JOBSHEET 7

OVERLOADING DAN OVERRIDING

1. Kompetensi

After completing this chapter, students are able to:

a. Understand overloading and overriding concept,
b. Undertand the difference of overloading and overriding,
c. Identify overloading and overriding method correctly
d. Perform the instruction correctly
e. Implement overloading and overriding method

2. Pendahuluan

2.1 Overloading

Is a method that has the same name in the same class. The purpose if overloading is to simplify the call to a method that is similar in functionality. The rule to declare overloading method are:

- Method has the same name
- Has different parameters signature
- Different or same return type

There are several parameters in overloading:

- The difference in parameter signature of a method is not only in the amount of the parameters, but also the sequence of it.
- For example, there are two method parameters:
  - Function_member (int x, string n)
  - Function_member (String n, int x)
- Those two parameters can also be considered different parameter signature.
- Parameter signature is not related with the name of parameter.
- For example, there are two parameters :
  - function_member(int x)
  - function_member(int y)
- Those two parameters can be considered the same parameter signature

Overloading can also happened in a parent class and its subclass, if it qualify all the rule of overloading:

- Primitive widening conversion is prioritized compared to boxing and var args.
- We cannot perform widening process fro wrapper type to another wrapper type (converting Integer to Long).
- We cannot perform widening process and then proceed to boxing process (from int to Long).
- We can perform boxing and then widening (int can be converted to Object via Integer wrapper).
- We can combine var args with either widening or boxing.

2.2 Overriding

Is a subclass performing to modify the behaviour of its superclass. The purpose for it is to allow specific behaviour of a subclass by re-declare existing method in its parent class. Method declaration in subclass must be same with the one in the parent class. The similarity must be applied to:

- Method name
- Return type
- Amount of parameter (amount, type and sequence)

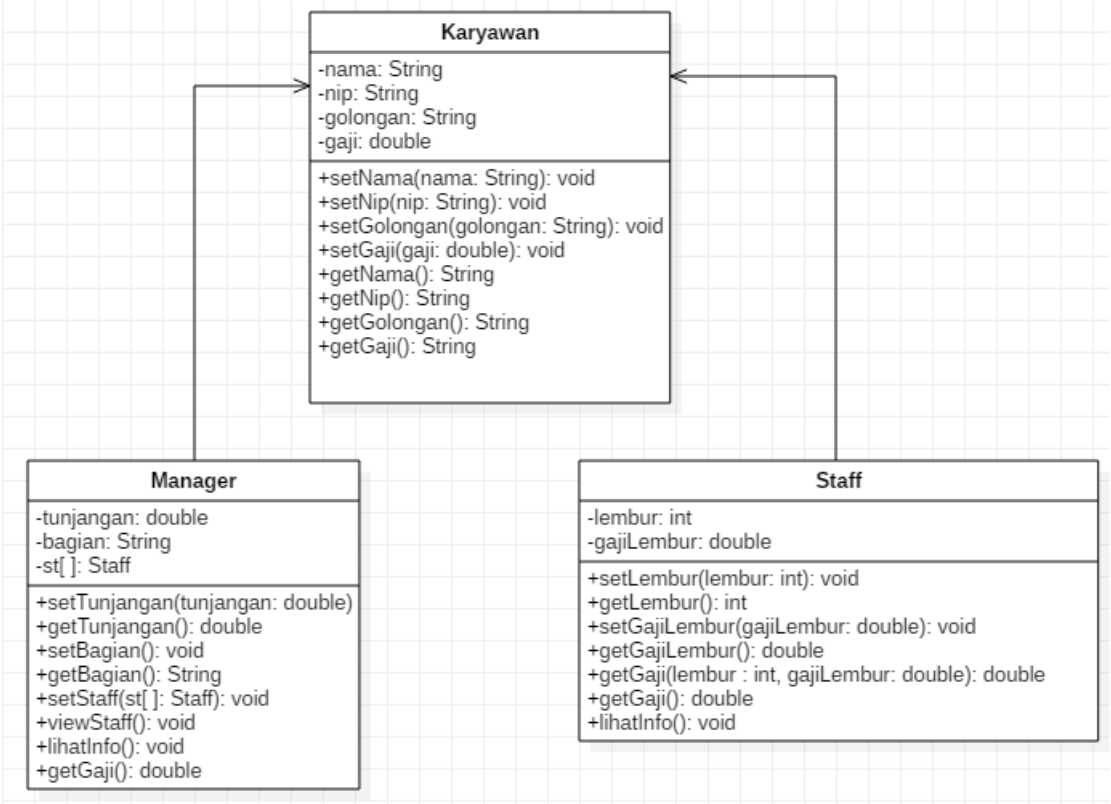
Therefore, method in the parent class is called overridden method, and in the subclass called overriding method. There are several rules of overriding:

- Access modifier in overriding method must be the same or wider than overridden method.
- Subclass can only override a method once.
- Overriding method cannot throw checked exceptions that is not declared by the overridden method

3. Praktikum

3.1 Percobaan 1

For this experiment case, there are three classes, Karyawan, Manager and Staff. Karyawan class is a superclass of Manager and Staff in which subclass Manager and Staff each has different method to calculate different



Continue to the next page.

3.2 Karyawan

```
public class Karyawan {

    /**
     * @param args the command line arguments
     */
  //  public static void main(String[] args) {
        // TODO code application logic here
 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

```
 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.printf("Gaji Lembur :%.0f\n", this.getGajiLembur());
  System.out.printf("Gaji  :%.0f\n",this.getGaji());
 }
}
```

Overloading

Overriding

3.4 Manager

```
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.printf("Tunjangan :%.0f\n",this.getTunjangan());
 System.out.printf("Gaji  :%.0f\n",this.getGaji());
 System.out.println("Bagian  :"+this.getBagian());
 this.viewStaff();
}
public double getGaji()
{
 return super.getGaji()+tunjangan;
}
}
```

3.5 Utama

```
public class Utama {
public static void main(String[] args)
{
 System.out.println("Program Testing Class Manager & 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 dari manager + staffnya
man[0].lihatInfo();
man[1].lihatInfo();
}
```

4. Exercise

```
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 From the above source code, where is the overloading?

_____

4.2 If there any overloading, how many parameters are different?

_____

```java
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 From the above source code, where is the overloading?

4.4 If there any overloading, how many parameters are different?

```java
  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 From the above source code, where is the overloading?

_____

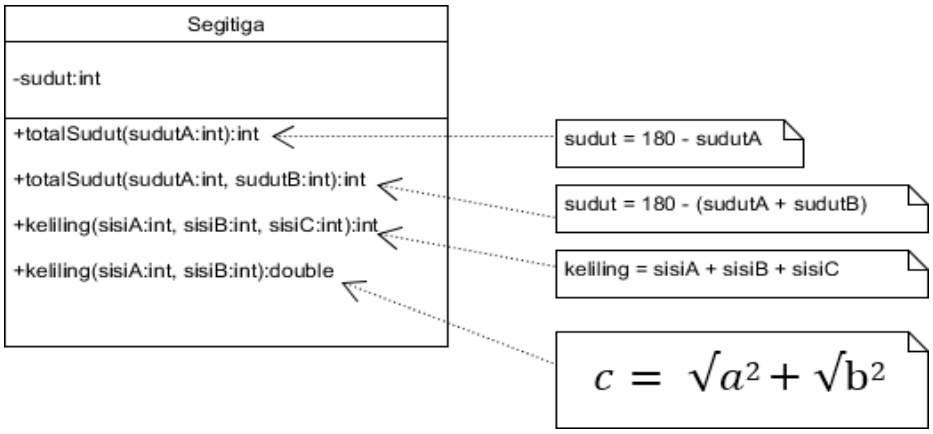
4.6 If there any overloading, how many parameters are different?

_____

5. Task

5.1 Overloading

Implement overloading concept into this class diagram:



5.2 Overriding

Implement overriding for these class using dynamic method dispatch :