Bahasa Pemrograman :: Inheritance

Julio Adisantoso
ILKOM IPB

17 Mei 2010

Class Person

```
public class Person {
   private String name;
   private String address;
   private int age;

   //constructors
   Person() { name=address=""; age=0; }
   public Person(String nm, String ad, int th) {
      name=nm; address=ad; age=th; }

   ... //kode selanjutnya
}
```

Class Person

```
public class Person {

   ... //kode sebelumnya
   public void setName(String nm) { name=nm; }
   public void setAddress(String ad) { address=ad; }
   public void setAge(int th) { age=th; }

   public String getName() { return name; }
   public String getAddress() { return address; }
   public int getAge() { return age; }
   public void print() {
      System.out.print  ( name+", " );
      System.out.println( address+", "+age ); }
}
```

Class Student

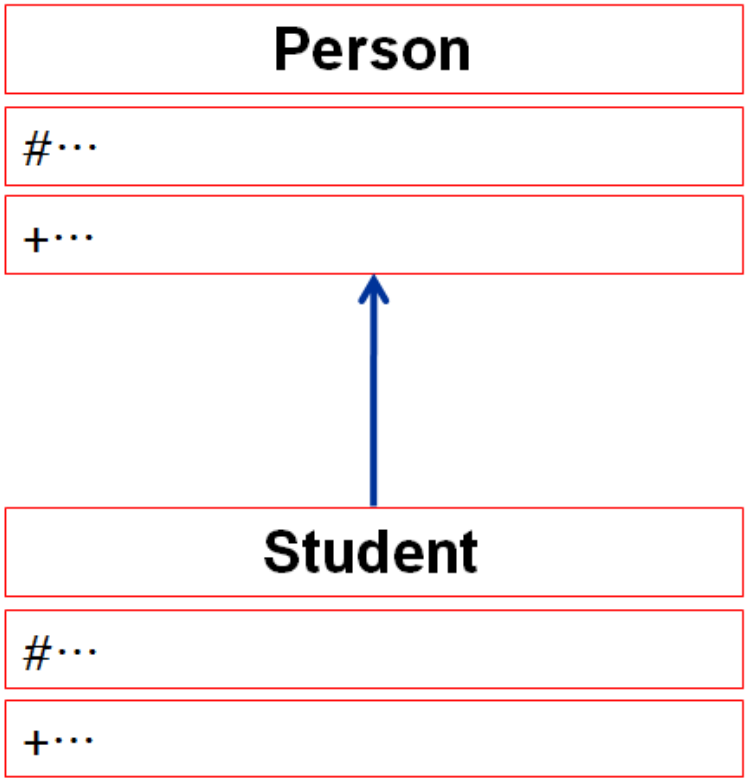
```
public class Student {
   private String name;
   private String address;
   private int age;
   private String nim;
   private double ipk;

   Student() { ... }
   public Student(String nm, String ad, int th,
      String nim, double ipk) { ... }
   public setName(String nm) { ... }
   public setAddress(String ad) { ... }
   public setAge(int ag) { ... }
   public setNIM(String nim) { ... }
   public setIPK(double ipk) { ... }
   ...
}
```

Pewarisan

- Beberapa sifat class Person sama dengan class Student:
  - Instance variables : name, address, age
  - Instance methods : setName(), setAddress(), setAge(), dsb
- Ada mekanisme pewarisan, artinya beberapa sifat class Person diturunkan/diwariskan ke class Student.
- Class Student merupakan turunan dari class Person:
  - Class Person disebut Base Class, Super Class
  - Class Student disebut Derive Class, Sub Class

UML Presentation



Class Person

```
public class Person {
   protected String name;
   protected String address;
   protected int age;

   //constructors
   public Person() { name=address=""; age=0; }
   public Person(String nm, String ad, int ag) {
      name=nm; address=sd; age=ag; }

   ... //lainnya sama
}
```

Class Student

```
public class Student extends Person {
   private String nim;
   private double ipk;

   //constructors
   public Student() { super(); nim=""; ipk=0.0; }
   public Student( String nm, String ad, int ag,
      String nim, double ipk ) {
      super(nm, ad, ag);
      this.nim=nim; this.ipk=ipk; }

   ... //kode lainnya
}
```

Class Test

```
public class Test {
   public static void main( String[] args ) {
      Person  p1=new Person("Gayus", "Jakarta", 40);
      Student s1=new Student("Manohara", "Bogor", 21,
         "G64221234", 3.7);
      Person  p2=new Student();
      p1.print();
      s1.print();
      p2.print();
      System.out.println( s1.getAge() );
   }
}
```

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.name="Syahrini";
   super.address="Jakarta";
}
```

Override

- Untuk beberapa pertimbangan, kadang-kadang method pada subclass 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.
- Jadi, method pada subclass sama dengan superclass, tetapi memiliki implementasi yang berbeda, sehingga akan di-override

Contoh Override

```
public class Person {
    ...
    public void print() {
        System.out.println( "Nama Orang : "+name ) }
    ...
}

public class Student extends Person {
    ...
    public void print() {
        System.out.println( "Nama Siswa : "+name ) }
    ...
}
```

Final

- Ada 2 jenis: final class dan final method
- Final class adalah class yang sifat-sifatnya tidak dapat di-override, misalnya String, Integer, Double
- Final method adalah method yang tidak dapat di-override
- Contoh

```
public final String getName(){
    return name;
}
```

Class Segiempat

Buat *class* bernama Segiempat untuk mengimplementasikan obyek segiempat yang memiliki atribut panjang dan lebar, keduanya bertipe *double*. *Class* ini memiliki *default constructor* dan *constructor* lainnya, juga memiliki *mutator method* dan *accessor method* yang umum. Disamping itu, *class* ini juga memiliki *method* bernama **area()** untuk menghitung luas segiempat.