JOBSHEET W07

INHERITANCE, OVERLOADING, OVERRIDING

1. KOMPETENSI

1. Memahami konsep dasar inheritance
2. Mampu membuat suatu subclass dari suatu superclass tertentu.
3. Mampu membuat objek dari suatu subclass dan melakukan pengaksesan terhadap atribut dan method baik yang dimiliki sendiri atau turunan dari superclass nya.
4. Mampu membuat method overloading
5. Mampu membuat method overriding

2. PENDAHULUAN

Inheritance pada object oriented programming merupakan konsep **pewarisan** dari suatu class yang lebih umum ke suatu class yang lebih spesifik. Kelas yang menurunkan disebut kelas dasar (**base class/super class/parent class**), sedangkan kelas yang diturunkan disebut kelas turunan (**derived class/sub class/child class**). Setiap **subclass** akan "mewarisi" atribut dan method dari **superclass** yang bersifat *public* ataupun *protected*. Manfaat pewarisan adalah *reusability* atau penggunaan kembali baris kode.

Pada bahasa pemrograman Java, deklarasi inheritance dilakukan dengan cara menambahkan kata kunci **extends** setelah deklarasi nama class, kemudian diikuti dengan nama parent class nya. Kata kunci extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan **extension/ perluasan** class. Berikut adalah contoh deklarasi inheritance.

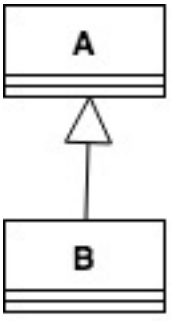
```
public class B extends A {

...
}
```

Contoh diatas memberitahukan kompiler Java bahwa class B meng-**extend** class A. Artinya, class B adalah subclass dari class A dengan melakukan extension/perluasan. Extension atau perluasan ini akan dilakukan dengan panambahan atribut dan method khusus yang hanya dimiliki oleh class B.

Terdapat 3 bentuk pewarisan: single inheritance, multilevel inheritance, dan multiple inheritance.

1. Single Inheritance
   Single inheritance adalah inheritance dimana suatu subclass hanya mempunyai satu parent class.

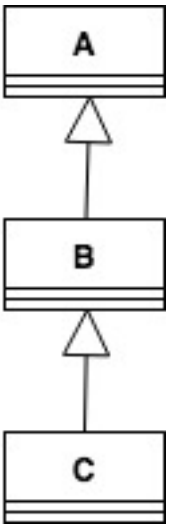


Gambar 1. Contoh Single Inheritance

2. Multilevel Inheritance

   Multilevel inheritance adalah inheritance dengan subclass yang menjadi superclass bagi classyang lain.

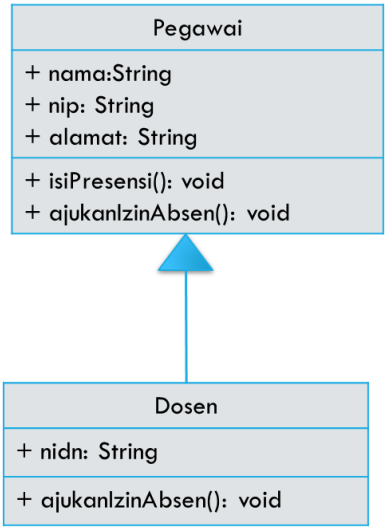
   Contoh:



Gambar 2. Contoh Multilevel Inheritance

Pada Gambar 2 di atas dapat dilihat bahwa class B merupakan subclass dari class A, namun dia jugamerupakan superclass dari class C. Inheritance dengan 2 level ini disebut multilevel inheritance.

Pada class diagram, inheritance digambarkan dengan sebuah garis solid dengan segitiga di ujungnya. Class yang dekat pada segitiga merupakan superclass, sedangkan class yang jauh dari segitiga merupakan subclass. Berikut ini adalah contoh class diagram dengan relasi inheritance:



Gambar 3. Contoh class diagram dengan inheritance

Suatu parent class bisa membatasi atribut dan method yang akan diwariskan kepada subclass-nya. Pembatasan tersebut dilakukan melalui penentuan access level modifier. Di dalam java, access level modifier atribut dan method dirangkum dalam tabel berikut ini:

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

Atribut dan method yang akan diwariskan dari parent class ke child class adalah atribut dan method dengan modifier protected atau public.

Kata kata kunci **this** dipakai untuk merujuk pada object/class itu sendiri. Sementara itu,kunci **super** dipakai untuk merujuk pada parent object/class. Format penulisannya adalah sebagai berikut:

- **super.<namaAtribut>**
  Mengakses atribut parent
- **super.<namaMethod>()**
  Memanggil method parent
- **super()**
  Memganggil constructor parent, hanya dapat dilakukan pada baris pertama dalam constructor child
- **super(parameter1, parameter2,dst)**
  Memanggil constructor parent class dengan parameter, hanya dapat dilakukan pada baris pertama dalam constructor child class

Saat instansiasi objek dari subclass dilakukan, objek pada superclass juga akan terbentuk. Dengan kata lain, ketika constructor subclass dijalankan, pada "baris pertama" (atau sebelum baris- baris lainnya dalam constructor subclass dieksekusi) constructor superclass akan dijalankan terlebih dahulu.

Polymorphism terdiri dari 2 kata, yaitu poly (banyak), morph (bentuk). Konsep polimorfisme pada OOP membolehkan sebuah aksi diimplementasikan secara berbeda. Ada 2 bentuk polimorfisme, yaitu:

1. **Overloading**
   - Method overloading berarti kondisi dimana ada method dengan nama yang sama, tetapimemiliki method signature yang berbeda.
   - Method signature: jumlah, tipe data dan susunan parameter
   - Method overloading dapat terjadi pada kelas yang sama atau kelas lain yang terkait dalamhierarki pewarisan.
   - Karakteristik overloading: nama method sama, method signature berbeda, return type bolehsama atau berbeda.
   - JVM menentukan method mana yang akan dipanggil pada compile-time compile-timepolymorphism
   - Disebut juga static binding atau early binding

2. **Overriding**
   - Overriding terjadi ketika child class memiliki method dengan nama dan signature yang samadengan parent class nya.
   - Karakteristik: terjadi pada child class/kelas turunan, nama method sama, method signature sama, return type sama
   - JVM menentukan method mana yang akan dipanggil pada saat run-time run-time polymorphism
   - Disebut juga dynamic binding atau late binding

3. PERCOBAAN 1 (extends)

A. TAHAPAN PERCOBAAN

1. Buatlah sebuah parent class dengan nama Pegawai. Lalu buat constructor tanpa parameter dengan baris kode sebagai berikut:

```
public class Pegawai {

    public Pegawai() {
        System.out.println("Objek dari class Pegawai dibuat");
    }
}
```

2. Buatlah subclass dari class Pegawai dengan nama Dosen, kemudian buat juga constructor tanpa parameter dengan baris kode berikut:

```
public class Dosen extends Pegawai {

    public Dosen() {
        System.out.println("Objek dari class Dosen dibuat");
    }

}
```

3. Buatlah main class, misal InheritanceDemo.java, lakukan instansiasi objek baru bernama dosen1 dari class Dosen sebagai berikut:

```
public static void main(String[] args) {
    Dosen dosen1 = new Dosen();
}
```

4. Run programnya kemudian amati hasilnya.

B. PERTANYAAN

1. Pada percobaan 1 diatas, tentukan child class dan parent class!
2. Kata kunci apa yang membuat child class dan parent class tersebut memiliki relasi?
3. Berdasarkan hasil yang ditampilkan oleh program, ada berapa constructor yang dieksekusi saat instansiasi objek dosen1? Constructor class mana yang lebih dulu dieksekusi?

4. PERCOBAAN 2 (Pewarisan)

A. TAHAPAN PERCOBAAN

1. Tambahkan atribut nip, nama, dan gaji serta method getInfo() pada class Pegawai

```
public class Pegawai {
    public String nip;
    public String nama;
    public double gaji;

    public Pegawai() {
        System.out.println("Objek dari class Pegawai dibuat");
    }

    public String getInfo(){
        String info = "";
        info += "NIP         : " + nip + "\n";
        info += "Nama         : " + nama + "\n";
        info += "Gaji         : " + gaji + "\n";

        return info;
    }
}
```

2. Tambahkan pula atribut NIDN pada class Dosen

```
public class Dosen extends Pegawai {
    public String nidn;

    public Dosen() {
        System.out.println("Objek dari class Dosen dibuat");
    }
}
```

3. Pada class InheritanceDemo.java tuliskan baris kode berikut:

```
public static void main(String[] args) {
    Dosen dosen1 = new Dosen();

    dosen1.nama = "Yansy Ayuningtyas";
    dosen1.nip = "34329837";
    dosen1.gaji = 3000000;
    dosen1.nidn = "1989432439";

    System.out.println(dosen1.getInfo());
}
```

4. Run program kemudian amati hasilnya

B. PERTANYAAN

1. Pada percobaan 2 diatas, apakah program dapat berhasil dijalankan ataukah terjadi error?
2. Jika program berhasil dijalankan, mengapa tidak terjadi error pada assignment/pengisian nilaiatribut nip, gaji, dan NIDN pada object dosen1 padahal tidak ada deklarasi ketiga atribut tersebut pada class Dosen?
3. Jika program berhasil dijalankan, mengapa tidak terjadi error pada pemanggilan method getInfo() oleh object dosen1 padahal tidak ada deklarasi method getInfo() pada class Dosen?

5. PERCOBAAN 3 (Hak akses)

A. TAHAPAN PERCOBAAN

1. Modifikasi access level modifier pada atribut gaji menjadi private pada class Pegawai.java

```
public class Pegawai {
    public String nip;
    public String nama;
    private double gaji;
```

2. Run program kemudian amati hasilnya.
3. Ubah access level modifier atribut gaji menjadi protected kemudian pindah class Pegawai ke package baru, misalnya "testpackage".

```
package testpackage;

public class Pegawai {
    public String nip;
    public String nama;
    protected double gaji;
```

4. Import class Pegawai dari testpackage pada class Dosen.

```
import testpackage.Pegawai;
```

5. Akses atribut gaji pada class Dosen dengan coba mencetak atribut gaji pada constructor Dosen

```
public Dosen() {
    System.out.println(gaji);
    System.out.println("Objek dari class Dosen dibuat");
}
```

6. Run program kemudian amati hasilnya
7. Ubah kembali access level modifier menjadi public dan kembalikan class Pegawai ke package semula.

B. PERTANYAAN

1. Pada langkah 1-2 di atas, terjadi error karena object dosen1 tidak dapat mengakses atributgaji. Padahal gaji merupakan atribut Pegawai yang merupakan parent class dari Dosen. Mengapa hal ini dapat terjadi?
2. Pada langkah 5-6, setelah class Pegawai berpindah ke package yang berbeda, class Dosen masih dapat mengakses atribut gaji. Mengapa?
3. Berdasarkan percobaan tersebut, bagaimana cara menentukan atribut dan method yang akan diwariskan oleh parent class ke child class?

6. PERCOBAAN 4 (Super - atribut)

A. TAHAPAN PERCOBAAN

1. Butlah method getAllInfo() pada class Dosen

```
public String getAllInfo(){
    String info = "";
    info += "NIP         : " + nip + "\n";
    info += "Nama        : " + nama + "\n";
    info += "Gaji        : " + gaji + "\n";
    info += "NIDN        : " + nidn + "\n";

    return  info;
}
```

2. Lakukan pemanggil method getAllInfo() oleh object dosen1 pada main function.

```
public static void main(String[] args) {
    Dosen dosen1 = new Dosen();

    dosen1.nama = "Yansy Ayuningtyas";
    dosen1.nip = "34329837";
    dosen1.gaji = 3000000;
    dosen1.nidn = "1989432439";

    System.out.println(dosen1.getAllInfo());
}
```

3. Run program kemudian amati hasilnya
4. Lakukan modifikasi method getAllInfo() pada class Dosen

```
public String getAllInfo(){
    String info = "";
    info += "NIP         : " + this.nip + "\n";
    info += "Nama        : " + this.nama + "\n";
    info += "Gaji        : " + this.gaji + "\n";
    info += "NIDN        : " + this.nidn + "\n";

    return  info;
}
```

5. Run program kemudian bandingkan hasilnya dengan langkah no 2.
6. Lakukan modifikasi method getAllInfo() pada class Dosen kembali

```
public String getAllInfo(){
    String info = "";
    info += "NIP         : " + super.nip + "\n";
    info += "Nama        : " + super.nama + "\n";
    info += "Gaji        : " + super.gaji + "\n";
    info += "NIDN        : " + super.nidn + "\n";

    return  info;
}
```

7. Run program kemudian bandingkan hasilnya dengan progam pada no 1 dan no 4.

8. Lakukan modifikasi method getAllInfo() pada class Dosen kembali

```
public String getAllInfo(){
    String info = "";
    info += "NIP         : " + super.nip + "\n";
    info += "Nama        : " + super.nama + "\n";
    info += "Gaji        : " + super.gaji + "\n";
    info += "NIDN        : " + this.nidn + "\n";

    return  info;
}
```

9. Run program kemudian bandingkan hasilnya dengan progam pada no 2 dan no 4.

B. PERTANYAAN

1. Apakah terdapat perbedaan output pada langkah 1, 4, dan 8? Mengapa?
2. Mengapa error terjadi pada program no 6?

7. PERCOBAAN 5 (super & overriding)

A. TAHAPAN PERCOBAAN

1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya.

```
public String getAllInfo(){
    String info = getInfo();
    info += "NIDN        : " + nidn;

    return info;
}
```

2. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amatihasilnya

```
public String getAllInfo(){
    String info = this.getInfo();
    info += "NIDN        : " + nidn;

    return info;
}
```

3. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amatihasilnya

```
public String getAllInfo(){
    String info = super.getInfo();
    info += "NIDN        : " + nidn;

    return info;
}
```

4. Tambahkan method getInfo() pada class Dosen dan modifikasi method getAllInfo() sebagai berikut

```
public class Dosen extends Pegawai {
    public String nidn;

    public Dosen() {
        System.out.println("Objek dari class Dosen dibuat");
    }

    public String getInfo(){
        return "NIDN       : " + this.nidn + "\n";
    }

    public String getAllInfo(){
        String info = super.getInfo();
        info += this.getInfo();

        return info;
    }
}
```

B. PERTANYAAN

1. Apakah ada perbedaan method getInfo() yang diakses pada langkah 1, 2, dan 3?
2. Apakah ada perbedaan method super.getInfo() dan this.getInfo() yang dipanggil dalam method getAllInfo() pada langkah 4? Jelaskan!
3. Pada method manakah terjadi overriding? Jelaskan!
4. Tambahkan keyword final pada method getInfo() di class Pegawai. Apakah program dapat dicompile? Mengapa?

8. PERCOBAAN 6 (overloading)

A. TAHAPAN PERCOBAAN

1. Tambahkan constructor baru untuk class Dosen sebagai berikut

```
public Dosen(String nip, String nama, double gaji, String nidn){
    System.out.print("Objek dari class Dosen dibuat dengan constructor berparameter");
}
```

2. Modifikasi class InheritanceDemo untuk menginstansiasi object baru dengan nama dosen2 dengan constructor yang berparameter. Run program kemudian amati hasilnya.

```
public static void main(String[] args) {
    Dosen dosen2 = new Dosen("34329837","Yansy Ayuningtyas",3000000,"1989432439");
    System.out.println(dosen2.getAllInfo());
}
```

B. PERTANYAAN

1. Bagaimana hasil nilai nip, nama, gaji, dan nidn yang ditampilkan pada langkah 2? Mengapa demikian?
2. Jelaskan apakah 2 constructor pada class Dosen memiliki signature yang sama?
3. Konsep apa dalam OOP yang membolehkan suatu class memiliki constructor atau method dengan nama yang sama dan signature yang berbeda pada satu class?

9. PERCOBAAN 7 (super - constructor)

A. TAHAPAN PERCOBAAN

1. Modifikasi constructor berparameter pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    this.nip = nip;
    this.nama = nama;
    this.gaji = gaji;
    this.nidn = nidn;
}
```

2. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    super.nip = nip;
    super.nama = nama;
    super.gaji = gaji;
    this.nidn = nidn;
}
```

3. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    super();
    super.nip = nip;
    super.nama = nama;
    super.gaji = gaji;
    this.nidn = nidn;
}
```

4. Hapus/comment constructor tanpa parameter dari class Pegawai. Tambahkan constructor baru untuk class Pegawai sebagai berikut. Run program kemudian amati hasilnya.

```
public class Pegawai {
    public String nip;
    public String nama;
    public double gaji;

//    public Pegawai() {
//        System.out.println("Objek dari class Pegawai dibuat");
//    }

    public Pegawai(String nip, String nama, double gaji) {
        this.nip = nip;
        this.nama = nama;
        this.gaji = gaji;
    }

    public String getInfo(){
        String info = "";
        info += "NIP         : " + nip + "\n";
        info += "Nama        : " + nama + "\n";
        info += "Gaji        : " + gaji + "\n";

        return info;
    }
}
```

5. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    this.nidn = nidn;
    super(nip, nama, gaji);
}
```

6. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    super(nip, nama, gaji);
    this.nidn = nidn;
}
```

B. PERTANYAAN

1. Apakah terdapat perbedaan hasil pada langkah 1 dan 2? Jelaskan!
2. Apakah terdapat perbedaan hasil pada langkah 2 dan 3? Jelaskan!
3. Mengapa terjadi error pada langkah 4?
4. Apa perbedaan super() yang dipanggil pada langkah 3 dan 6?
5. Mengapa terjadi error pada langkah 5?

10. TUGAS

1. Tentukan sebuah class yang merupakan turunan dari class yang lain.
2. Buat 3 atribut pada parent class kemudian tambahkan minimal 1 atribut pada child class.
3. Lakukan method overloading dengan membuat 2 constructor yaitu constructor tanpa parameter dan constructor berparameter pada masing-masing class. Panggil constructor super() berparameter untuk membuat object dari parent class pada constructor child class.
4. Lakukan method overriding dengan membuat method dengan nama dan signature yang sama pada parent class dan child class.
5. Lakukan instansiasi 2 objek child class pada main class kemudian print info nya.