MODUL 7

OVERLOADING DAN OVERRIDING

1. Kompetensi

Setelah menempuh pokok bahasan ini, mahasiswa mampu :

a. Memahami konsep overloading dan overriding,
b. Memahami perbedaan overloading dan overriding,
c. Ketepatan dalam mengidentifikasi method overriding dan overloading
d. Ketepatan dalam mempraktekkan instruksi pada modul
e. Mengimplementasikan method overloading dan overriding.

2. Pendahuluan

2.1 Overloading

Overloading adalah menuliskan kembali method dengan nama yang sama pada suatu class. Tujuannya dapat memudahkan penggunaan/pemanggilan method dengan fungsionalitas yang mirip.

Untuk aturan pendeklarasian method Overloading sebagai berikut:

- Nama method harus sama.
- Daftar parameter harus berbeda.
- Return type boleh sama, juga boleh berbeda.

Ada beberapa daftar parameter pada overloading dapat dilihat sebagai berikut:

- Perbedaan daftar parameter bukan hanya terjadi pada perbedaan banyaknya parameter, tetapi juga urutan darai parameter tersebut.
- Misalnya saja dua buah parameter berikut ini:
  - Function_member (int x, string n)
  - Function_member (String n, int x)
- Dua parameter tersebut juga di anggap berbeda daftar parameternya.
- Daftar parameter tidak terkait dengan penamaan variabel yang ada dalam parameter.

- Misalnya saja 2 daftar parameter berikut :
  - function_member(int x)
  - function_member(int y)
- Dua daftar parameter diatas dianggap sama karena yang berbeda hanya penamaan variable parameternya saja.

Overloading juga bisa terjadi antara parent class dengan subclass-nya jika memenuhi ketiga syarat overload.

2.2 Overriding

Overriding adalah Subclass yang berusaha memodifkasi tingkah laku yang diwarisi dari superclass. Tujuannya subclass dapat memiliki tingkah laku yang lebih spesifik sehingga dapat dilakukan dengan cara mendeklarasikan kembali method milik parent class di subclass.

Deklarasi method pada subclass harus sama dengan yang terdapat di super class. Kesamaan pada:

- Nama
- Return type (untuk return type : class A atau merupakan subclass dari class A)
- Daftar parameter (jumlah, tipe dan urutan)

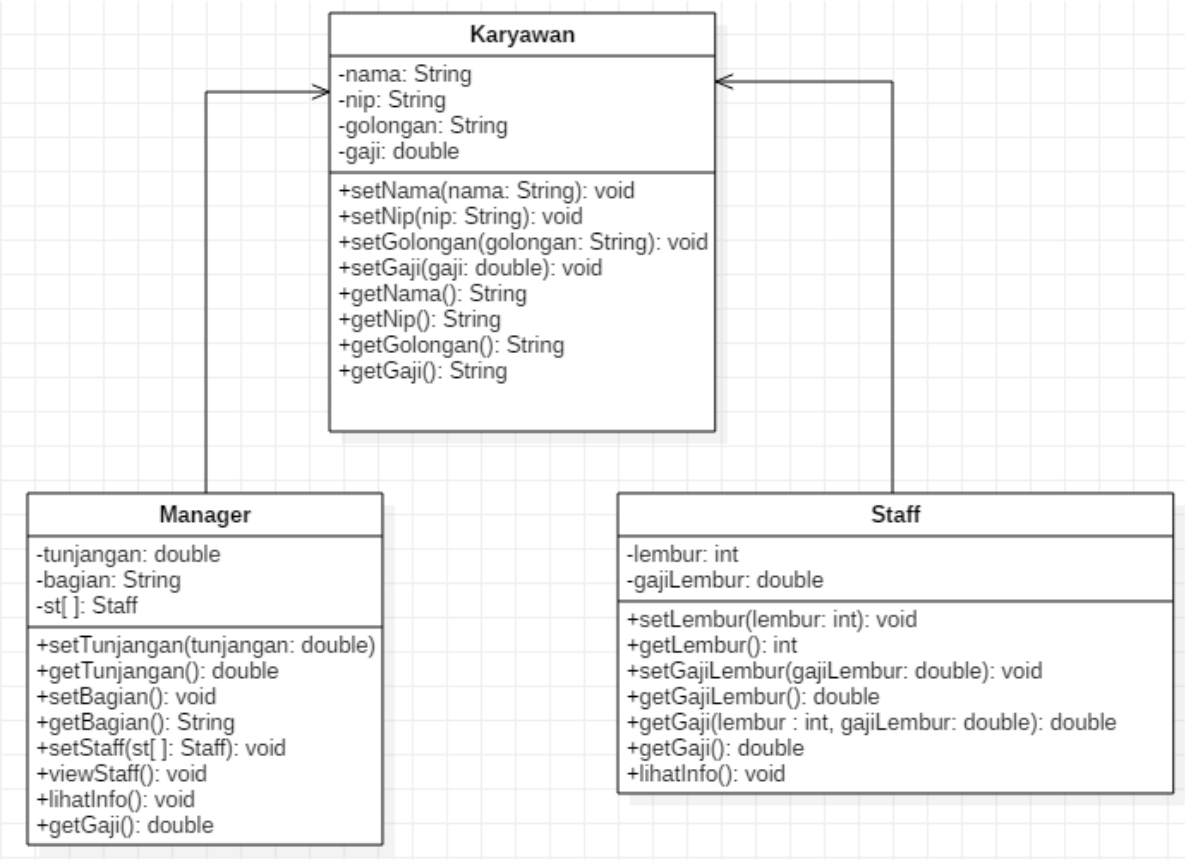
Sehingga method pada parent class disebut overridden method dan method pada subclass disebut overriding method. Ada beberapa aturan method didalam overriding:

- Mode akses overriding method harus sama atau lebih luas dari pada overridden method.
- Subclass hanya boleh meng-override method superclass satu kali saja, tidak boleh ada lebih dari satu method pada kelas yang sama persis.
- Overriding method tidak boleh throw checked exceptions yang tidak dideklarasikan oleh overridden method.

3. Praktikum

3.1 Percobaan 1

Untuk kasus contoh berikut ini, terdapat tiga kelas, yaitu Karyawan, Manager, dan Staff. Class Karyawan merupakan superclass dari Manager dan Staff dimana subclass Manager dan Staff memiliki method untuk menghitung gaji yang berbeda.



3.2 Karyawan

```
package ilkom.uniwara.modul07.percobaan1;
public class Karyawan {
    private String nama;
    private String nip;
    private String golongan;
    private double gaji;

    public void setNama(String nama){
        this.nama = nama;
    }
    public void setNip(String nip){
        this.nip =  nip;
    }
    public void setGolongan(String golongan){
        this.golongan = golongan;
        switch(golongan.charAt(0)){
            case '1':this.gaji=5000000;
              break;
            case '2':this.gaji=3000000;
              break;
            case '3':this.gaji=2000000;
              break;
            case '4':this.gaji=1000000;
              break;
            case '5':this.gaji=750000;
              break;
        }
    }

    public void setGaji(double gaji){
        this.gaji = gaji;
    }
    public String getNama(){
        return nama;
    }
    public String getNip(){
        return nip;
    }
    public String getGolongan(){
        return golongan;
    }
    public double getGaji(){
        return gaji;
    }
}
```

3.3 Staff

```
package ilkom.uniwara.modul07.percobaan1;
public class Staff extends Karyawan {
    private int lembur;
    private double gajiLembur;

    public void setLembur(int lembur){
        this.lembur = lembur;
    }
    public int getLembur(){
        return lembur;
    }
    public void setGajiLembur(double gajiLembur){
        this.gajiLembur = gajiLembur;
    }
    public double getGajiLembur(){
        return gajiLembur;
    }
    public double getGaji(int lembur, double gajiLembur){
        return super.getGaji()+lembur*gajiLembur;
    }
    public double getGaji(){
        return super.getGaji()+lembur*gajiLembur;
    }
    public void lihatInfo(){
        System.out.println("NIP         : "+this.getNip());
        System.out.println("Nama        : "+this.getNama());
        System.out.println("Golongan    : "+this.getGolongan());
        System.out.println("Jml Lembur  : "+this.getLembur());
        System.out.println("Gaji Lembur: "+this.getGajiLembur());
        System.out.println("Gaji        : "+this.getGaji());
    }
}
```

Overloading

Overriding

3.4 Manager

```
package ilkom.uniwara.modul07.percobaan1;
public class Manager extends Karyawan {
    private double tunjangan;
    private String bagian;
    private Staff st[];

    public void setTunjangan(double tunjangan){
        this.tunjangan = tunjangan;
    }
    public double getTunjangan(){
        return tunjangan;
    }
    public void setBagian(String bagian){
        this.bagian = bagian;
    }
    public String getBagian(){
        return bagian;
    }
    public void setStaff(Staff st[]){
        this.st = st;
    }
    public void viewStaff(){
        int i;
        System.out.println("-------------------------");
        for(i=0; i<st.length; i++){
            st[i].lihatInfo();
        }
        System.out.println("-------------------------");
    }

    public void lihatInfo(){
        System.out.println("Manager    : "+this.getBagian());
        System.out.println("NIP        : "+this.getNip());
        System.out.println("Nama       : "+this.getNama());
        System.out.println("Golongan   : "+this.getGolongan());
        System.out.println("Tunjangan  : "+this.getTunjangan());
        System.out.println("Gaji       : "+this.getGaji());
        System.out.println("Bagian     : "+this.getBagian());
        this.viewStaff();
    }
    public double getGaji(){
        return super.getGaji()+tunjangan;
    }
}
```

3.5 Utama

```
package ilkom.uniwara.modul07.percobaan1;
public class IlkomUniwaraModul07Percobaan1 {
    public static void main(String[] args) {
        System.out.println("Program Testing Class Manager dan Staff");
        Manager man[] = new Manager[2];
        Staff staff1[] = new Staff[2];
        Staff staff2[] = new Staff[3];

        //pembuatan manager
        man[0]=new Manager();
        man[0].setNama("Tedjo");
        man[0].setNip("101");
        man[0].setGolongan("1");
        man[0].setTunjangan(5000000);
        man[0].setBagian("Administrasi");

        man[1]=new Manager();
        man[1].setNama("Atika");
        man[1].setNip("102");
        man[1].setGolongan("1");
        man[1].setTunjangan(2500000);
        man[1].setBagian("Pemasaran");

        staff1[0]=new Staff();
        staff1[0].setNama("Usman");
        staff1[0].setNip("0003");
        staff1[0].setGolongan("2");
        staff1[0].setLembur(10);
        staff1[0].setGajiLembur(10000);
```

```
        staff1[1]=new Staff();
        staff1[1].setNama("Anugrah");
        staff1[1].setNip("0005");
        staff1[1].setGolongan("2");
        staff1[1].setLembur(10);
        staff1[1].setGajiLembur(55000);
        man[0].setStaff(staff1);

        staff2[0]=new Staff();
        staff2[0].setNama("Hendra");
        staff2[0].setNip("0004");
        staff2[0].setGolongan("3");
        staff2[0].setLembur(15);
        staff2[0].setGajiLembur(5500);

        staff2[1]=new Staff();
        staff2[1].setNama("Arie");
        staff2[1].setNip("0006");
        staff2[1].setGolongan("4");
        staff2[1].setLembur(5);
        staff2[1].setGajiLembur(100000);

        staff2[2]=new Staff();
        staff2[2].setNama("Mentari");
        staff2[2].setNip("0007");
        staff2[2].setGolongan("3");
        staff2[2].setLembur(6);
        staff2[2].setGajiLembur(20000);
        man[1].setStaff(staff2);

        //cetak informasi manager - staff
        man[0].lihatInfo();
        man[1].lihatInfo();
    }
}
```

4. Latihan

```
public class PerkalianKu {

 void perkalian(int a, int b){

  System.out.println(a * b);

 }

 void perkalian(int a, int b, int c){

  System.out.println(a * b * c);

 }

 public static void main(String args []){

  PerkalianKu objek = new PerkalianKu();

  objek.perkalian(25, 43);
  objek.perkalian(34, 23, 56);
 }
}
```

4.1 Dari source coding diatas terletak dimanakah overloading?

_____

4.2 Jika terdapat overloading ada berapa jumlah parameter yang berbeda?

_____

```
public class PerkalianKu {

 void perkalian(int a, int b){

  System.out.println(a * b);

 }

 void perkalian(double a, double b){

  System.out.println(a * b);

 }

 public static void main(String args []){

  PerkalianKu objek = new PerkalianKu();

  objek.perkalian(25, 43);
  objek.perkalian(34.56, 23.7);
 }
}
```

4.3 Dari source coding diatas terletak dimanakah overloading?

_____

4.4 Jika terdapat overloading ada berapa tipe parameter yang berbeda?

_____

```
  class Ikan{
   public void swim(){
      System.out.println("Ikan bisa berenang");
   }
}
class Piranha extends Ikan{
   public void swim(){
      System.out.println("Piranha bisa makan daging");
   }
}
public class Fish {
    public static void main(String[] args) {
        Ikan a = new Ikan();
      Ikan b = new Piranha();
      a.swim();
      b.swim();
    }
   }
```

4.5 Dari source coding diatas terletak dimanakah overriding?

_____

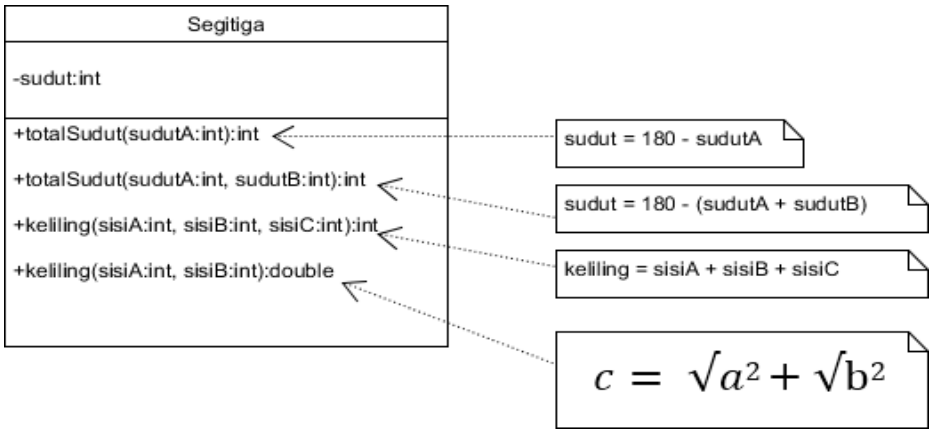
4.6 Jabarkanlah apabila sourcoding diatas jika terdapat overriding?

_____

5. Tugas

5.1 Overloading

Implementasikan konsep overloading pada class diagram dibawah ini :



5.2 Overriding

Implementasikan class diagram dibawah ini dengan konsep overriding pada pemrograman Java: