

## BAB 6

# Inheritance

### Tujuan

1. Praktikan dapat memahami sub konsep sub class dan super class dalam konsep Pewarisan
2. Praktikan mampu menerapkan konsep encapsulation dalam Pewarisan

### Ringkasan Materi

#### A. Konsep Dasar Inheritance

Inheritance (Pewarisan) merupakan salah satu dari tiga konsep dasar OOP. Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan.

Suatu class yang mempunyai class turunan dinamakan **parent class** atau **base class** atau **super class**. Sedangkan class turunan itu sendiri seringkali disebut **subclass** atau **child class**. Suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class.

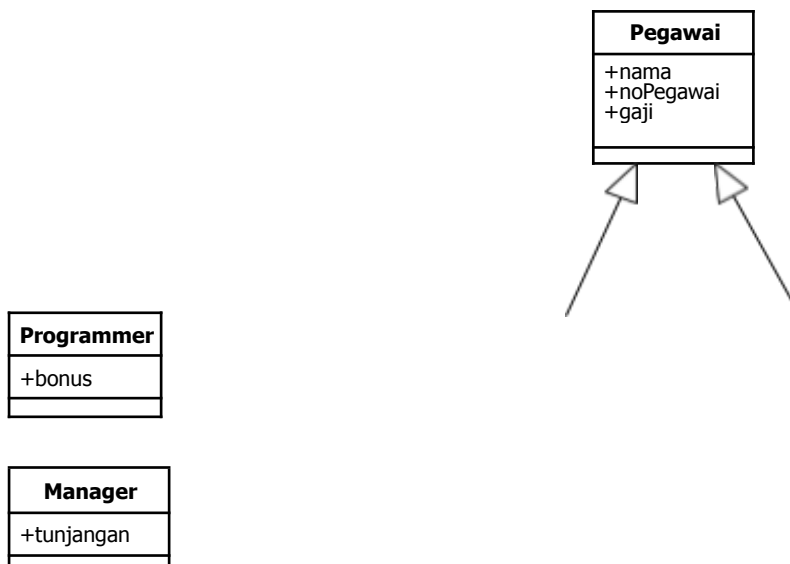
Karena suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class-nya, maka member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang ia warisi dari class parent-nya.

Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class-nya.

#### Diagram UML

Adalah sebuah Bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun dan mendokumentasikan dari sebuah system pengembangan software OOP (Object Oriented Programming).

Contoh Inheritance pada diagram UML :



#### B. Deklarasi Inheritance

Dengan menambahkan kata kunci **extends** setelah deklarasi nama class, kemudian diikuti dengan nama parent class-nya. Kata kunci **extends** tersebut memberitahu kompiler Java bahwa kita akan melakukan perluasan class, contoh deklarasi :

```

public class Programmer extends Pegawai { ... }
public class Manager extends Pegawai { ... }
  
```

**Pelaksanaan Percobaan**

Employee.java	
1	import java.util.*;
2	public class Employee {
3	private String name;
4	private double salary;
5	private Date hireday;
6	public Employee(String name, double salary, int year,int
7	month, int day){
8	this.name = name;
9	this.salary = salary;
10	GregorianCalendar calendar = new
11	GregorianCalendar(year,month-1,day);
12	this.hireday = calendar.getTime();
13	}
14	public Date getHireDay(){
15	return hireday;
16	}
18	public String getName(){
19	return name;
20	}
21	public double getSalary(){
22	return salary;
23	}
24	public void raiseSalary(double byPercent){
25	double raise = salary * byPercent/100;
26	salary+=raise;
27	}
28	}

**Ketikkan SubClass di bawah ini**

Manager.java	
1	public class Manager extends Employee {
2	private double bonus;
3	public Manager(String name, double salary, int year, int
4	month, int day){
5	super(name, salary, year, month, day);
6	bonus = 0;
7	}
8	public void setBonus(double bonus){
9	this.bonus = bonus;
10	}
11	public double getSalary(){
12	double baseSalary = super.getSalary();
13	return baseSalary+bonus;
14	}
15	
16	}

**Main Class**

Employee.java

```

1 public class MainEmployee {
2     public static void main(String[] args) {
3         Manager boss = new Manager("Steven", 80000, 1987, 12,
4 15);
5         boss.setBonus(5000);
6         Employee staff = new Employee("Donni", 50000, 1989, 10,
7 1);
8         System.out.println("nama boss : "+boss.getName()+"",
9 salary = "+boss.getSalary());
10        System.out.println("nama staff : "+staff.getName()+"",
11 salary = "+staff.getSalary());
12    }
13 }

```

**Data dan Analisis hasil percobaan**

Pertanyaan

1. Jalankan code program diatas dan benahi jika menemukan kesalahan!

Tidak ada kesalahan.

2. Bagaimana cara konstruktor pada subclass memanggil konstruktor di superclass nya? Apakah hal itu perlu dilakukan? Sertakan alasan anda !

Konstruktor dari superclass dipanggil menggunakan super(). Ini harus dilakukan di dalam konstruktor subclass dan harus merupakan statement pertama. Jika sebuah konstruktor tidak secara eksplisit memanggil konstruktor superclass menggunakan super(), kompiler Java akan secara otomatis memasukkan panggilan ke konstruktor default superclass (konstruktor tanpa argumen) jika ada. Jika superclass tidak memiliki konstruktor tanpa argumen dan subclass tidak memanggil konstruktor lain, maka akan terjadi error kompilasi.

3. Tambahkan constructor pada class Employee dengan parameter *String name*! amati perubahan apa yang terjadi, jelaskan jawaban anda!

```

public Employee(String name) {
    this.name = name;
}

```

Tidak ada perubahan signifikan yang terjadi, dan tidak terjadi error juga. Hal ini dikarenakan konstruktor Employee dengan parameter String name belum dipanggil di main class dan juga memiliki parameter yang berbeda dari konstruktor Employee yang telah ada, sehingga tidak juga terjadi error.

4. Pada Class Manager baris ke 5, setelah variable day tambahkan variable bonus! Amati apa yang terjadi dan mengapa demikian?

```

public Manager(String name, double salary, int year, int month, int day) {
    super(name, salary, year, month, day, bonus);
    bonus = 0;
}

```

Yang terjadi adalah error, hal ini dikarenakan class Manager merupakan anak dari class Employee yang dimana parameter dari konstruktornya juga harus sama seperti parameter yang ada di kelas parentnya

5. Untuk apa digunakan keyword this pada class manager dan employee? Hapus keyword this dan amati apa yang terjadi?

```
public void setBonus(double bonus) {
    bonus = bonus;
}
```

```
public Employee(String name, double salary, int year, int month, int day) {
    name = name;
    salary = salary;
    GregorianCalendar calendar = new GregorianCalendar(year, month - 1, day);
    hireday = calendar.getTime();
}
```

```
nama boss : null, salary = 0.0
nama staff : null, salary = 0.0
```

Yang terjadi adalah, programnya bingung variable name yang dimaksud ini merujuk ke variabel yang ada di parameter atau variabel kelas. Sehingga output yang dihasilkan adalah bernilai null dan 0.

6. Tambahkan constructor pada class Employee dengan parameter Bertipe data string bernama name yang nantinya bila constructor ini akan dipanggil akan menginisialisasi variable name! Amati perubahannya pada class anak dan jelaskan! Benahi bila terjadi kesalahan!

```
public Employee(String name) {
    this.name = name;
}
```

Tidak ada perubahan signifikan yang terjadi pada kelas anak. Hal ini dikarenakan konstruktor baru yang ditambahkan pada Employee memungkinkan Manager untuk menginisialisasi nama karyawan saat objek Manager dibuat. Sehingga, nama karyawan akan diinisialisasi melalui konstruktor kelas induk Employee.

7. Pada bab sebelumnya anda telah belajar mengenai konsep encapsulation, jelaskan mengapa pada super class menggunakan modifier protected? Apa yang terjadi jika modifier anda ubah menjadi private atau public? Jelaskan !
  - Dengan menggunakan modifier protected, atribut atau metode tersebut dapat diakses oleh kelas-kelas dalam package yang sama dan oleh subclass di package yang berbeda. Hal ini memungkinkan subclass untuk mengakses atribut atau metode tersebut.
  - Jika modifier diubah menjadi private, maka atribut atau metode tersebut hanya dapat diakses di dalam kelas tersebut sendiri. Ini berarti kelas turunan tidak akan dapat mengakses atau menggunakan atribut atau metode tersebut.
  - Jika modifier diubah menjadi public, maka atribut atau metode tersebut akan dapat diakses oleh kelas-kelas di luar package dan tanpa batasan. Hal ini dapat mengakibatkan hilangnya kontrol terhadap penggunaan dan manipulasi atribut atau metode tersebut.

8. Ubahlah acces modifier method pada kelas employee menjadi :
- Private
  - Protected

Amati perubahan apa yang terjadi? Jelaskan jawaban anda dengan detail!

```
Employee.java 4
MainEmploy... 4
Manager.java 2
```

a.

Yang terjadi adalah error di kelas lainnya, termasuk subclass dari Employee itu sendiri. Hal ini dikarenakan atribut atau metode tersebut hanya dapat diakses di dalam kelas tersebut sendiri. Ini berarti kelas turunan tidak akan dapat mengakses atau menggunakan atribut atau metode tersebut.

```
Employee.java
MainEmployee.java
Manager.java
```

b.

```
nama boss : Steven, salary = 85000.0
nama staff : Donni, salary = 50000.0
```

Tidak terjadi error, dan program dapat berjalan dengan lancar. Hal ini dikarenakan atribut atau metode tersebut dapat diakses oleh kelas-kelas dalam package yang sama dan oleh subclass di package yang berbeda.

### Tugas Praktikum

Susunlah program sesuai studi kasus di bawah ini:

#### 1. Manusia.java

Kelas manusia merupakan kelas induk dengan definisi sebagai berikut:

- nama : String
- jenisKelamin : boolean (true : laki-laki, false : perempuan)
- nik : String
- menikah : boolean
- + setter, getter
- + getTunjangan() : double
- + getPendapatan() : double
- + toString() : String

(Keterangan)

- o Tunjangan untuk yang telah menikah adalah apabila laki-laki akan mendapat \$25 sedangkan perempuan mendapat \$20.
- o Tunjangan untuk yang belum menikah adalah \$15 .
- o toString() menampilkan nama, nik, jenis kelamin, dan jumlah pendapatan.

#### 2. MahasiswaFILKOM.java

Kelas mahasiswa merupakan kelas turunan dari Manusia dengan definisi sebagai berikut:

- nim : String
- ipk : double
- + setter, getter
- + getStatus() : String
- + getBeasiswa() : double
- + toString() : String

(Keterangan)

- o Beasiswa untuk ipk 3.0 – 3.5 mendapat \$50 dan untuk 3.5 – 4 mendapat \$75
- o Status untuk mendapatkan angkatan dan prodi (menurut kaidah FILKOM UB) 1651506XXXXXX
  - o Digit ke 1-2 adalah angkatan
  - o Digit ke 7 adalah prodi
    - 2 Teknik Informatika

- 3 Teknik Komputer
- 4 Sistem Informasi
- 6 Pendidikan Teknologi Informasi
- 7 Teknologi Informasi

Dengan pengembalian dengan format : prodi angkatan, contoh : Sistem Informasi, 2017

- toString() menampilkan atribut induk + nim, ipk, dan status.

### 3. Pekerja.java

Kelas Pekerja merupakan kelas turunan dari Manusia dengan definisi sebagai berikut:

- gaji : double
- tahunMasuk : LocalDate
- jumlahAnak : int
- + setter, getter
- + getBonus() : double
- + getGaji() : double
- + toString() : String

(Keterangan)

- o Bonus didapatkan oleh pegawai sesuai lama bekerja :
  - o Jika lama bekerja 0 – 5 tahun, maka bonus sebesar 5% dari gaji
  - o Jika lama bekerja 5 – 10 tahun, maka bonus sebesar 10% dari gaji
  - o Jika lebih dari 10 tahun, maka bonus sebesar 15% dari gaji
- o Tunjangan ditambah apabila memiliki anak, yaitu \$20 per anak.
- o toString() menampilkan atribut induk + tahun masuk, jumlah anak, dan gaji.

### 4. Manager.java

Kelas Manager merupakan kelas turunan dari Pekerja dengan definisi sebagai berikut:

- departemen : String
- + setter, getter

(Keterangan)

- o Tunjangan ditambah sebesar 10% dari gaji.
- o toString() menampilkan atribut induk + departemen.

Dari semua kelas yang telah dibuat, buatlah testcase (mencetak objek / toString()) untuk kasus berikut :

#### 1. Manusia

- a. Laki-laki telah menikah.
- b. Perempuan telah menikah.
- c. Belum menikah.

#### 2. MahasiswaFilkom (sesuai status Anda)

- a. Ipk < 3
- b. Ipk 3 – 3.5
- c. Ipk 3.5 – 4

#### 3. Pekerja

- a. Lama bekerja 2 tahun, anak 2
- b. Lama bekerja 9 tahun
- c. Lama bekerja 20 tahun, anak 10

#### 4. Manager, lama bekerja 15 tahun dengan gaji \$7500

## Contoh Input testcase – output

(\*Tidak perlu dimasukkan laporan\*)

Manusia a = new Manusia("A", "111", true, true); System.out.println(a);	nama : A nik : 111 jenisKelamin : Laki-laki pendapatan : 25.0
MahasiswaFILKOM b = new MahasiswaFILKOM("165150300111100", 4.0, "B", "111", false, false); System.out.println(b);	nama : B nik : 111 jenisKelamin : Perempuan pendapatan : 90.0 nim : 165150300111100 ipk : 4.0 status : Teknik Komputer, 2016
Pekerja c = new Pekerja(1000, 2016, 3, 2, 4, "C", "111", true, true); System.out.println(c);	nama : C nik : 111 jenisKelamin : Laki-laki pendapatan : 1155.0 tahun masuk2 3 2016 jumlah anak : 4 gaji : 1000.0
Manager d = new Manager("HRD", 1000, 2017, 1, 2, 3, "D", "111", true, true); System.out.println(d);	nama : D nik : 111 jenisKelamin : Laki-laki pendapatan : 1235.0 tahun masuk2 1 2017 jumlah anak : 3 gaji : 1000.0 departemen : HRD