IPB University
Bogor Indonesia

KOM120C -- PEMROGRAMAN

Object Oriented Programming

Pewarisan dalam Java

Tim Pengajar Pemrograman IPB University

Pewarisan

Konsep-konsep dasar OO pada Java mirip dengan yang telah dipelajari pada C++ dengan beberapa perbedaan pada implementasi.

Inheritance pada Java hanya bersifat **single-inheritance**. Sebagai gantinya, Java mendukung multiple interface-inheritance. Konsep interface akan dijelaskan di pertemuan yang lain.

```
<Subclass> extends <Superclass>{...}
```

Pewarisan

Dalam Java, semua class, termasuk class yang membangun Java API, adalah subclasses dari superclass Object.

Definisi subclass

```
<modifier> class <subclass> extends <superclass> {
   <attributeDeclaration>*
   <constructorDeclaration>*
   <methodDeclaration>*
}
```

Contoh:

```
public class Student extends Person {
            //beberapa kode di sini
}
```

Super

Memanggil constructor secara eksplisit dari superclass terdekat.

Pemanggil super() hanya dapat digunakan dalam definisi constructor.

Pemanggil super() harus dijadikan sebagai pernyataan atau instruksi pertama dalam constructor.

Dapat dipakai sebagai penunjuk anggota superclass, misalnya

```
public Student() {
    super.nama="Siapa Saja";
    super.usia=17;
}
```

is-a Relationship

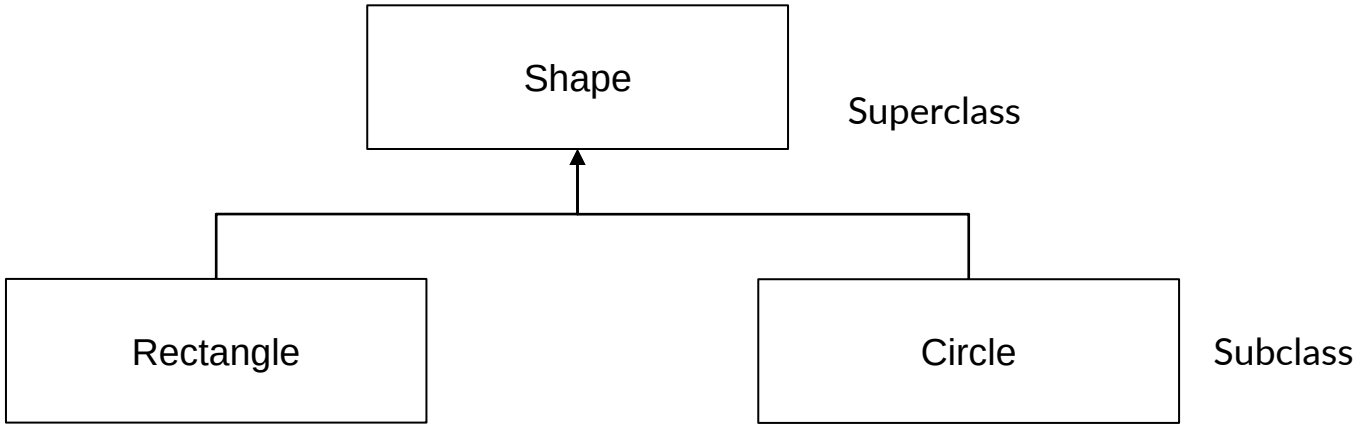
Dalam Java, pewarisan adalah bentuk hubungan **is-a**.

Artinya, kita menggunakan pewarisan hanya jika ada hubungan **is-a** antar 2 class.

Contoh:

- Car is a Vehicle
- Orange is a Fruit
- Surgeon is a Doctor
- Dog is an Animal

is-a Relationship



Contoh dan Latihan

```
class Shape{
  protected String color;
  protected double area;

  public Shape(String a){
    color = a;
  }
  public String getColor(){
    return color;
  }
  public double getArea(){
    return area;//error?}

  public void print(){
    System.out.println("Color:" + color);
  }
}
```

```
class Rectangle extends Shape{
  protected double width;
  protected double height;

  public Rectangle(String a, double b, double c){
    super(a);
    width = b;
    height = c;
  }

  public double getWidth(){return width;}
  public double getHeight(){return height;}

  public void print(){
    super.print();
    System.out.printf("W:%.2f H:%.2f\n",
      width, height);
  }
}
```

Latihan: Buatlah class Circle, turunan dari Shape, yang memiliki atribut radius. Sesuaikan implementasi fungsi print() pada class Circle.

Contoh

```
public class Driver
{
  public static void main(String[] args){
    Shape shp = new Shape("White");
    shp.print();

    Rectangle rect = new Rectangle("Black", 10, 15);
    rect.print();

    Circle circle = new Circle("Blue", 10);
    circle.print();
  }
}
```

Apakah keluaran dari program di atas? Konsep apakah yang digunakan pada contoh di atas?

Override

Subclass (kelas turunan) dapat meng-**override** method pada superclass dengan memberikan implementasi yang berbeda.

Method pada superclass akan memiliki signature dan tipe kembalian yang identik dengan method yang ada pada superclass.

Subclass akan menggunakan implementasi pada dirinya.

Superclass:

```
public class A{
  protected int x, y;
  ...
  public void calc(){return x + y;}
}
```

Subclass:

```
public class B extends A{
  protected int x, y;
  ...
  public void calc(){return x * y;}
}
```

Latihan

Tambahkan method **calculateArea()** pada class Shape, Rectangle, dan Circle.

```
public class Driver2{
  public static void main(String[] args){

    Rectangle rect = new Rectangle("Black", 10, 15);
    System.out.println(rect.getArea());

    Circle circle = new Circle("Blue", 10);
    System.out.println(circle.getArea());
  }
}
```

Output:

```
150.0
314.0
```

Apa implementasi method calculateArea() pada class Shape? Mengapa?

Keyword `final`

Keyword **final** pada Java merupakan suatu modifiers terhadap suatu class, method, atau atribut.

Pada class, final membuat bahwa definisi dari class telah lengkap dan class tersebut tidak dapat diwarisi.

```
public final MyClass{...}
```

Pada method, final membuat method tersebut tidak dapat di-*override*.

```
public final int calculate(...){...}
```

Pada atribut / variabel, final membuat atribut tersebut nilainya tidak dapat diubah setelah diinisialisasi. Inisialisasi wajib dilakukan.

```
public final String language = "Java";
```

Final

Upaya untuk menurunkan class, meng-override method, atau memberikan nilai pada atribut final akan menyebabkan kode program gagal dikompilasi.

```
class Shape{
  ...
  public final void print(){
    System.out.println("Color:" +
color);
  }
}
```

```
class Rectangle extends Shape{
  ...
  public void print(){
    super.print();
    System.out.printf("W:%.2f H:%.2f\n",
      width, height);
  }
}
```

```
Driver2.java:30: error: print() in Rectangle cannot override print() in Shape
  public void print(){
              ^
  overridden method is final
```

Latihan

Pak Agria memiliki tanah di N buah lokasi. Tanah tersebut berbentuk persegi panjang (panjang dan lebar) atau lingkaran (radius). Pak Agria memagari tanah miliknya. Akan tetapi, karena dana Pak Agria terbatas, ia hanya mampu memasang pagar di sebagian tanah miliknya. Dalam hal ini, Pak Agria hanya akan memasang pagar di tanah yang kelilingnya di atas rata-rata. Bantulah Pak Agria menghitung panjang pagar yang perlu ia sediakan.

Format Masukan:

- Baris pertama: N
- Digit pertama setiap baris adalah bentuk tanah (0 = persegi panjang, 1 = lingkaran).
- Digit-digit berikutnya adalah panjang dan lebar, atau radius bergantung pada bentuk tanah.

Contoh Masukan:

```
3
0 10.0 10.0
0 20.0 20.0
1 10.0
```

Contoh Keluaran:

```
142.83
```

Tantangan: bagaimana jika bentuk tanah pada masukan tidak tersedia?