JOBSHEET 7

OVERLOADING AND OVERRIDING

1. Competence

After taking this subject, students are able to:

a. Understand the concepts of overloading and overriding,
b. Understand the difference between overloading and overriding,
c. Accuracy in identifying overriding and overloading methods
d. Accuracy in practicing instructions on the jobsheet
e. Implement overloading and overriding methods.

2. Introduction

2.1 Overloading

is to rewrite a method with the same name on a class. The goal is to facilitate the use/invocation of methods with similar functionality. The Overloading method declaration rules are as follows:

- The method name must be the same.
- The list of parameters should be different.
- The return type can be the same, or it can be different.

There are several lists of parameters on overloading can be seen as follows:

- The difference in the list of parameters does not only occur in the difference in the number of parameters, but also in the order of the parameters.
- For example, the following two parameters:
  - Function_member (int x, string n)
  - Function_member (String n, int x)
- The two parameters are also considered different in the list of parameters.
- The parameter list is not related to the naming of the variables present in the parameter.
- For example, the following 2 list of parameters:
  - function_member(int x)
  - function_member(int y)

- The two lists of parameters above are considered the same because the only difference is the naming of the variable parameters.

Overloading can also occur between the parent class and its subclass if it meets all three overload conditions. There are several overloading rules, namely:

- Primitive widening conversions take precedence over overloading over boxing and var args.
- We can't do the widening process from one wrapper type to another (changing the Integer to Long).
- We can't do the widening process followed by boxing (from int to Long)
- We can do boxing followed by widening (int can be an Object via an Integer)
- We can combine var args with either widening or boxing

2.2 Overriding

is a Subclass that seeks to modify behaviors inherited from super classes. The goal is that the subclass can have more specific behavior so that it can be done by redeclaring the parent class's method in the subclass.

The method declaration in the subclass must be the same as the one in the super class. Similarities on:

- Name
- Return type (for return type: class A or is a subclass of class A)
- List of parameters (number, type and order)

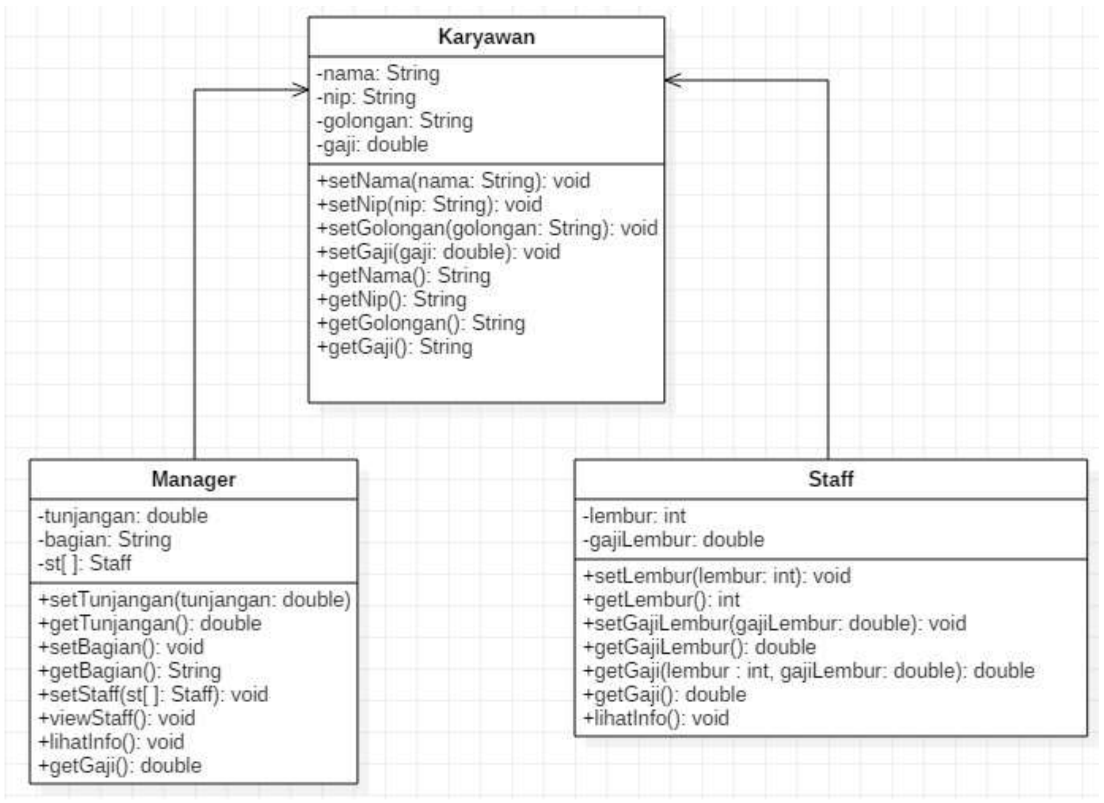
So that the method in the parent class is called the overridden method and the method in the subclass is called the overriding method. There are several method rules in overriding:

- The access mode of the overriding method must be the same or broader than the overridden method.
- A subclass can only override a superclass method once, there must not be more than one method in the exact same class.
- The overriding method must not throw checked exceptions that are not declared by the overridden method.

3. Practicum

3.1 Experiment 1

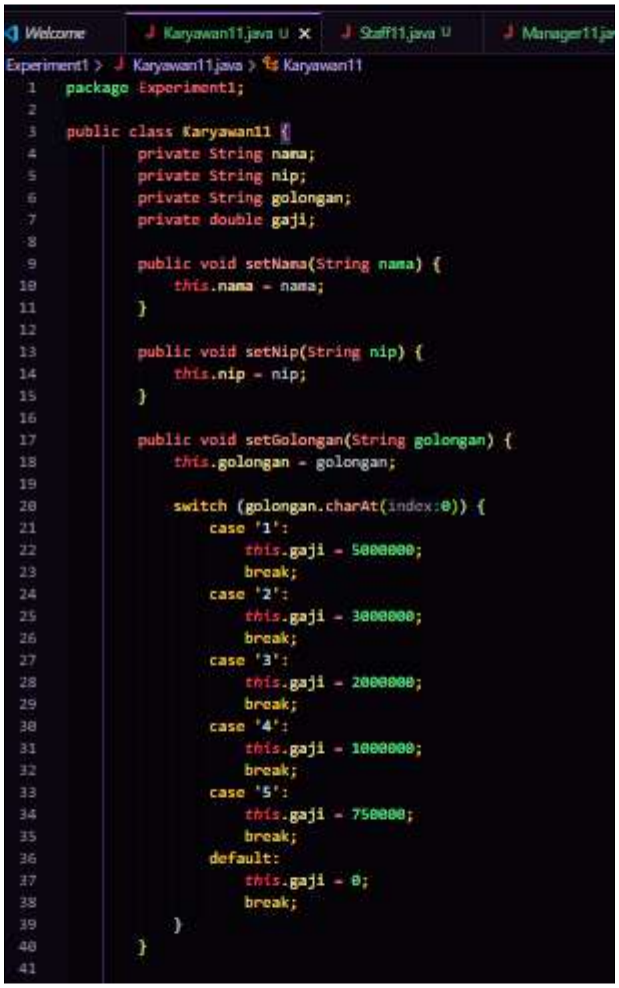
For the following example case, there are three classes, namely Karyawan, Manager, and Staff. Employee Class is a superclass of Manager and Staff where the Manager and Staff subclasses have different methods for calculating salaries.



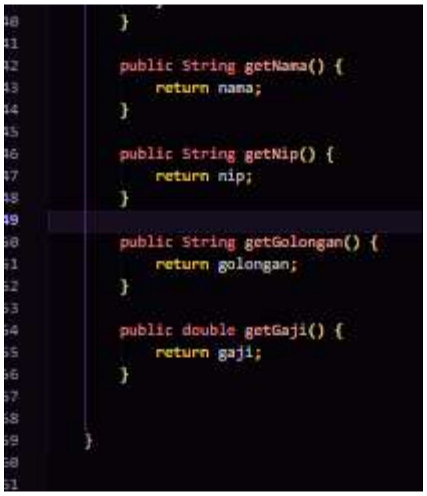
3.2 Karyawan

3.3

```java
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;
  }
```



```java
package Experiment1;

public class Karyawan11 {
        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(index: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;
                default:
                    this.gaji = 0;
                    break;
            }
        }
```



```java
        }

        public String getNama() {
            return nama;
        }

        public String getNip() {
            return nip;
        }

        public String getGolongan() {
            return golongan;
        }

        public double getGaji() {
            return gaji;
        }


    }
```

```
 public double getGaji()
 {
  return gaji;
 }
}
```

3.4 Staff

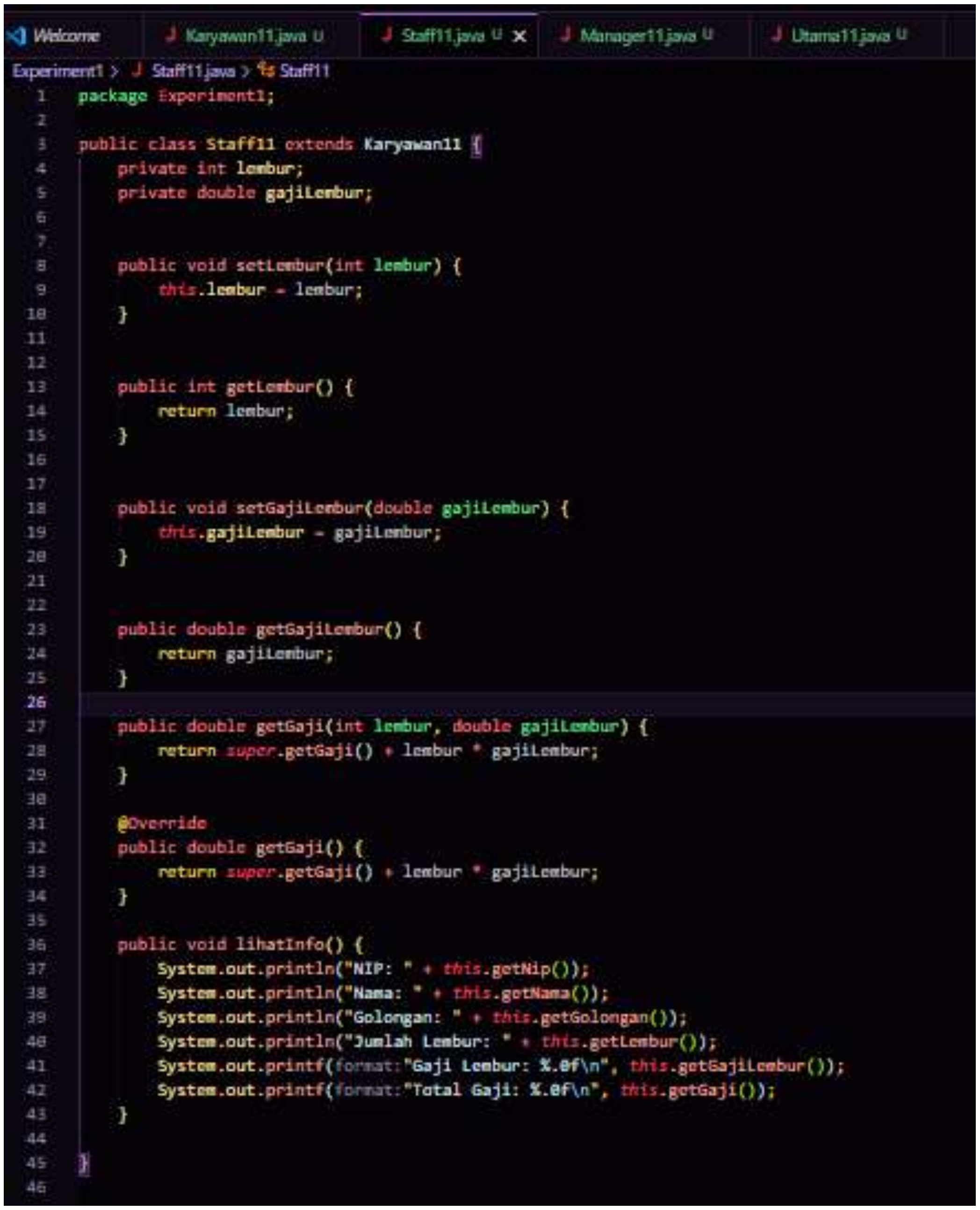
```
 public class Staff extends Karyawan {
private int lembur;
private double gajiLembur;

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;
}
```

Overloading

```
 public double getGaji()
 {
  return super.getGaji()+lembur*gajiLembur;
 }
```

Overriding

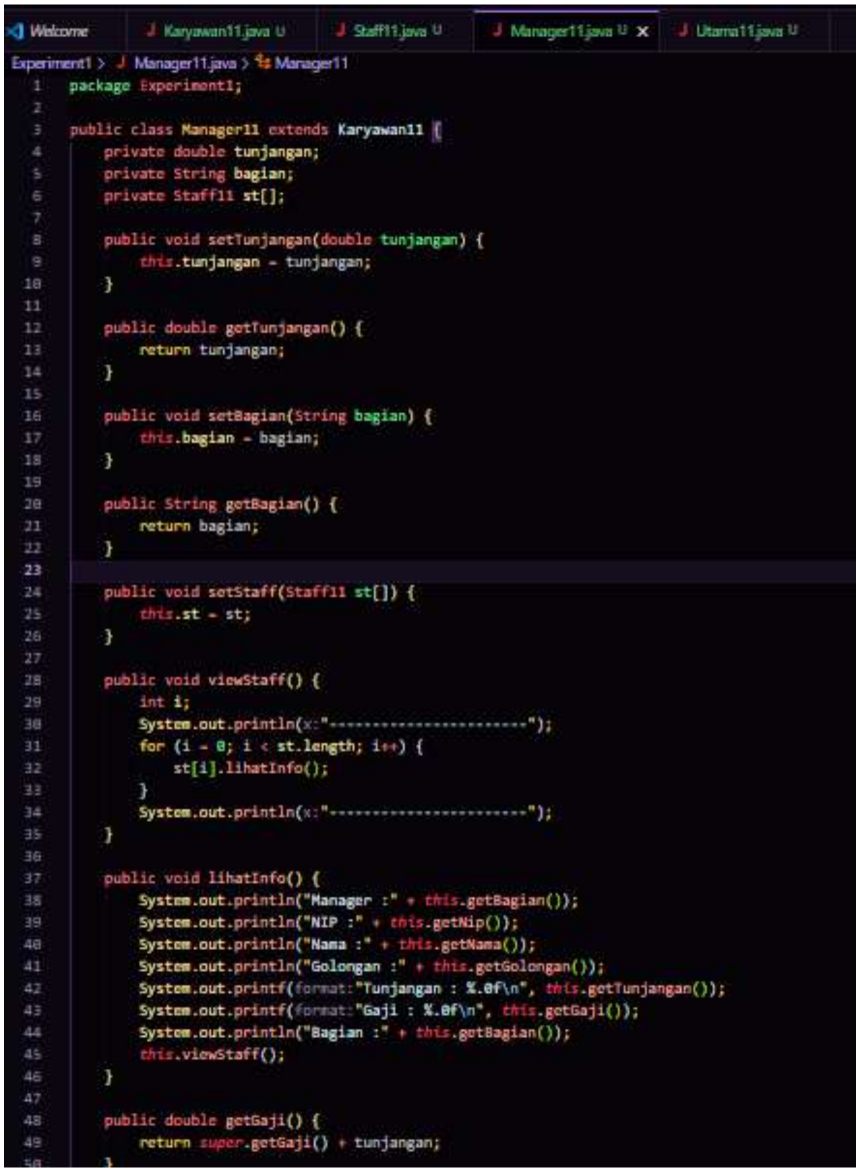
```
 public void lihatInfo()
 {
  System.out.println("NIP  :"+this.getNip());
  System.out.println("Nama  :"+this.getNama());
  System.out.println("Golongan :"+this.getGolongan());
  System.out.println("Jml Lembur :"+this.getLembur());
  System.out.printf("Gaji Lembur :%.0f\n", this.getGajiLembur());
  System.out.printf("Gaji  :%.0f\n",this.getGaji());
 }
}
```



```java
package Experiment1;

public class Staff11 extends Karyawan11 {
    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;
    }

    @Override
    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("Jumlah Lembur: " + this.getLembur());
        System.out.printf(format:"Gaji Lembur: %.0f\n", this.getGajiLembur());
        System.out.printf(format:"Total Gaji: %.0f\n", this.getGaji());
    }

}
```

3.5 Manager

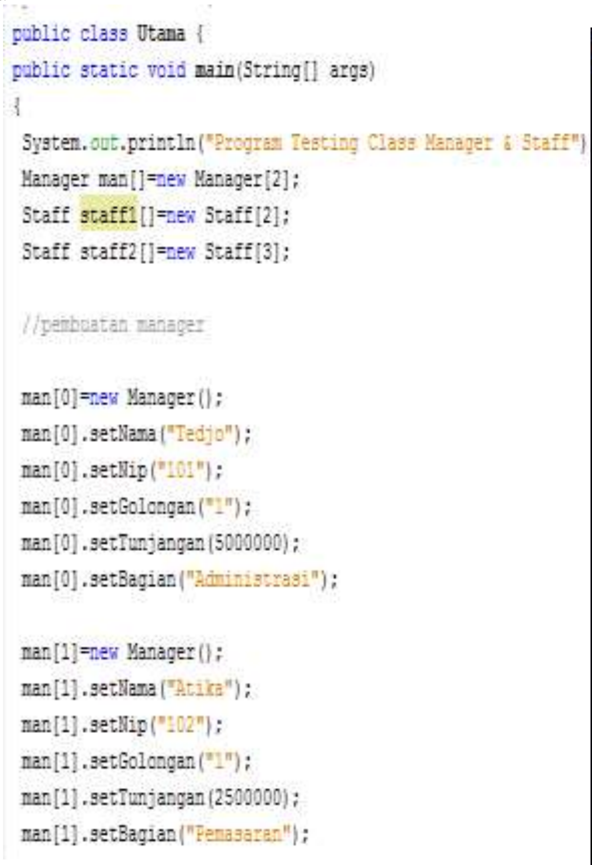
```java
public class Manager extends Karyawan {
private double tunjangan;
private String bagian;
private Staff st[];

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;
}
}
```



```java
package Experiment1;

public class Manager11 extends Karyawan11 {
    private double tunjangan;
    private String bagian;
    private Staff11 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(Staff11 st[]) {
        this.st = st;
    }

    public void viewStaff() {
        int i;
        System.out.println(x:"----------------------");
        for (i = 0; i < st.length; i++) {
            st[i].lihatInfo();
        }
        System.out.println(x:"----------------------");
    }

    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(format:"Tunjangan : %.0f\n", this.getTunjangan());
        System.out.printf(format:"Gaji : %.0f\n", this.getGaji());
        System.out.println("Bagian :" + this.getBagian());
        this.viewStaff();
    }

    public double getGaji() {
        return super.getGaji() + tunjangan;
    }
```

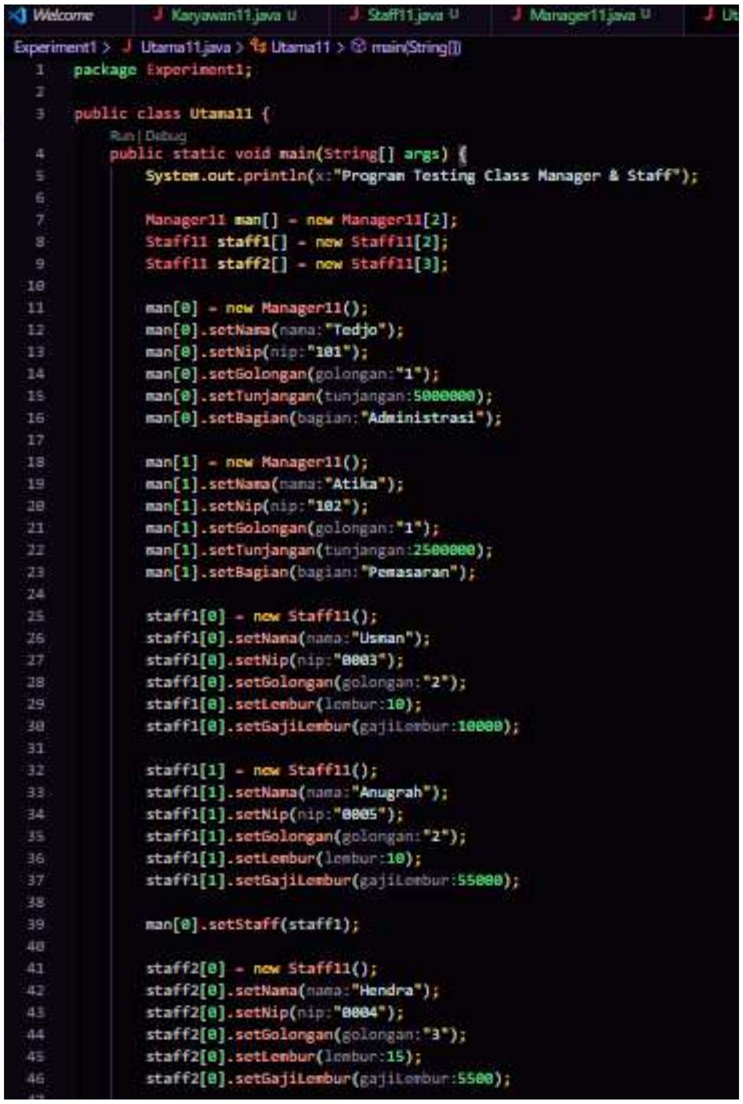
3.6 Main

3.7



```
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");
```

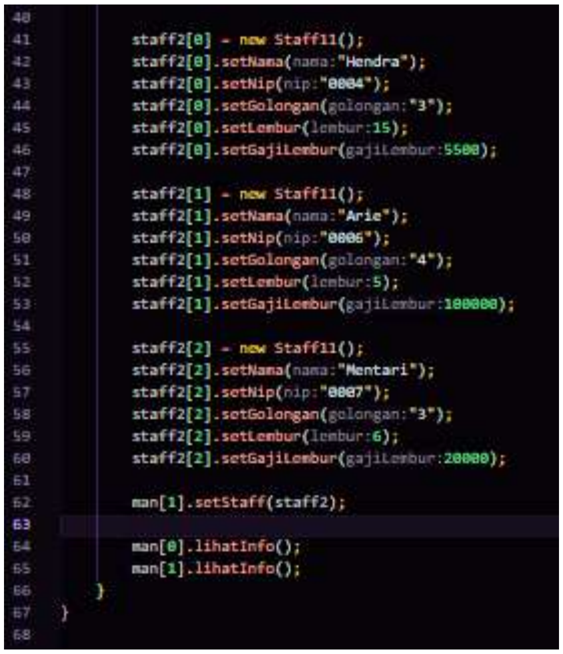
```
 staff1[0]=new Staff();
 staff1[0].setNama("Usman");
 staff1[0].setNip("0003");
 staff1[0].setGolongan("2");
 staff1[0].setLembur(10);
 staff1[0].setGajiLembur(10000);

 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);
```



Welcome | Karyawan11.java U | Staff11.java U | Manager11.java U | Ut

Experiment1 > Utama11.java > Utama11 > main(String[])

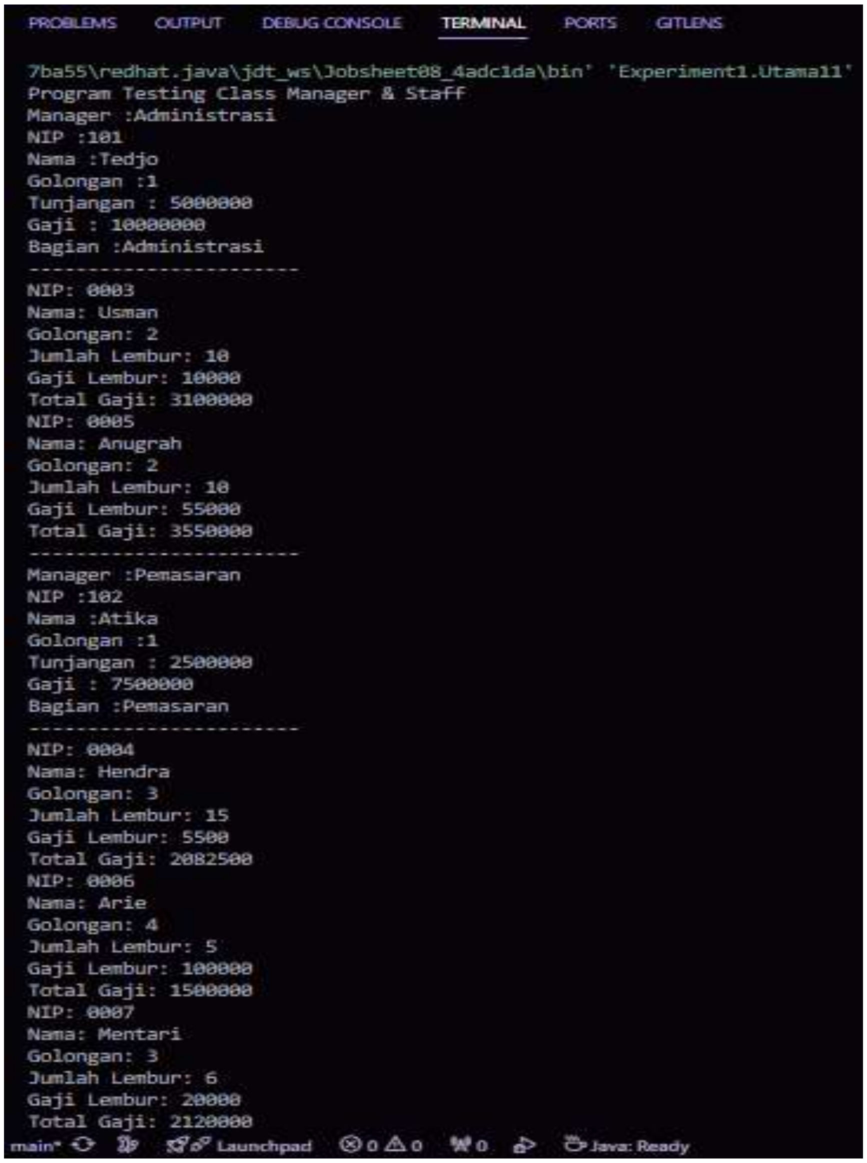
```
package Experiment1;

public class Utama11 {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"Program Testing Class Manager & Staff");

        Manager11 man[] = new Manager11[2];
        Staff11 staff1[] = new Staff11[2];
        Staff11 staff2[] = new Staff11[3];

        man[0] = new Manager11();
        man[0].setNama(nama:"Tedjo");
        man[0].setNip(nip:"101");
        man[0].setGolongan(golongan:"1");
        man[0].setTunjangan(tunjangan:5000000);
        man[0].setBagian(bagian:"Administrasi");

        man[1] = new Manager11();
        man[1].setNama(nama:"Atika");
        man[1].setNip(nip:"102");
        man[1].setGolongan(golongan:"1");
        man[1].setTunjangan(tunjangan:2500000);
        man[1].setBagian(bagian:"Pemasaran");

        staff1[0] = new Staff11();
        staff1[0].setNama(nama:"Usman");
        staff1[0].setNip(nip:"0003");
        staff1[0].setGolongan(golongan:"2");
        staff1[0].setLembur(lembur:10);
        staff1[0].setGajiLembur(gajiLembur:10000);

        staff1[1] = new Staff11();
        staff1[1].setNama(nama:"Anugrah");
        staff1[1].setNip(nip:"0005");
        staff1[1].setGolongan(golongan:"2");
        staff1[1].setLembur(lembur:10);
        staff1[1].setGajiLembur(gajiLembur:55000);

        man[0].setStaff(staff1);

        staff2[0] = new Staff11();
        staff2[0].setNama(nama:"Hendra");
        staff2[0].setNip(nip:"0004");
        staff2[0].setGolongan(golongan:"3");
        staff2[0].setLembur(lembur:15);
        staff2[0].setGajiLembur(gajiLembur:5500);
```



```

        staff2[0] = new Staff11();
        staff2[0].setNama(nama:"Hendra");
        staff2[0].setNip(nip:"0004");
        staff2[0].setGolongan(golongan:"3");
        staff2[0].setLembur(lembur:15);
        staff2[0].setGajiLembur(gajiLembur:5500);

        staff2[1] = new Staff11();
        staff2[1].setNama(nama:"Arie");
        staff2[1].setNip(nip:"0006");
        staff2[1].setGolongan(golongan:"4");
        staff2[1].setLembur(lembur:5);
        staff2[1].setGajiLembur(gajiLembur:100000);

        staff2[2] = new Staff11();
        staff2[2].setNama(nama:"Mentari");
        staff2[2].setNip(nip:"0007");
        staff2[2].setGolongan(golongan:"3");
        staff2[2].setLembur(lembur:6);
        staff2[2].setGajiLembur(gajiLembur:20000);

        man[1].setStaff(staff2);

        man[0].lihatInfo();
        man[1].lihatInfo();
    }
}

```

```
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();
}
```

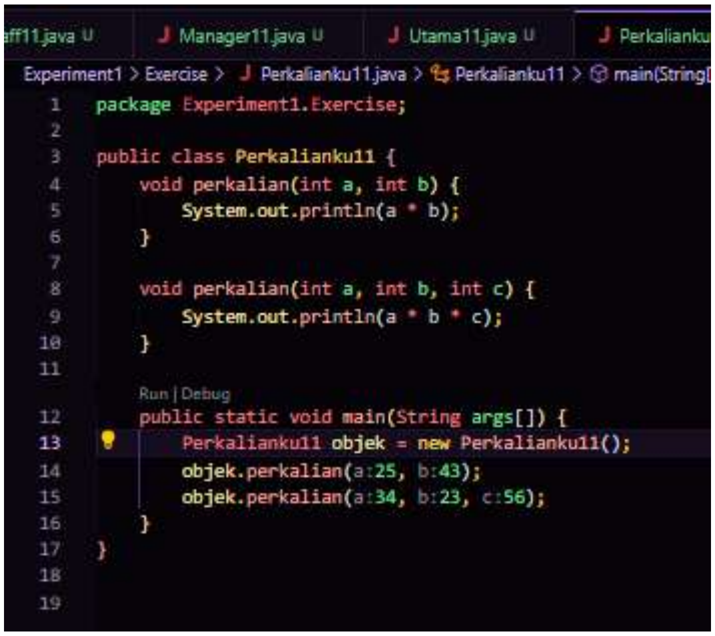
Hasil Run :



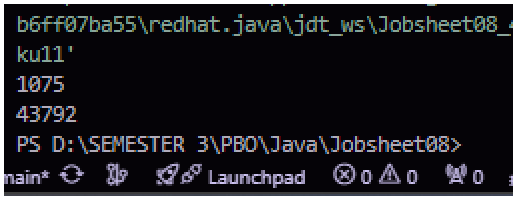
```
7ba55\redhat.java\jdt_ws\Jobsheet08_4adc1da\bin' 'Experiment1.Utama11'
Program Testing Class Manager & Staff
Manager :Administrasi
NIP :101
Nama :Tedjo
Golongan :1
Tunjangan : 5000000
Gaji : 10000000
Bagian :Administrasi
-----------------------
NIP: 0003
Nama: Usman
Golongan: 2
Jumlah Lembur: 10
Gaji Lembur: 10000
Total Gaji: 3100000
NIP: 0005
Nama: Anugrah
Golongan: 2
Jumlah Lembur: 10
Gaji Lembur: 55000
Total Gaji: 3550000
-----------------------
Manager :Pemasaran
NIP :102
Nama :Atika
Golongan :1
Tunjangan : 2500000
Gaji : 7500000
Bagian :Pemasaran
-----------------------
NIP: 0004
Nama: Hendra
Golongan: 3
Jumlah Lembur: 15
Gaji Lembur: 5500
Total Gaji: 2082500
NIP: 0006
Nama: Arie
Golongan: 4
Jumlah Lembur: 5
Gaji Lembur: 100000
Total Gaji: 1500000
NIP: 0007
Nama: Mentari
Golongan: 3
Jumlah Lembur: 6
Gaji Lembur: 20000
Total Gaji: 2120000
```

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);
 }
}
```


ff11.java U
Manager11.java U
Utama11.java U
Perkalianku
Experiment1 > Exercise > Perkalianku11.java > Perkalianku11 > main(String
Run | Debug

```
package Experiment1.Exercise;

public class Perkalianku11 {
    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[]) {
        Perkalianku11 objek = new Perkalianku11();
        objek.perkalian(a:25, b:43);
        objek.perkalian(a:34, b:23, c:56);
    }
}
```



```
b6ff07ba55\redhat.java\jdt_ws\Jobsheet08_
ku11'
1075
43792
PS D:\SEMESTER 3\PBO\Java\Jobsheet08>
```

4.1 From the source coding above, where is the overloading?

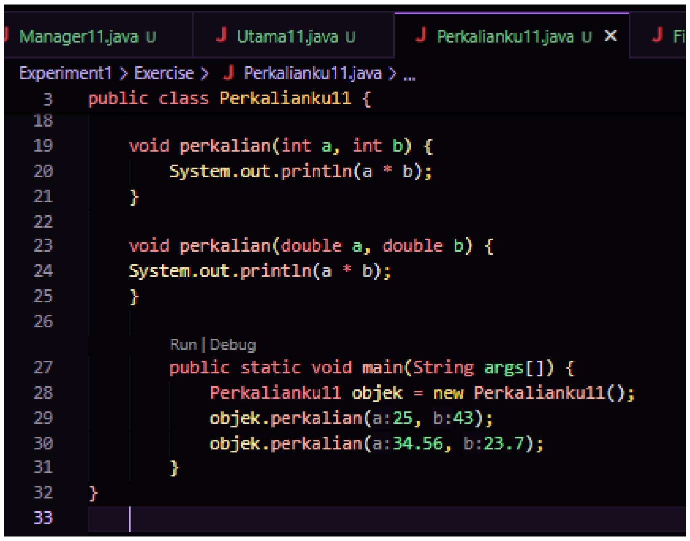
Overloading occurs in the multiplication method

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

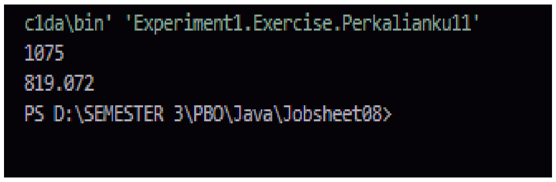
There are two ways to create overloading:

- **changing the number of parameters.**
- **changing the type of parameters, even if the number is the same.**

```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);
 }
}
```



```java
public class Perkalianku11 {

    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[]) {
            Perkalianku11 objek = new Perkalianku11();
            objek.perkalian(a:25, b:43);
            objek.perkalian(a:34.56, b:23.7);
        }
}
```



```
c1da\bin' 'Experiment1.Exercise.Perkalianku11'
1075
819.072
PS D:\SEMESTER 3\PBO\Java\Jobsheet08>
```

4.3 From the source coding above, where is the overloading?

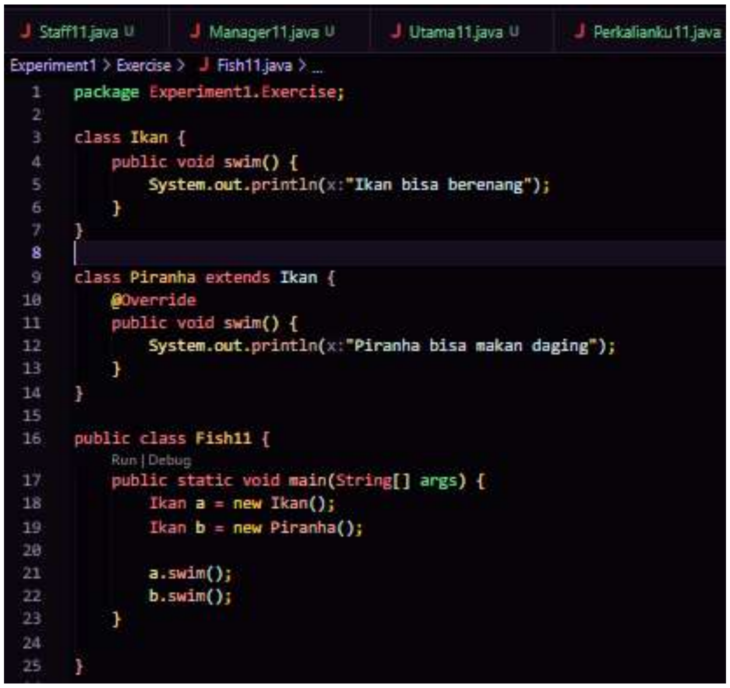
Overloading occurs in the multiplication method

4.4 If there is overloading, how many different types of parameters are there?

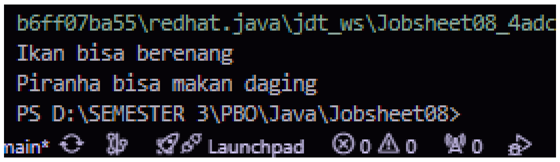
There are two multiplication methods that demonstrate method overloading :

- **There are two different parameter types yaitu int and double**.

```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();
    }
   }
```



```java
package Experiment1.Exercise;

class Ikan {
    public void swim() {
        System.out.println(x:"Ikan bisa berenang");
    }
}

class Piranha extends Ikan {
    @Override
    public void swim() {
        System.out.println(x:"Piranha bisa makan daging");
    }
}

public class Fish11 {
    public static void main(String[] args) {
        Ikan a = new Ikan();
        Ikan b = new Piranha();

        a.swim();
        b.swim();
    }

}
```



```
b6ff07ba55\redhat.java\jdt_ws\Jobsheet08_4adc
Ikan bisa berenang
Piranha bisa makan daging
PS D:\SEMESTER 3\PBO\Java\Jobsheet08>
```

4.5 From the source coding above, where is the overriding?

The overriding lies in the Piranha class, where the swim() method of the Fish class is modified to print a different message.

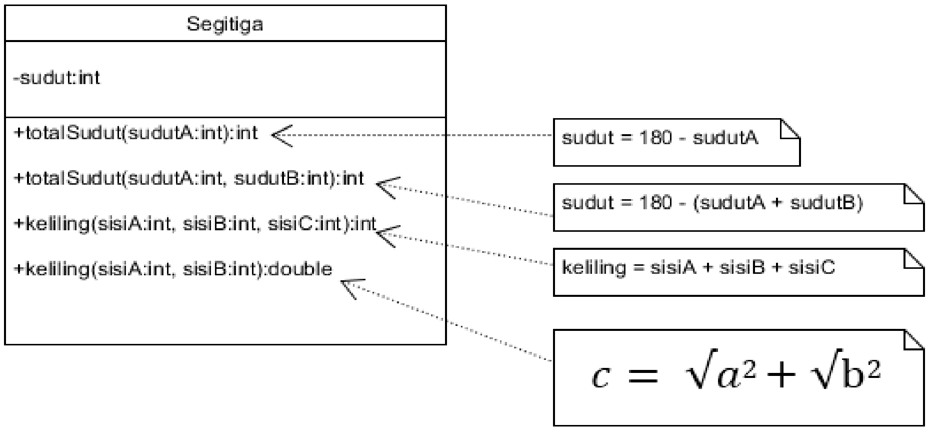
4.6 Describe when sourcoding above if there is overriding?

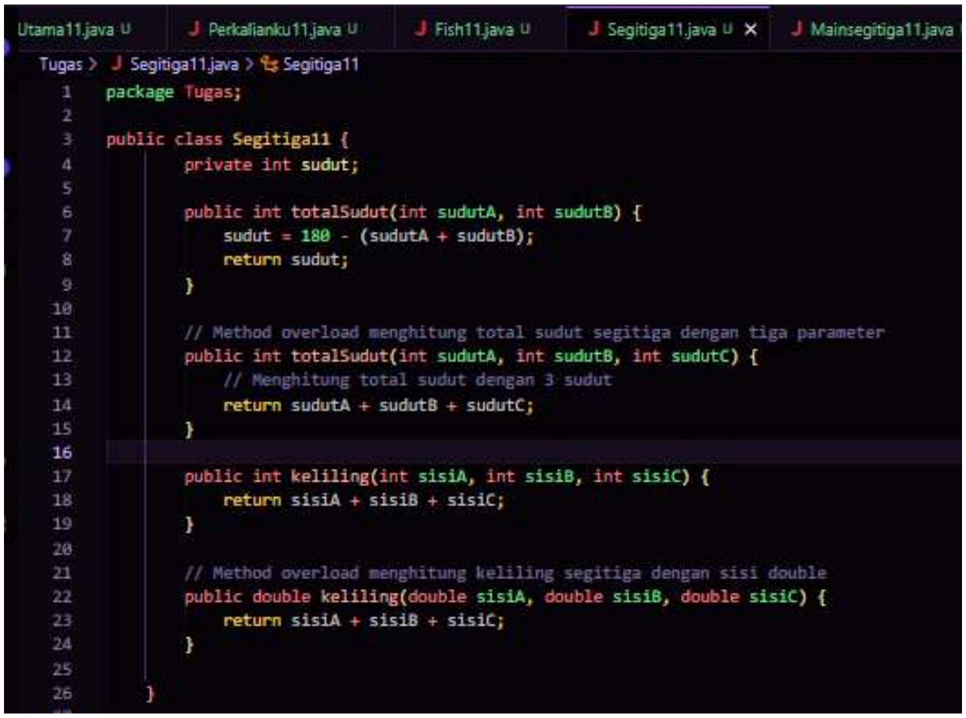
Fish is a superclass with a swim() method
Piranha is a subclass that overrides the swim() method
Overriding occurs when a subclass provides a new implementation for a method inherited from the superclass. Piranha changes the swim() behavior of Fish, demonstrating the power of overriding in Java.

5. Tasks

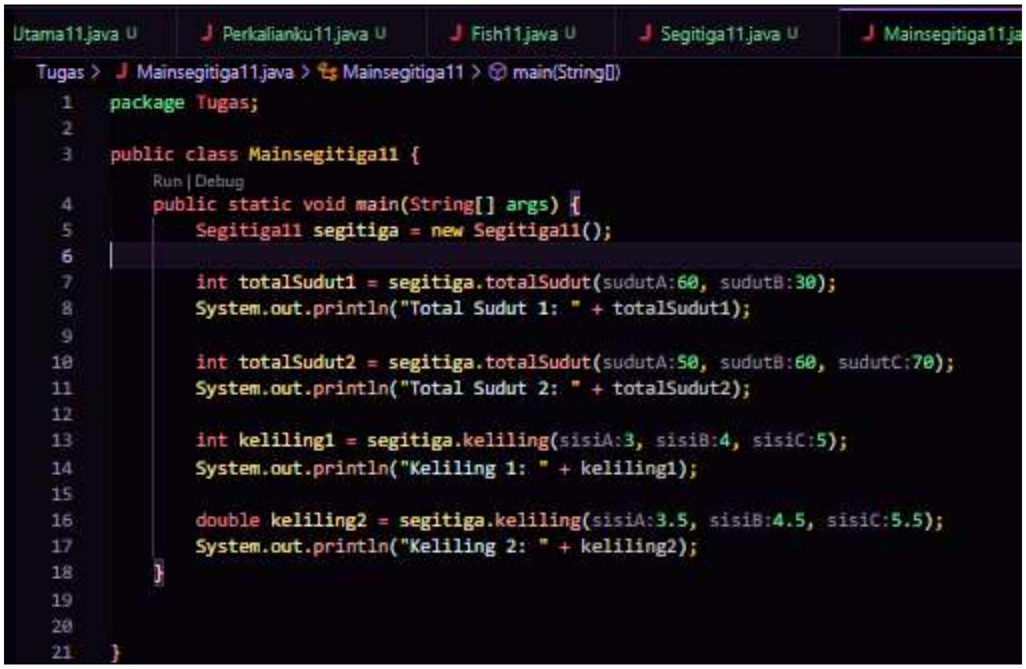
5.1 Overloading

Implement the overloading concept in the diagram class below:

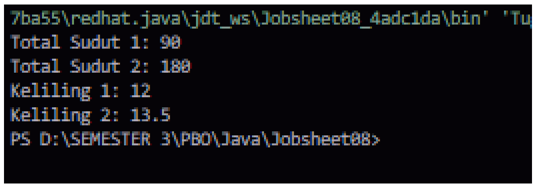

Segitiga
-sudut:int
+totalSudut(sudutA:int):int
+totalSudut(sudutA:int, sudutB:int):int
+keliling(sisiA:int, sisiB:int, sisiC:int):int
+keliling(sisiA:int, sisiB:int):double
sudut = 180 - sudutA
sudut = 180 - (sudutA + sudutB)
keliling = sisiA + sisiB + sisiC
$c = \sqrt{a^2} + \sqrt{b^2}$



```java
package Tugas;

public class Segitiga11 {
        private int sudut;

        public int totalSudut(int sudutA, int sudutB) {
            sudut = 180 - (sudutA + sudutB);
            return sudut;
        }

        // Method overload menghitung total sudut segitiga dengan tiga parameter
        public int totalSudut(int sudutA, int sudutB, int sudutC) {
            // Menghitung total sudut dengan 3 sudut
            return sudutA + sudutB + sudutC;
        }

        public int keliling(int sisiA, int sisiB, int sisiC) {
            return sisiA + sisiB + sisiC;
        }

        // Method overload menghitung keliling segitiga dengan sisi double
        public double keliling(double sisiA, double sisiB, double sisiC) {
            return sisiA + sisiB + sisiC;
        }

    }
```



```java
package Tugas;

public class Mainsegitiga11 {
    public static void main(String[] args) {
        Segitiga11 segitiga = new Segitiga11();

        int totalSudut1 = segitiga.totalSudut(sudutA:60, sudutB:30);
        System.out.println("Total Sudut 1: " + totalSudut1);

        int totalSudut2 = segitiga.totalSudut(sudutA:50, sudutB:60, sudutC:70);
        System.out.println("Total Sudut 2: " + totalSudut2);

        int keliling1 = segitiga.keliling(sisiA:3, sisiB:4, sisiC:5);
        System.out.println("Keliling 1: " + keliling1);

        double keliling2 = segitiga.keliling(sisiA:3.5, sisiB:4.5, sisiC:5.5);
        System.out.println("Keliling 2: " + keliling2);
    }


}
```



```
7ba55\redhat.java\jdt_ws\Jobsheet08_4adc1da\bin' 'Tu
Total Sudut 1: 90
Total Sudut 2: 180
Keliling 1: 12
Keliling 2: 13.5
PS D:\SEMESTER 3\PBO\Java\Jobsheet08>
```

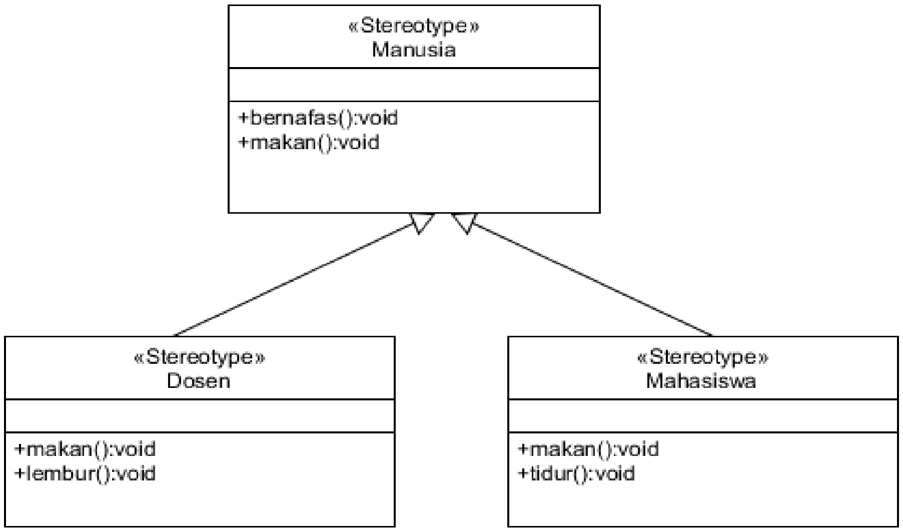
Attributes: The Triangle class has angle attributes.

Methods:

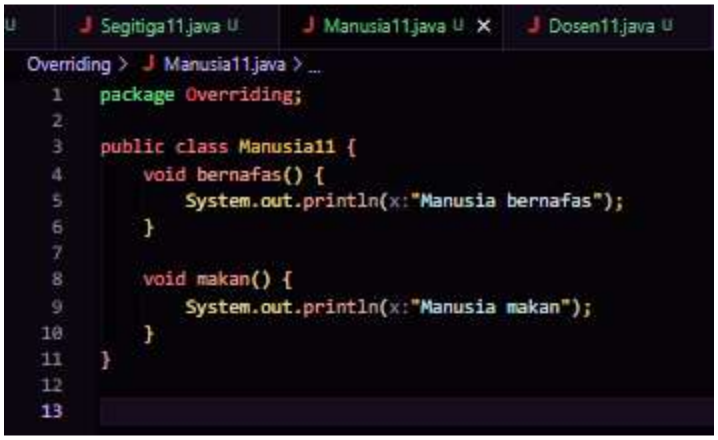
- **totalAngles(int angleA, int angleB): Calculates the third angle of a triangle.**
- **totalAngles(int angleA, int angleB, int angleC): Calculates the total angle of a triangle with three angles (overload).**
- **perimeter(int sideA, int sideB, int sideC): Calculates the circumference of a triangle with rounded sides.**
- **perimeter(double sideA, double sideB, double sideC): Calculates the circumference of a triangle with double sides (overload).**

5.2 Overriding

Implement the diagram class below using the dynamic method dispatch technique:

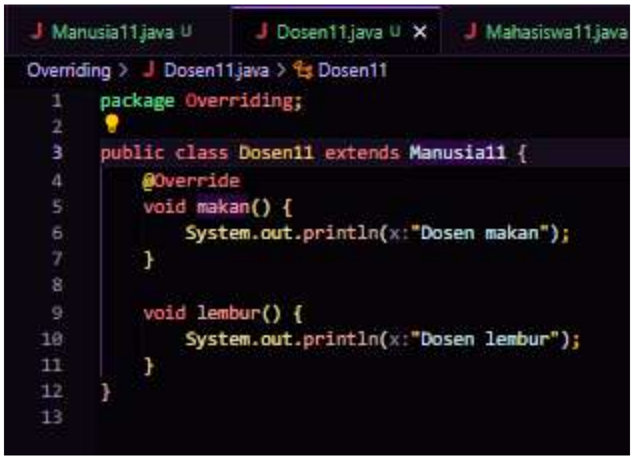


1. Class Human



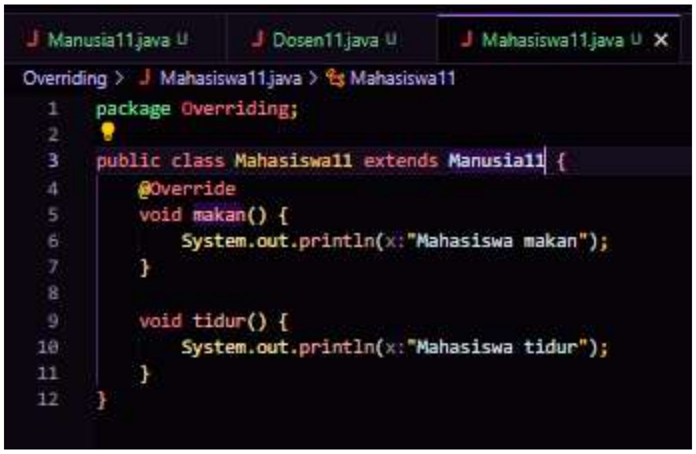
```
package Overriding;

public class Manusia11 {
    void bernafas() {
        System.out.println(x:"Manusia bernafas");
    }

    void makan() {
        System.out.println(x:"Manusia makan");
    }
}

```

2. Class lecturer



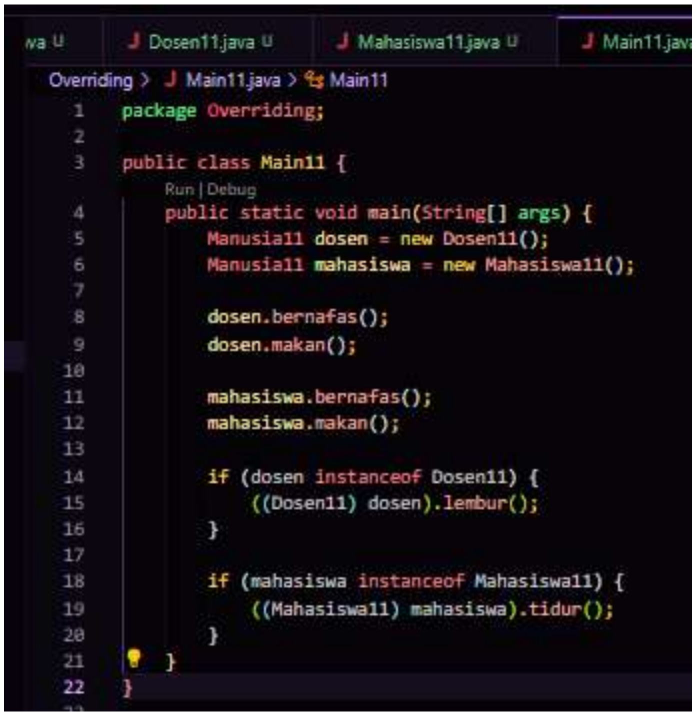
```
package Overriding;

public class Dosen11 extends Manusia11 {
    @Override
    void makan() {
        System.out.println(x:"Dosen makan");
    }

    void lembur() {
        System.out.println(x:"Dosen lembur");
    }
}

```

3. Class student



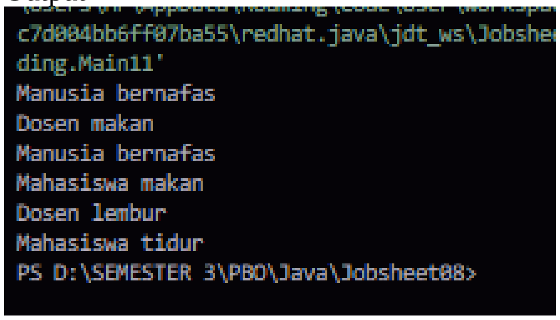
```java
package Overriding;

public class Mahasiswa11 extends Manusia11 {
    @Override
    void makan() {
        System.out.println(x:"Mahasiswa makan");
    }

    void tidur() {
        System.out.println(x:"Mahasiswa tidur");
    }
}
```

4. Main



```java
package Overriding;

public class Main11 {
    public static void main(String[] args) {
        Manusia11 dosen = new Dosen11();
        Manusia11 mahasiswa = new Mahasiswa11();

        dosen.bernafas();
        dosen.makan();

        mahasiswa.bernafas();
        mahasiswa.makan();

        if (dosen instanceof Dosen11) {
            ((Dosen11) dosen).lembur();
        }

        if (mahasiswa instanceof Mahasiswa11) {
            ((Mahasiswa11) mahasiswa).tidur();
        }
    }
}
```

5. Output



```
c7d004bb6ff07ba55\redhat.java\jdt_ws\Jobshe
ding.Main11'
Manusia bernafas
Dosen makan
Manusia bernafas
Mahasiswa makan
Dosen lembur
Mahasiswa tidur
PS D:\SEMESTER 3\PBO\Java\Jobsheet08>
```

Human : Base class that defines the methods breathe() and eat().
Lecturer : Derived class that overrides the eat() method and adds the overtime() method.
Student : Derived class that overrides the eat() method and adds the sleep() method.
Main : Main class that tests all methods using dynamic method dispatch.
This code demonstrates the use of inheritance and method overriding in Java, as well as the application of dynamic method dispatch and casting to call specific methods from subclasses.