Volume [2-3] : 1
Jenis TV [1-2] : 2
Nomer Kanal/Volume [0-4] : 3
TV Samsung pindah channel ke ANTV
BUILD SUCCESSFUL (total time: 8 seconds)
```