

SAFRIZAL RAHMAN_19_SIB_2G

Link

https://github.com/safrizalrahman46/PBO_SAFRIZ_THEVIGILANTE

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/invoke 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)

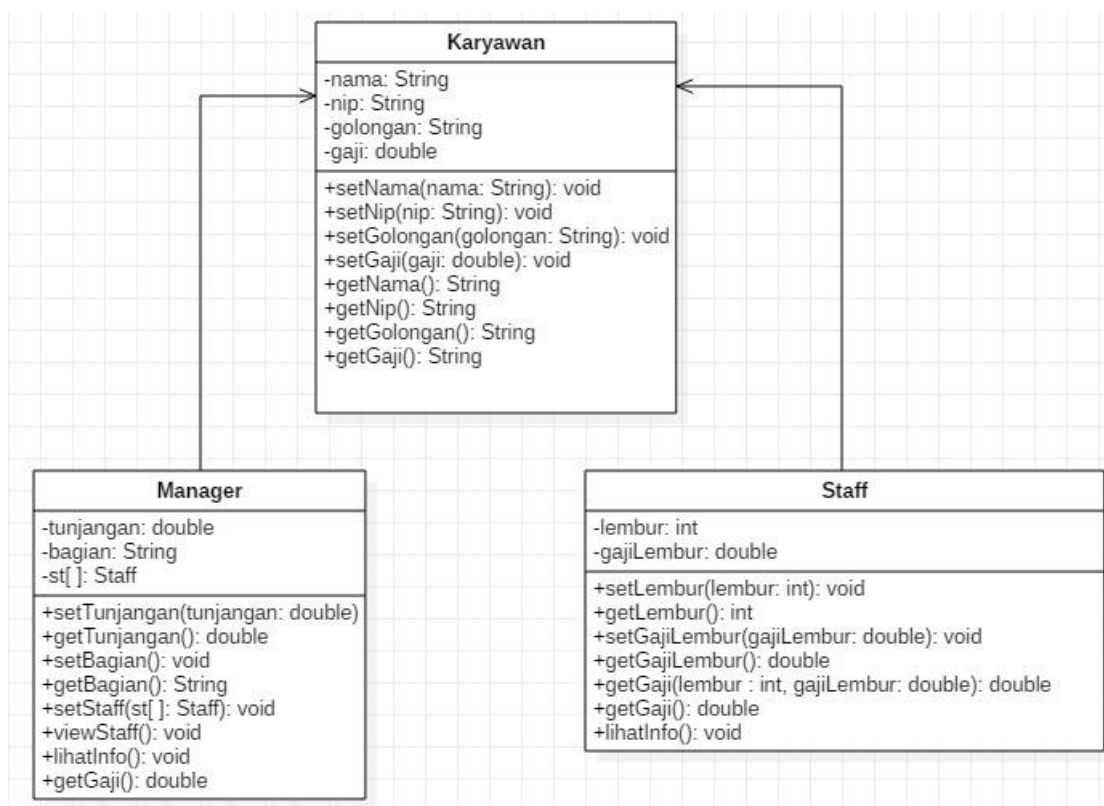
So that the method in the parent class is called the overridden method and the method in the subclass is called the overriding method. There are several method rules in overriding:

- 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

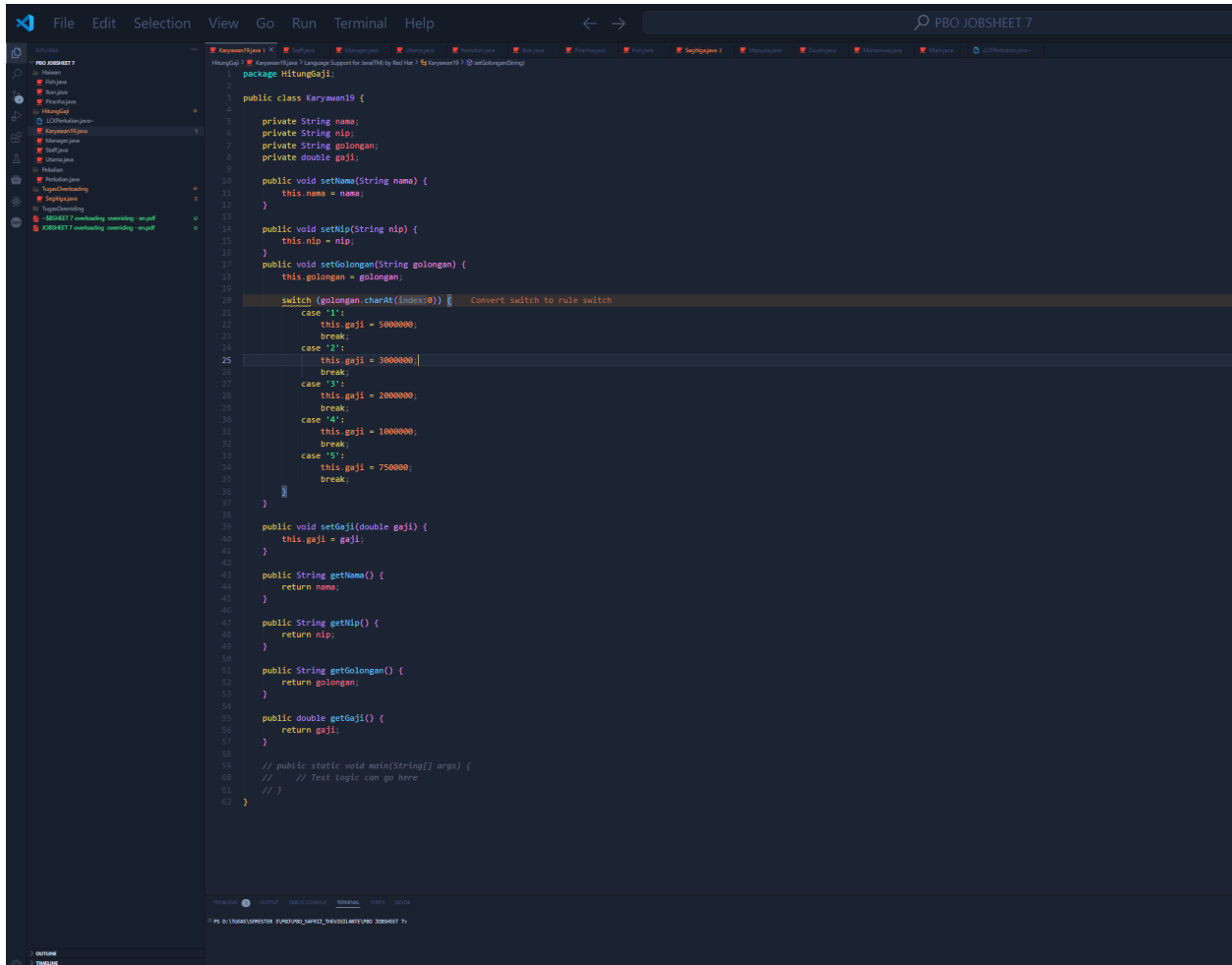
For the following example case, there are three classes, namely Karyawan, Manager, and Staff. Employee Class is a superclass of Manager and Staff where the Manager and Staff subclasses have different methods for calculating salaries.



3.2 Karyawan

```
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;
}
}
```



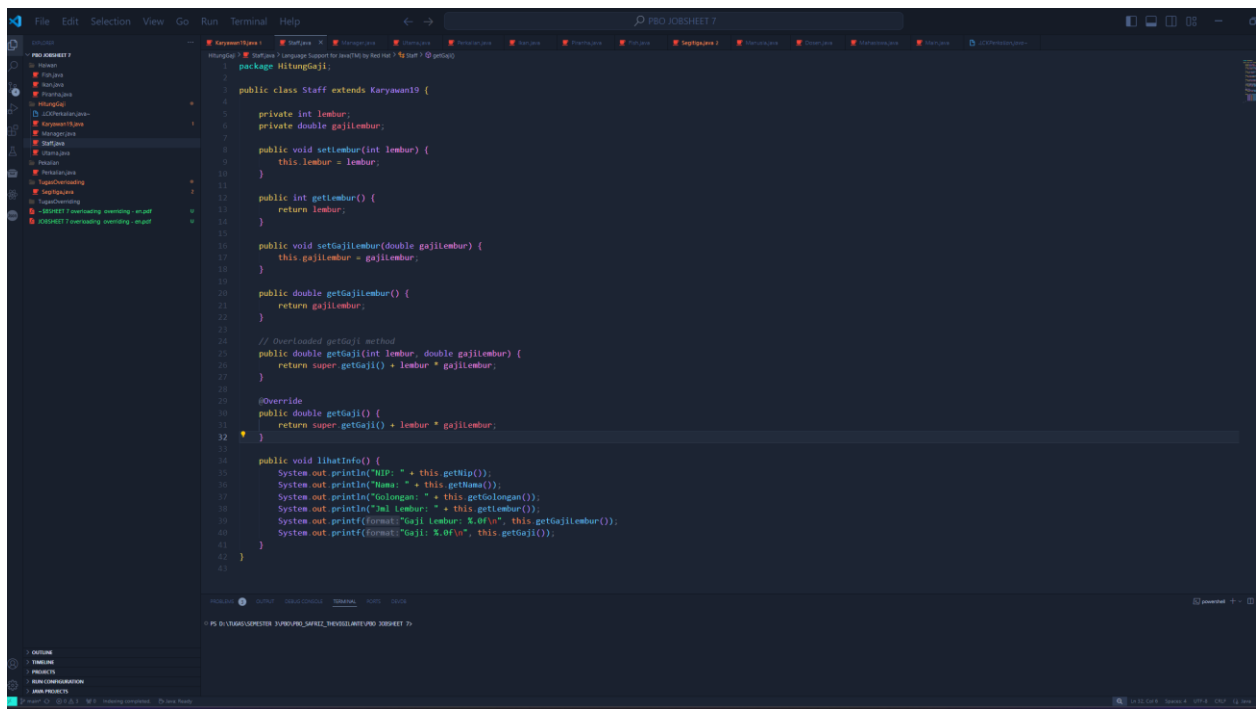
3.3 Staff

```
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;
    }
}

```

```
File Edit Selection View Go Run Terminal Help
PROJ KOSHERET 7

package hitungGaji;

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() {
        for (int i = 0; i < st.length; i++) { // the enhanced for loop to iterate over the array
            st[i].lihatInfo();
        }
        System.out.println("\n-----");
    }

    // @Override
    public void lihatInfo() {
        System.out.println("Manager: " + this.getBagian());
        System.out.println("NIP: " + this.getNIP());
        System.out.println("Nama: " + this.getNama());
        System.out.println("Gaji: " + this.getGaji());
        System.out.println("Tunjangan: " + this.getTunjangan());
        System.out.println("Bagian: " + this.getBagian());
        System.out.println("Staff: " + this.getStaff());
        this.viewStaff();
    }

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

3.5 Main

```
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();
}

```

```

package hitungaji;

public class Main {

    public static void main(String[] args) {
        // Run Main (Debug Mode) Run (Debug)
        Manager man[] = new Manager[2];
        Staff staff[] = new Staff[2];
        Staff staff2[] = new Staff[2];

        // Creating Managers
        man[0] = new Manager();
        man[0].setNama("Taufiq");
        man[0].setNip("0001");
        man[0].setGolongan("1");
        man[0].setLembur(100000);
        man[0].setGajiLembur("Administrasi");

        man[1] = new Manager();
        man[1].setNama("Adisa");
        man[1].setNip("0002");
        man[1].setGolongan("1");
        man[1].setLembur(200000);
        man[1].setGajiLembur("Pemerintah");

        // Creating staff numbers for Manager 0
        staff[0] = new Staff();
        staff[0].setNama("Manager");
        staff[0].setNip("0000");
        staff[0].setGolongan("P3");
        staff[0].setLembur(100);
        staff[0].setGajiLembur(100000);

        staff[1] = new Staff();
        staff[1].setNama("Manager");
        staff[1].setNip("0000");
        staff[1].setGolongan("P3");
        staff[1].setLembur(100);
        staff[1].setGajiLembur(10000);
        man[0].setStaff(staff);

        // Creating staff numbers for Manager 1
        staff2[0] = new Staff();
        staff2[0].setNama("Manager");
        staff2[0].setNip("0000");
        staff2[0].setGolongan("P3");
        staff2[0].setLembur(100);
        staff2[0].setGajiLembur(10000);

        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);

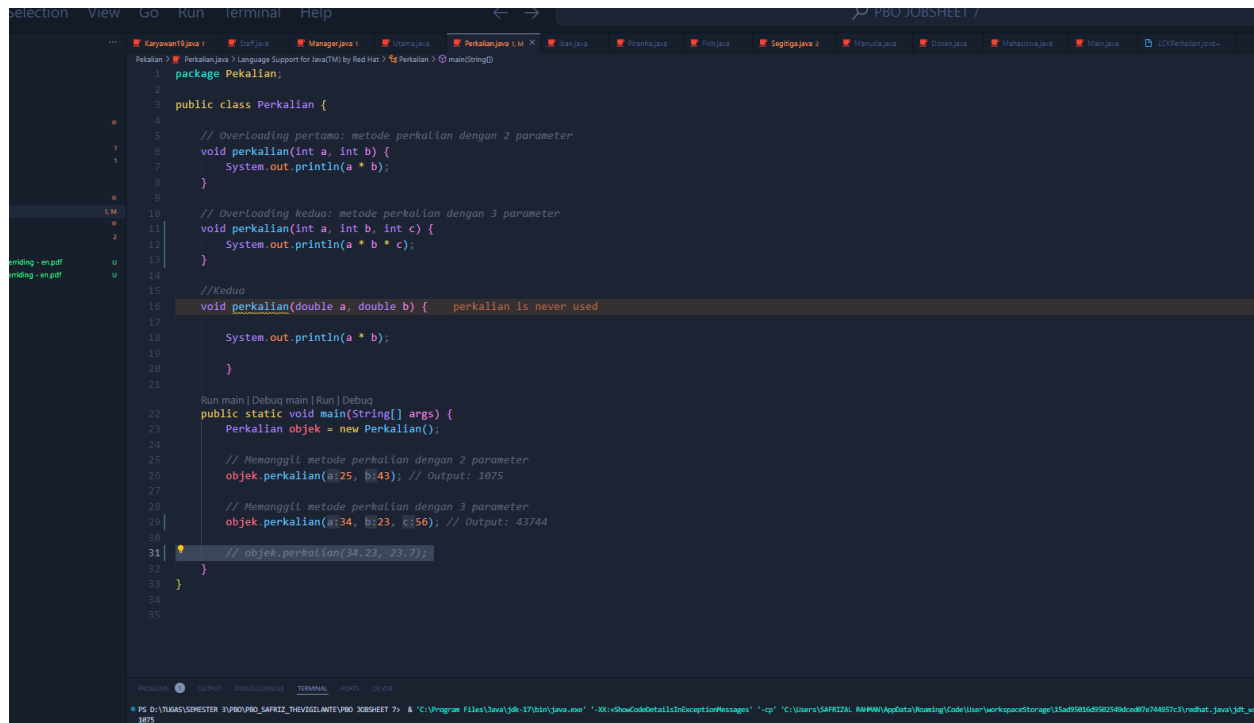
        // Print Information about Managers and their staff
        man[0].lihatInfo();
        man[1].lihatInfo();
    }
}

```

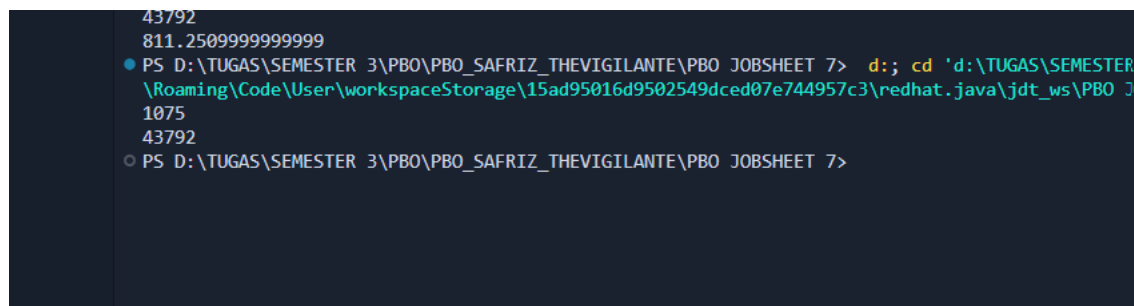
```
PS D:\TUGAS\SEMESTER 3\PROJ\PRO_SAFRIZ_THEVIGILANTE\PRO 308SHEET 7> & "C:\Program Files\Java\jdk-17\bin\java.exe" "-XX:+ShowCodeDetailsInExceptionMessages" "-cp" "C:\Users\SAFRIZAL_RAHMAN\AppData\Roaming\Code\User\workspaceStorage\15ad9581d1\pow
Program Testing Class Manager & Staff
Manager: Administrasi
NIP: 101
Nama: Tedjo
Golongan: 1
Tunjangan: 500000
Gaji: 1000000
Bagian: Administrasi
NIP: 0002
Nama: Usman
Golongan: 2
Jml Lembur: 10
Gaji lembur: 10000
Gaji: 1100000
NIP: 0005
Nama: Amarah
Golongan: 2
Jml Lembur: 10
Gaji lembur: 55000
Gaji: 1550000
-----
Manager: Pemasaran
NIP: 102
Nama: Alika
Golongan: 1
Tunjangan: 250000
Gaji: 750000
Bagian: Pemasaran
NIP: 0004
Nama: Hendra
Golongan: 3
Jml Lembur: 15
Gaji lembur: 5500
Gaji: 2002500
NIP: 0006
Nama: Rifa
Golongan: 4
Jml Lembur: 5
Gaji lembur: 100000
Gaji: 1500000
NIP: 0007
Nama: Rontari
Golongan: 3
Jml Lembur: 6
Gaji lembur: 20000
Gaji: 2120000
-----
PS D:\TUGAS\SEMESTER 3\PROJ\PRO_SAFRIZ_THEVIGILANTE\PRO 308SHEET 7>
```

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);  
    }  
}
```



```
1 package Perkalian;
2
3
4 public class Perkalian {
5     // Overloading pertama: metode perkalian dengan 2 parameter
6     void perkalian(int a, int b) {
7         System.out.println(a * b);
8     }
9
10    // Overloading kedua: metode perkalian dengan 3 parameter
11    void perkalian(int a, int b, int c) {
12        System.out.println(a * b * c);
13    }
14
15    //Kedua
16    void perkalian(double a, double b) { perkalian is never used
17
18        System.out.println(a * b);
19    }
20
21
22    Run main | Debug main | Run | Debug
23    public static void main(String[] args) {
24        Perkalian objek = new Perkalian();
25
26        // Memanggil metode perkalian dengan 2 parameter
27        objek.perkalian(8, 13); // Output: 1075
28
29        // Memanggil metode perkalian dengan 3 parameter
30        objek.perkalian(8, 13, 56); // Output: 43744
31
32        // objek.perkalian(34, 23, 23.7);
33    }
34
35 }
```



```
43792
811.2509999999999
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO JOBSHEET 7> d;; cd 'd:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO JOBSHEET 7' & java -cp . Perkalian
1075
43792
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO JOBSHEET 7>
```

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

Overloading occurs when you have multiple methods with the same name but different parameters (in number, type, or both). In the provided code, the method overloading happens in the `Perkalian` class. One takes two parameters (`int a, int b`)

Another takes three parameters (`int a, int b, int c`)

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

- One method accepts two parameters.

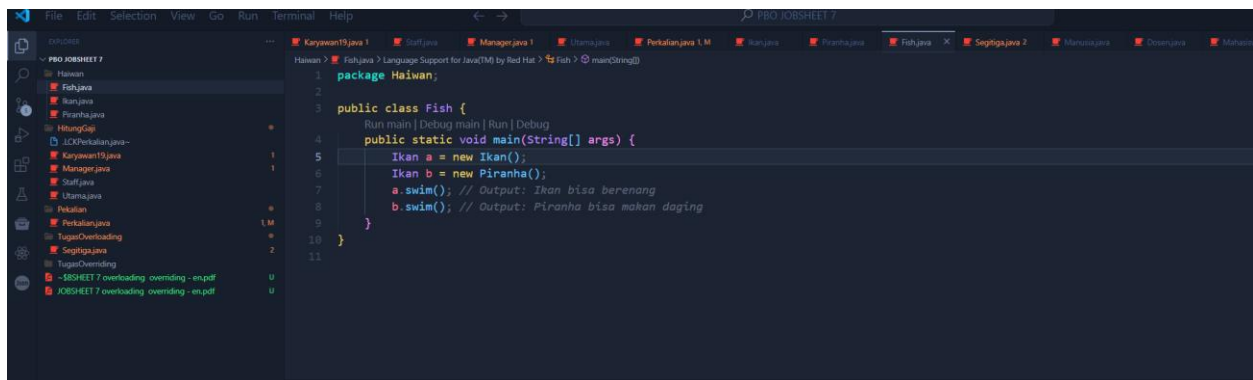
- Another method accepts three parameters.

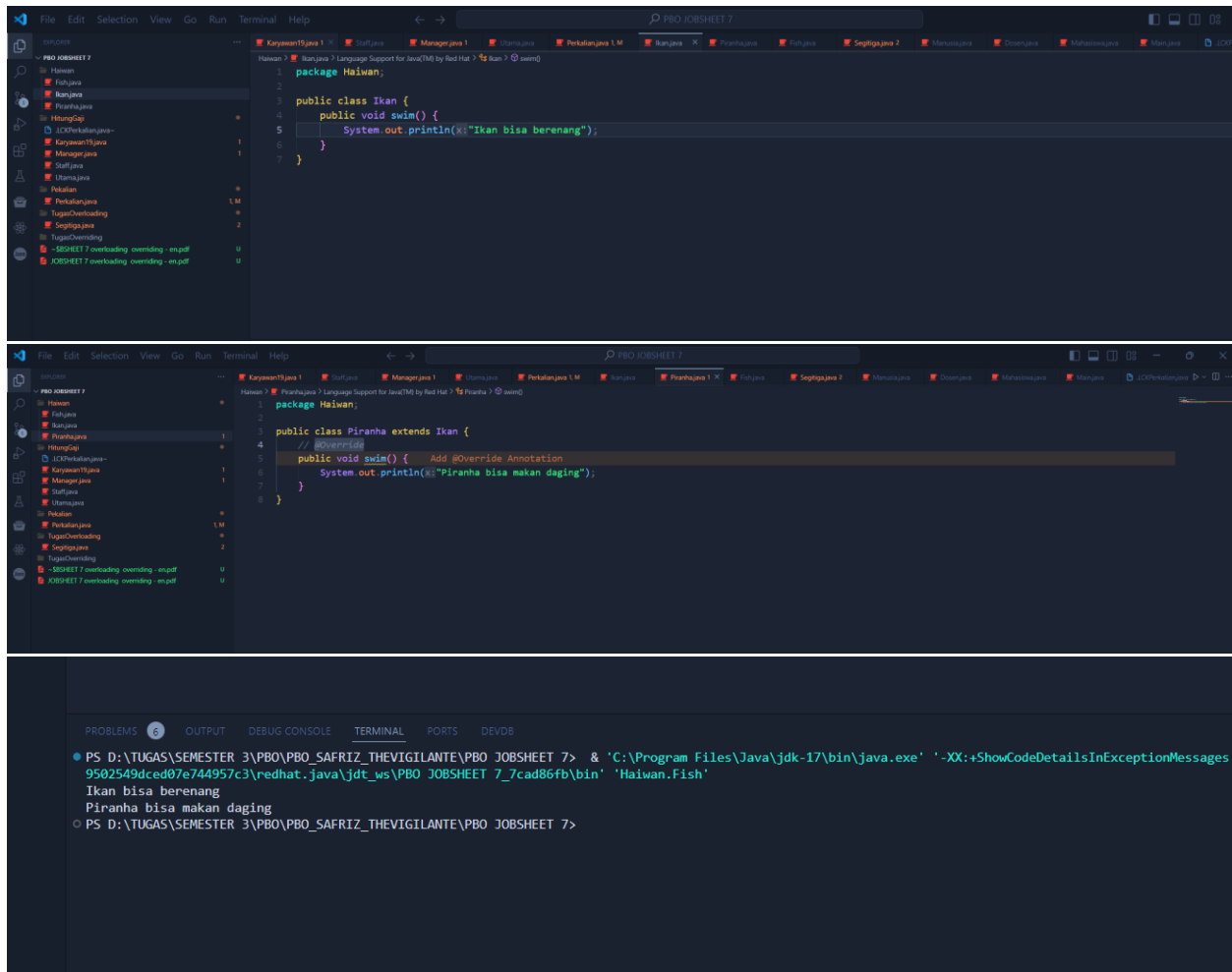
```
public class PerkalianKu {  
    void perkalian(int a, int b){  
        System.out.println(a * b);  
    }  
    void perkalian(double a, double b){  
        System.out.println(a * b);  
    }  
    public static void main(String args []){  
        PerkalianKu objek = new PerkalianKu();  
        objek.perkalian(25, 43);  
        objek.perkalian(34.56, 23.7);  
    }  
}
```


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

- **int** type: The method void perkalian(int a, int b) accepts two integers.
- **double** type: The method void perkalian(double a, double b) accepts two double values.

```
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 source coding above, where is the overriding?

Overriding occurs on the swim() method in the Piranha class, which overrides the swim() method in the parent fish class. Both methods have the same name and the same parameters, but their implementation is different.

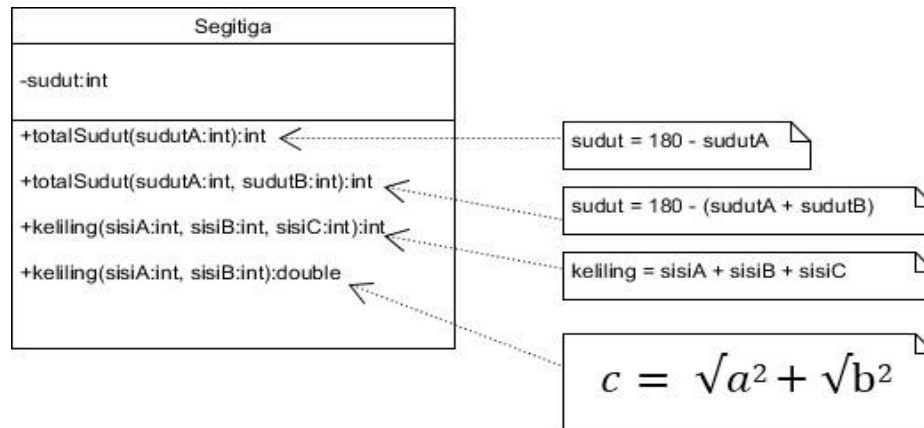
4.6 Describe when sourcoding above if there is overriding?

- The parent class Fish swims () method that displays "fish can swim".
- The Piranha child class overrides the swim() method of the parent class and provides a different implementation, which displays "piranhas can eat meat".

5. Tasks

5.1 Overloading

Implement the overloading concept in the diagram class below:



```
package TugasOverloading;

public class Segitiga {
    private int sudut; // The value of the field Segitiga.sudut is not used

    // Metode pertama: Menghitung total sudut segitiga
    public int totalSudut(int sudutA) {
        return 180 - sudutA;
    }

    // Metode kedua: Overloading untuk menghitung total sudut dengan dua sudut
    public int totalSudut(int sudutA, int sudutB) {
        return 180 - (sudutA + sudutB);
    }

    // Metode ketiga: Menghitung keliling segitiga dengan tiga sisi bertipe int
    public int keliling(int sisiA, int sisiB, int sisiC) {
        return sisiA + sisiB + sisiC;
    }

    // Metode keempat: Overloading keliling segitiga dengan dua sisi bertipe double
    public double keliling(double sisiA, double sisiB) {
        return Math.sqrt((sisiA * sisiA) + (sisiB * sisiB));
    }

    // public static void main(String[] args) {
    //     Segitiga segitiga = new Segitiga();

    //     // Contoh pemanggilan metode totalSudut
    //     System.out.println("Total sudut (1 sudut): " + segitiga.totalSudut(60));
    //     System.out.println("Total sudut (2 sudut): " + segitiga.totalSudut(60, 30));

    //     // Contoh pemanggilan metode keliling
    //     System.out.println("Keliling (3 sisi): " + segitiga.keliling(3, 4, 5));
    //     System.out.println("Keliling (2 sisi): " + segitiga.keliling(3.0, 4.0));
    // }
}
```

Terminal Output:

```
PS D:\TUGAS\SEMESTER 3\PROB\PRO_SAFRIZ\THEVIGILANTE\PRO 30BSHEET 7> & 'C:\Program Files\Java\jdk-17\bin\java.exe' ^
Total sudut (1 sudut): 120
Total sudut (2 sudut): 90
Keliling (3 sisi): 12
Keliling (2 sisi): 5.0
PS D:\TUGAS\SEMESTER 3\PROB\PRO_SAFRIZ\THEVIGILANTE\PRO 30BSHEET 7>
```

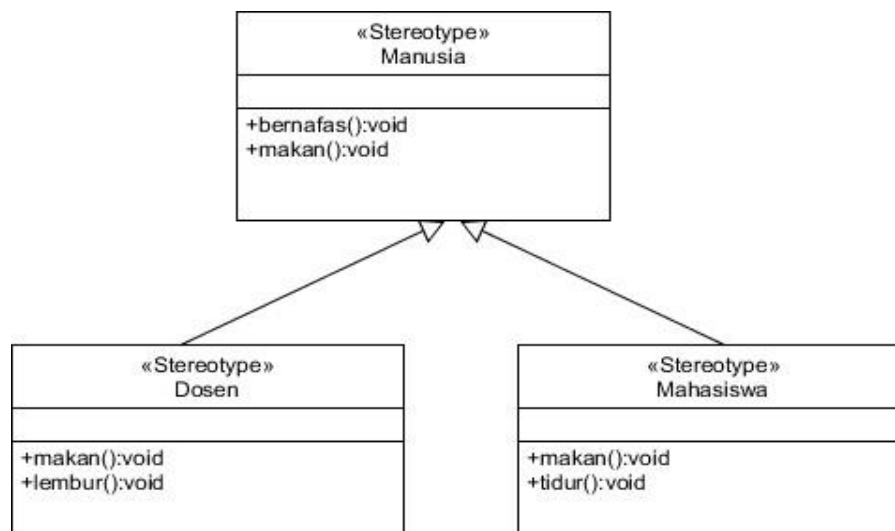
```
... Karyawan15.java 1 Staf.java 1 Manager.java 1 Utamajava 1 Pertalian.java 1.M 1 Rangkajava 1 Piranha.java 1 Poth.java 1 Segitajava 2.M 1 Manjaya TugasOverloading 1.U X 1 Manasajava 1 Doonajava 1 Mahas...

TugasOverloading > Manjava 2 ...
1 package TugasOverloading; Incorrect Package
2
3 public class Main {
4     Run main | Debug main | Run | Debug
5     public static void main(String[] args) {
6         Segitiga segitiga = new Segitiga();
7
8         // Contoh pemanggilan metode totalSudut
9         System.out.println("Total sudut (1 sudut): " + segitiga.totalSudut(sudutA:60));
10        System.out.println("Total sudut (2 sudut): " + segitiga.totalSudut(sudutA:60, sudutB:30));
11
12        // Contoh pemanggilan metode keliling
13        System.out.println("Keliling (3 sisi): " + segitiga.keliling(sisiA:3, sisiB:4, sisiC:5));
14        System.out.println("Keliling (2 sisi): " + segitiga.keliling(sisiA:3.0, sisiB:4.0));
15    }
16 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVICES
PS D:\TUGAS\SEMESTER 3\PROJ\PRO_SAFRIZ_THEVIGILANTE\PRO_JOBSSHEET 7> & "C:\Program Files\Java\jdk-17\bin\java.exe" "-XX:ShowCodeDetailsInExceptionMessages" "-cp" "C:\Users\SAFRIZAL_RAHMAN\AppData\Roaming\Code\User\workspaceStorage\15ad95f...
at java\jdk.unw\PRO_JOBSSHEET 7_7\cad\6th\bin" "TugasOverloading_Main"
Total sudut (1 sudut): 120
Total sudut (2 sudut): 90
Keliling (3 sisi): 12
Keliling (2 sisi): 5.0
PS D:\TUGAS\SEMESTER 3\PROJ\PRO_SAFRIZ_THEVIGILANTE\PRO_JOBSSHEET 7>
```

5.2 Overriding

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



```
View Go Run Terminal Help
PBO JOBSHEET 7

Karyawan19.java 1 Staff.java Manager.java 1 Utama.java Perkalian.java 1, M Ikan.java Piranha.java 1 Fish.java Segitiga.java 2, M

TugasOverriding > Dosen.java > Language Support for Java(TM) by Red Hat > Dosen
1 package TugasOverriding;
2
3 public class Dosen extends Manusia {
4     // @Override
5     public void makan() { Add @Override Annotation
6         System.out.println(x: "Dosen makan di kantin universitas.");
7     }
8
9     public void lembur() {
10        System.out.println(x: "Dosen sedang lembur menyiapkan materi.");
11    }
12 }
```

```
Go Run Terminal Help
PBO JOBSHEET 7

Karyawan19.java 1 Staff.java Manager.java 1 Utama.java Perkalian.java 1, M Ikan.java Piranha.java 1 Fish.java Segitiga.java 2, M Main.java TugasOverriding

TugasOverriding > Mahasiswa.java > Language Support for Java(TM) by Red Hat > Mahasiswa > makan()
1 package TugasOverriding;
2
3 public class Mahasiswa extends Manusia {
4     @Override
5     public void makan() {
6         System.out.println(x: "Mahasiswa makan mie instan.");
7     }
8
9     public void tidur() {
10        System.out.println(x: "Mahasiswa tidur setelah belajar.");
11    }
12 }
```

