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/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)

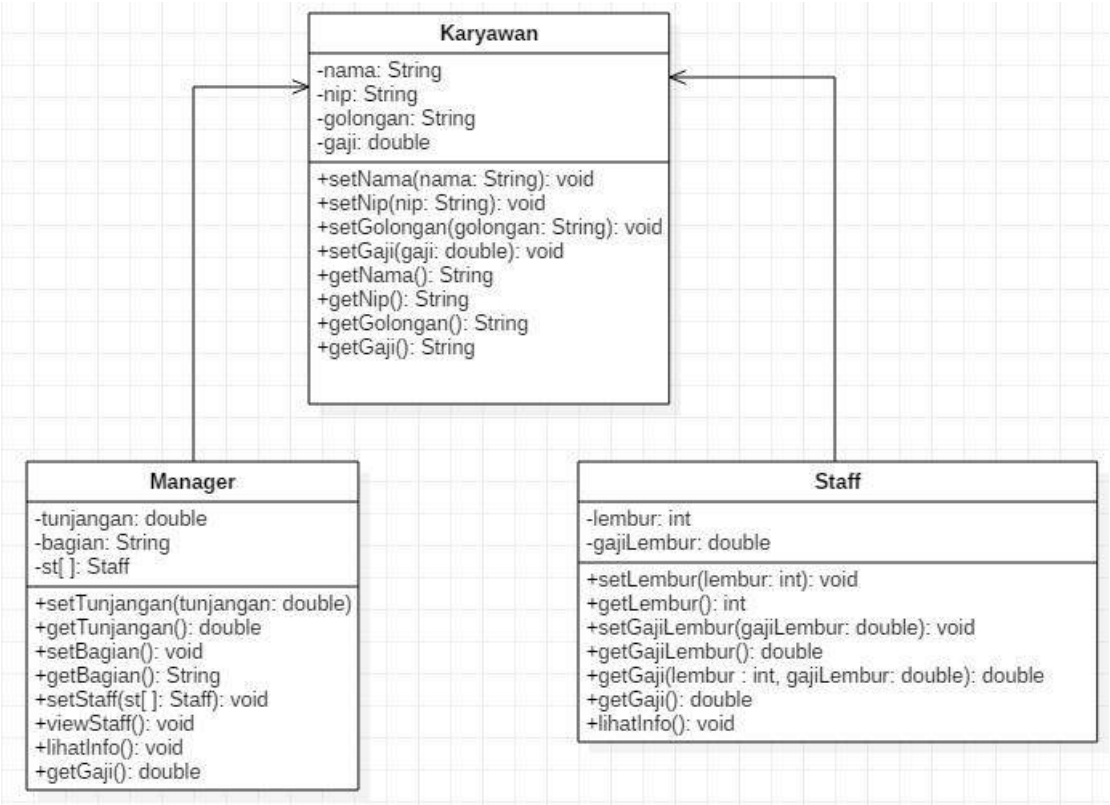
So that the method in the parent class is called the overridden method and the method in the subclass is called the overriding method. There are several method rules in overriding:

- 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

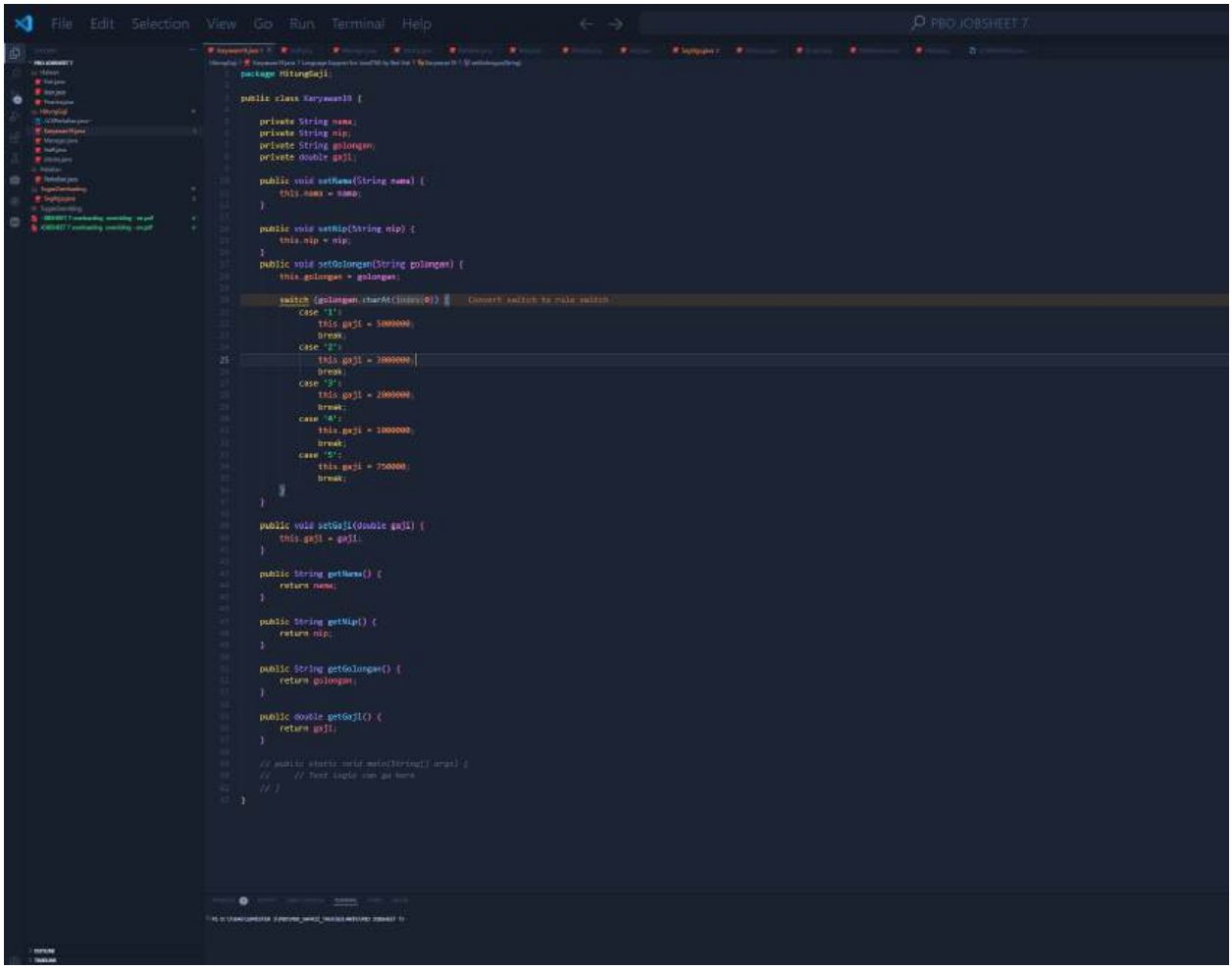
For the following example case, there are three classes, namely Karyawan, Manager, and Staff. Employee Class is a superclass of Manager and Staff where the Manager and Staff subclasses have different methods for calculating salaries.



3.2 Karyawan

```
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;
}
}
```

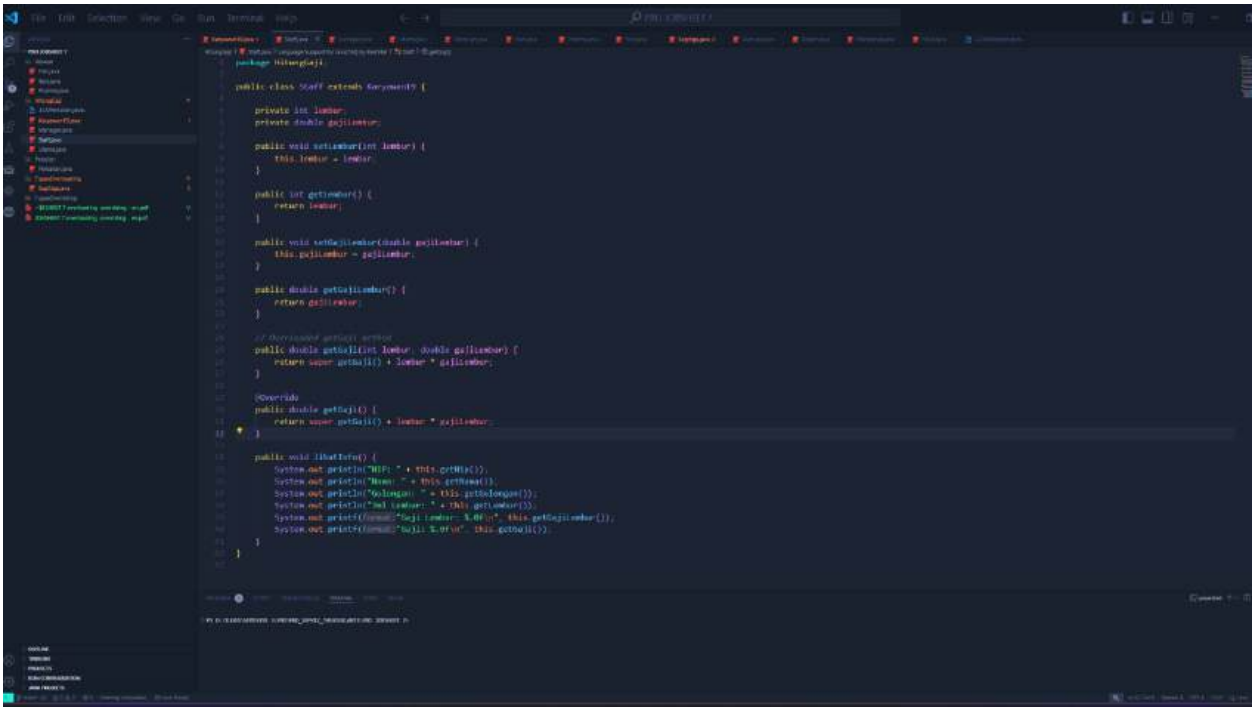

File Edit Selection View Go Run Terminal Help
PBO JOBSHEET 7
package HitungGaji;
public class Karyawan10 {
private String nama;
private String nip;
private String golongan;
private double gaji;
public void setNama(String nama) {
this.nama = nama;
}
public void setNip(String nip) {
this.nip = nip;
}
public 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;
}
}
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;
}
// public static void main(String[] args) {
// // Test logic can go here
// }
}

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());
 }
}
```

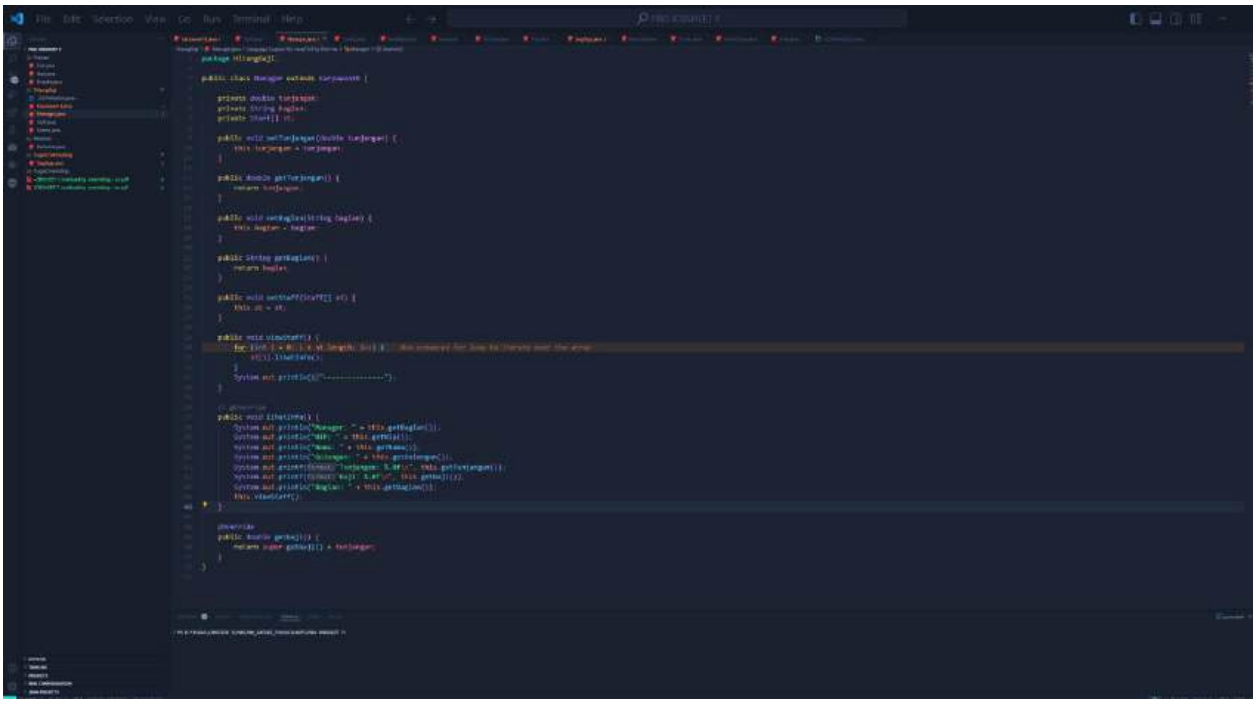
Overloading

Overriding



3.4 Manager

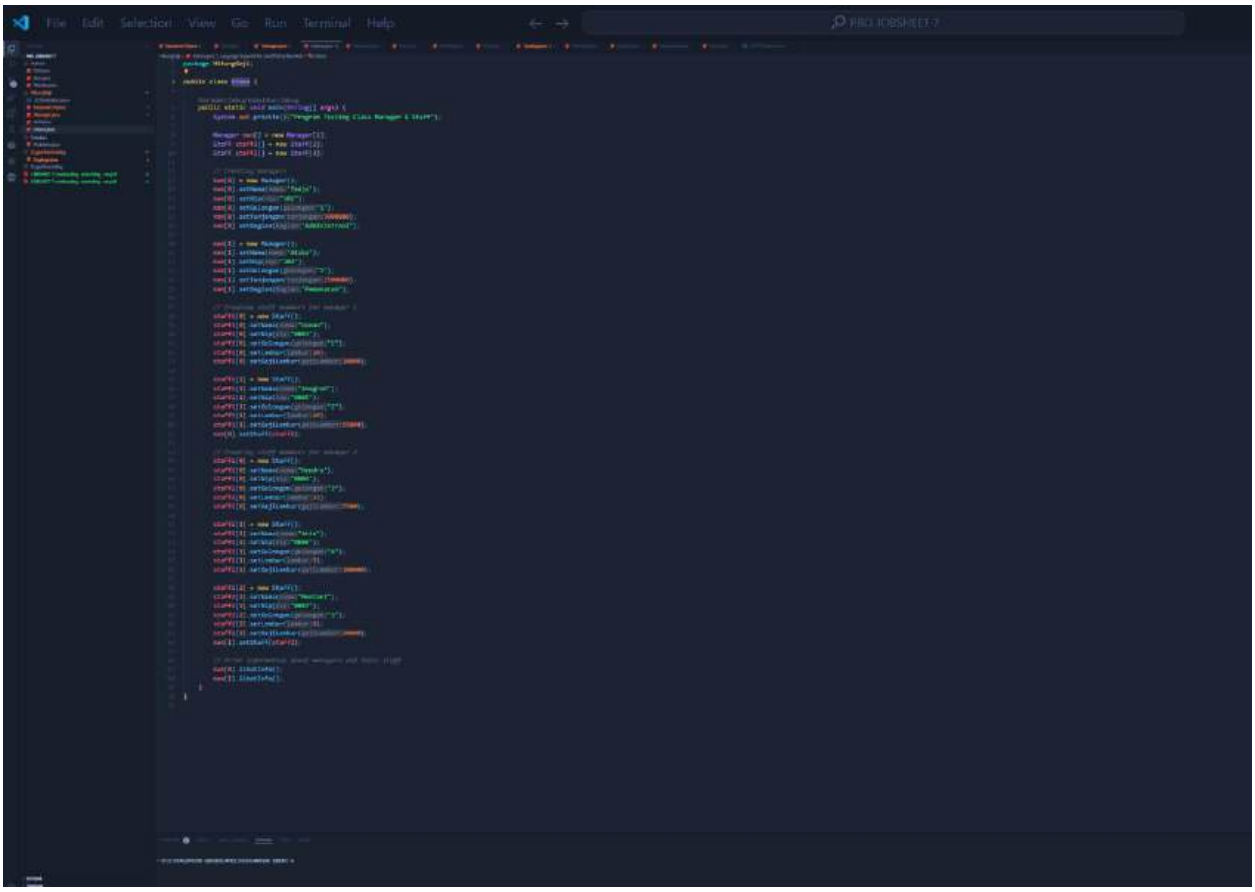
```
public class Manager extends Karyawan {
private double tunjangan;
private String bagian;
private Staff st[];

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;
}
}
```

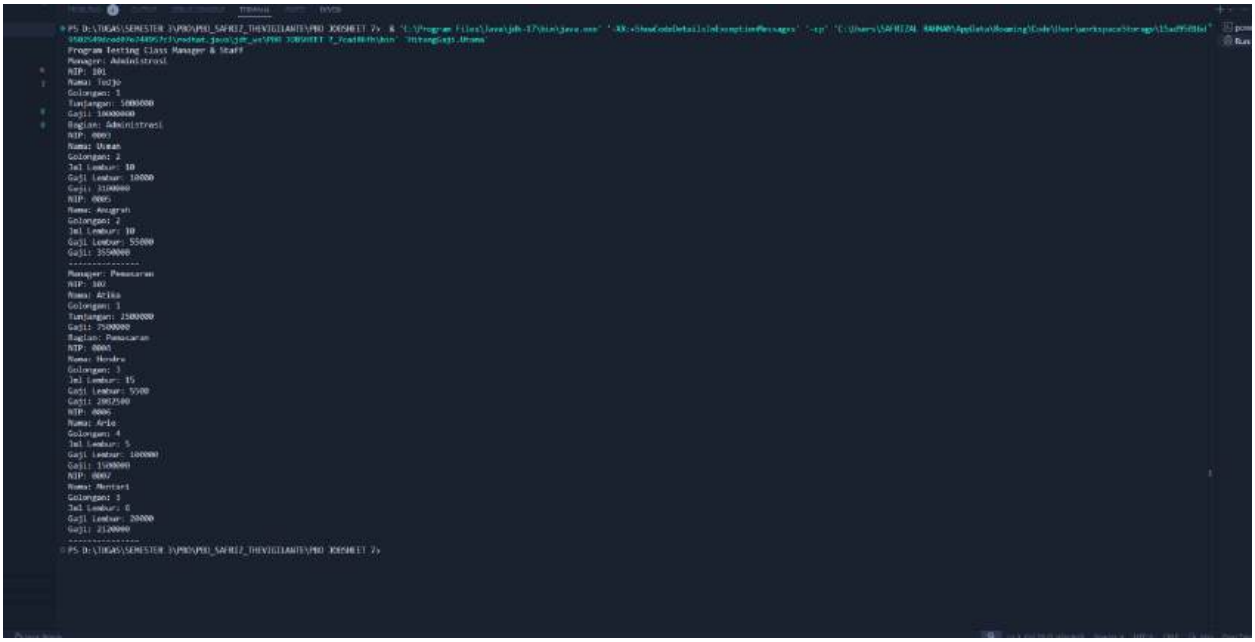


3.5 Main

```
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);
```

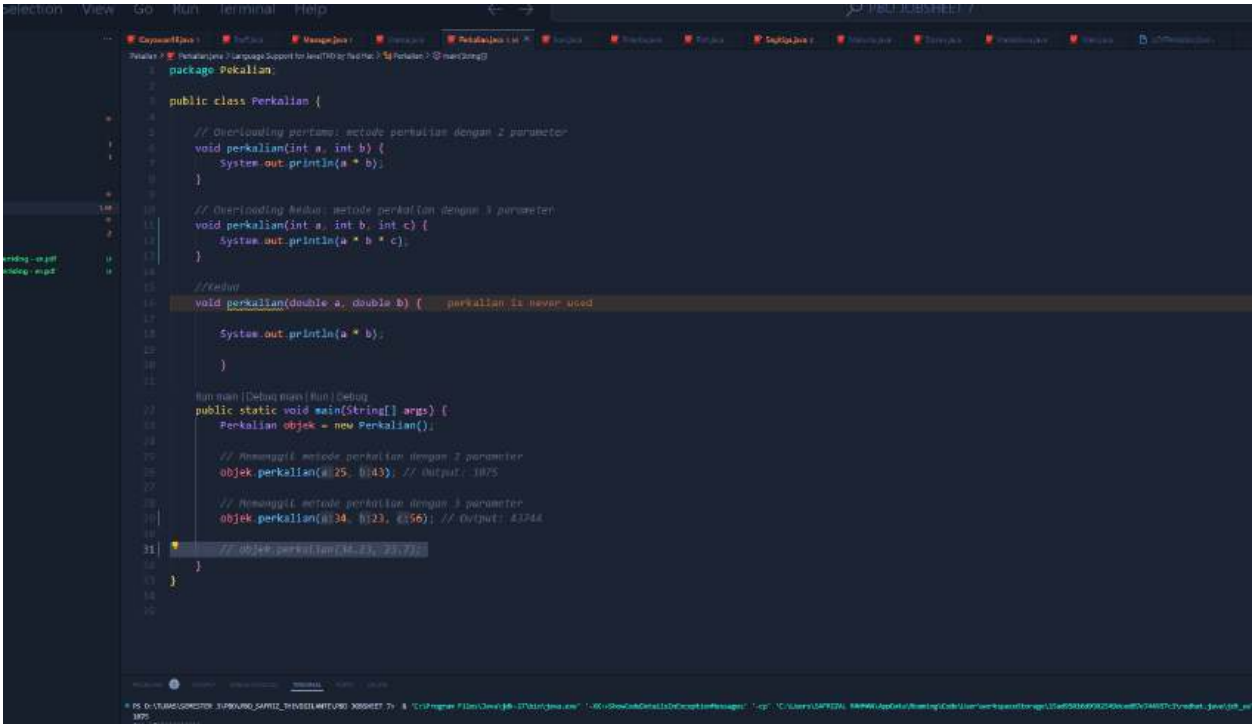
```
staff2[1]=new Staff();
staff2[1].setNama("Arie");
staff2[1].setNip("0006");
staff2[1].setGolongan("4");
staff2[1].setLembur(5);
staff2[1].setGajiLembur(100000);

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();
}
```



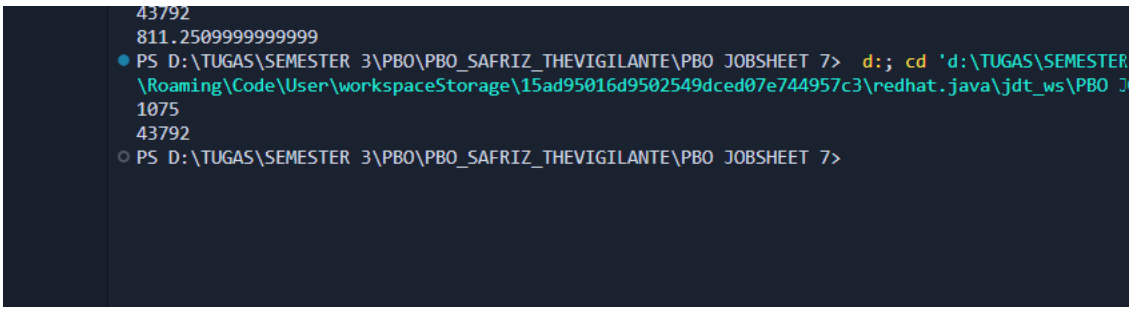


4. Exercise

```java
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);
 }
}
```



```
package Perkalian;

public class Perkalian {

    // Overloading pertama: metode perkalian dengan 2 parameter
    void perkalian(int a, int b) {
        System.out.println(a * b);
    }

    // Overloading kedua: metode perkalian dengan 3 parameter
    void perkalian(int a, int b, int c) {
        System.out.println(a * b * c);
    }

    //kedua
    void perkalian(double a, double b) {    perkalian is never used

        System.out.println(a * b);

        }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Perkalian objek = new Perkalian();

        // Memanggil metode perkalian dengan 2 parameter
        objek.perkalian(a: 25, b:43); // Output: 1075

        // Memanggil metode perkalian dengan 3 parameter
        objek.perkalian(a: 34, b:23, c:56); // Output: 43744

        // objek.perkalian(34.23, 23.7);
    }
}
```



```
43792
811.2509999999999
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO JOBSHEET 7>  d:; cd 'd:\TUGAS\SEMESTER
\Roaming\Code\User\workspaceStorage\15ad95016d9502549dced07e744957c3\redhat.java\jdt_ws\PBO J
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 Perkalianku 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);
 }
}
```

```java
package Pekalian;

public class Perkalian {

    // Overloading pertama: metode perkalian dengan 2 parameter
    void perkalian(int a, int b) {
        System.out.println(a * b);
    }

    // Overloading kedua: metode perkalian dengan 3 parameter
    // void perkalian(int a, int b, int c) {
    //     System.out.println(a * b * c);
    // }

    //Kedua
    void perkalian(double a, double b) {

        System.out.println(a * b);

        }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Perkalian objek = new Perkalian();

        // Memanggil metode perkalian dengan 2 parameter
        objek.perkalian(a:25, b:43); // Output: 1075

        // Memanggil metode perkalian dengan 3 parameter
        // objek.perkalian(34, 23, 56); // Output: 43744

        objek.perkalian(a:34.23, b:23.7);
    }
}
```

```java
        public static void main(String[] args) {
            Perkalian objek = new Perkalian();

            // Memanggil metode perkalian dengan 2 para
            objek.perkalian(a:25, b:43); // Output: 107
```

```
PROBLEMS 4  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  DEVDB
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO JOBSHEET 7>  & 'C:\Program Files\Java
1075
811.2509999999999
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO JOBSHEET 7>
```

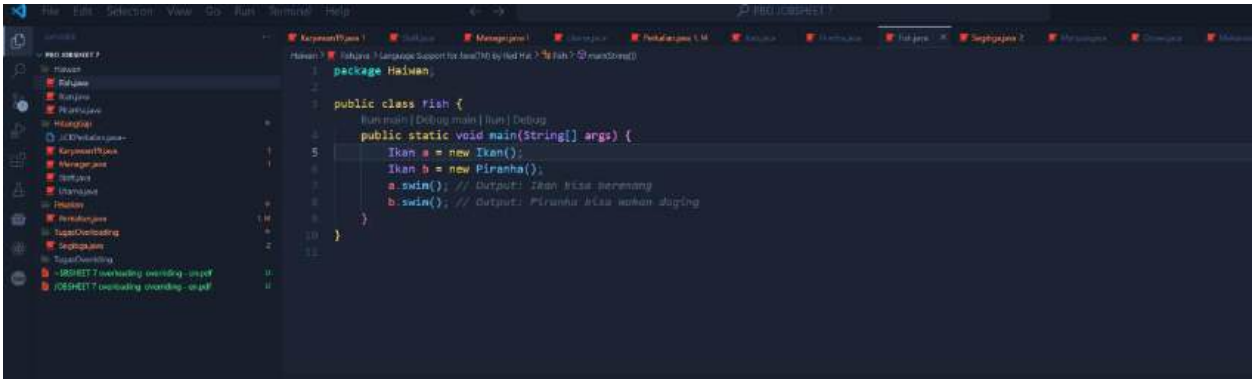
4.3 From the source coding above, where is the overloading?

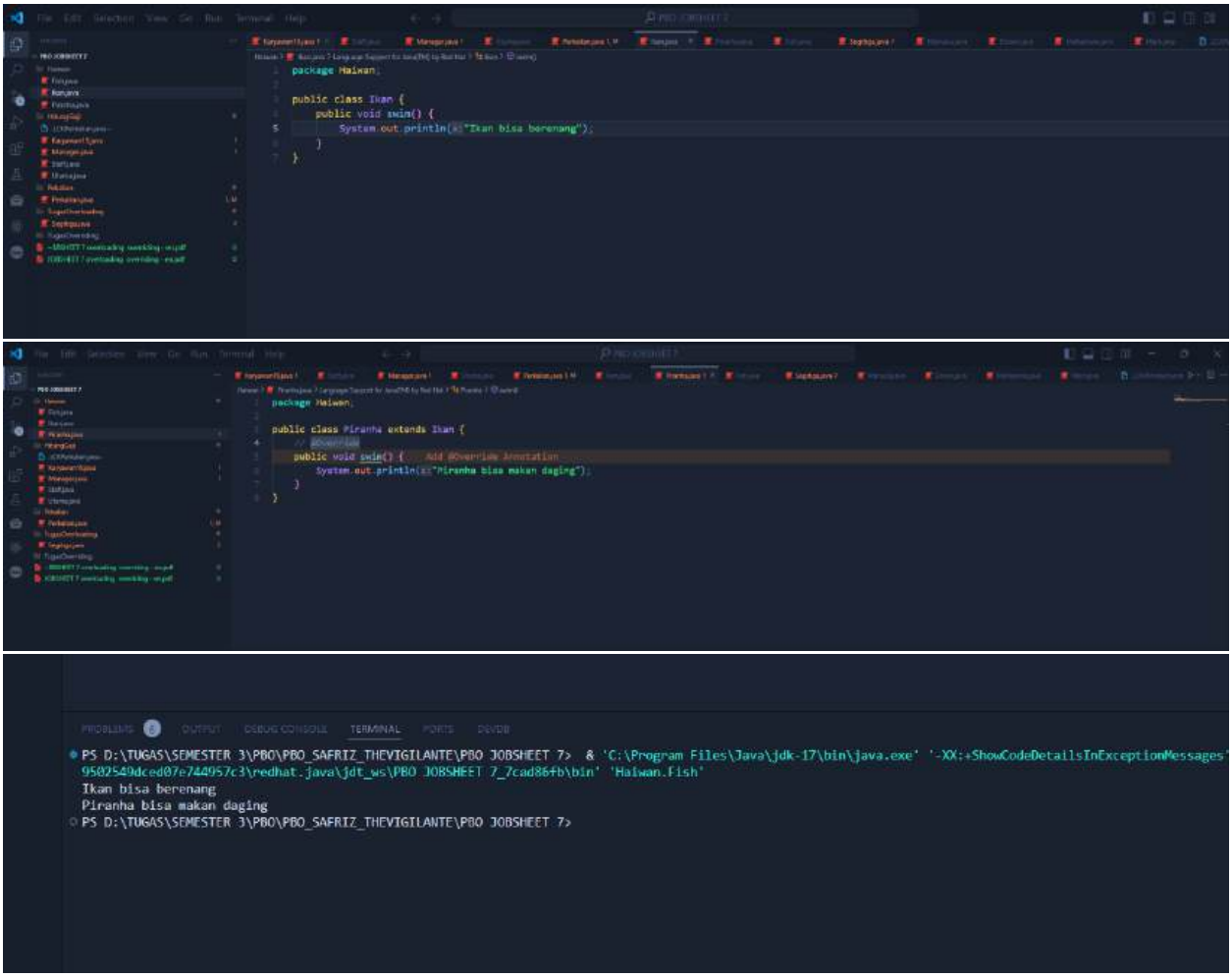
- void perkalian(int a, int b) – This method accepts two int parameters.
- void perkalian(double a, double b) – This method accepts two double parameters.

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.

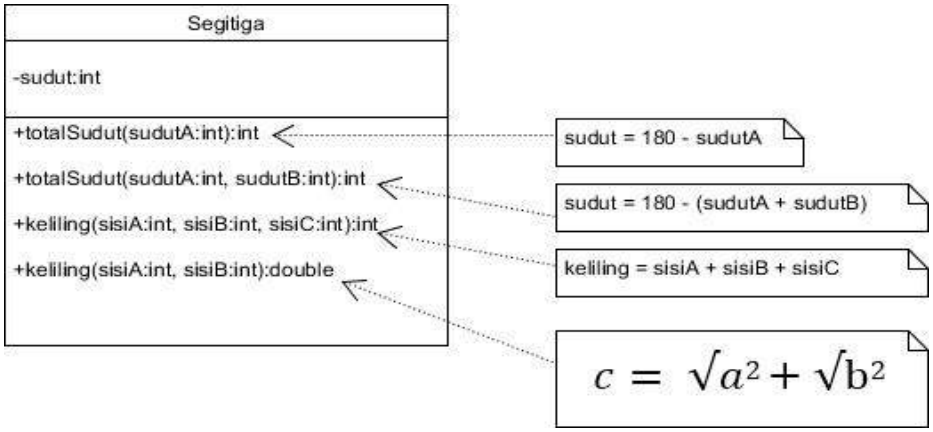
4.6 Describe when sourcoding above if there is overriding?

- 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

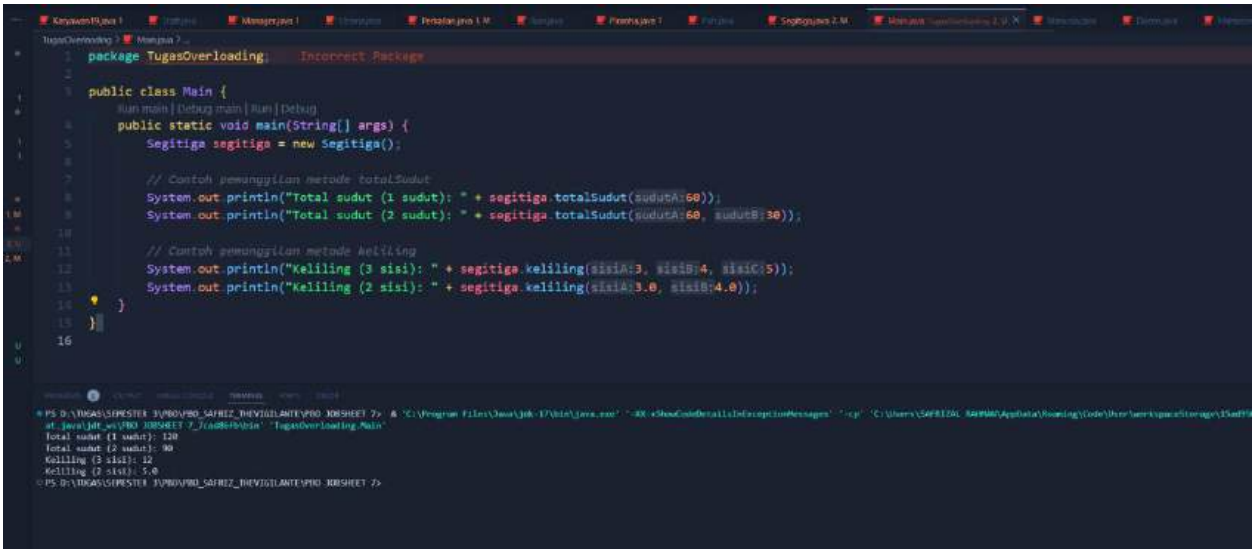
5.1 Overloading

Implement the overloading concept in the diagram class below:



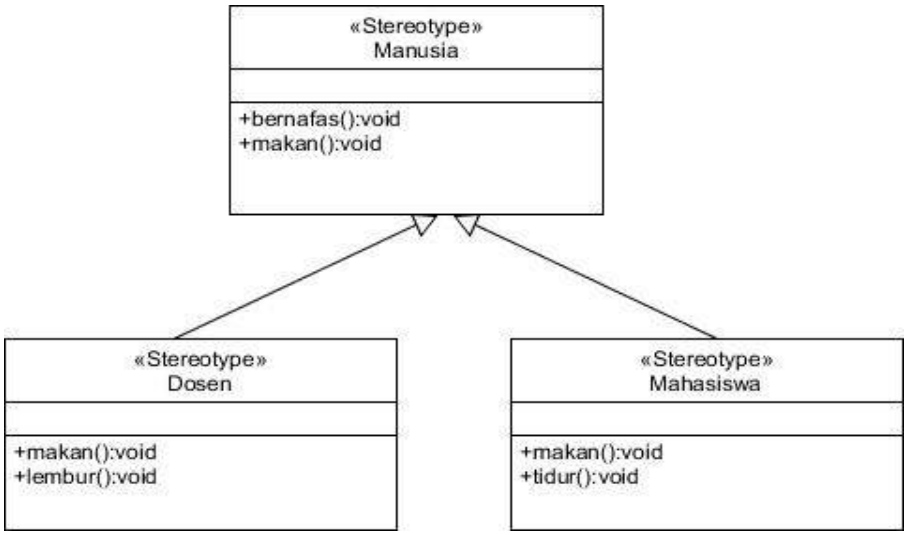

```java
package TugasOverloading;

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));
    // }
}
```

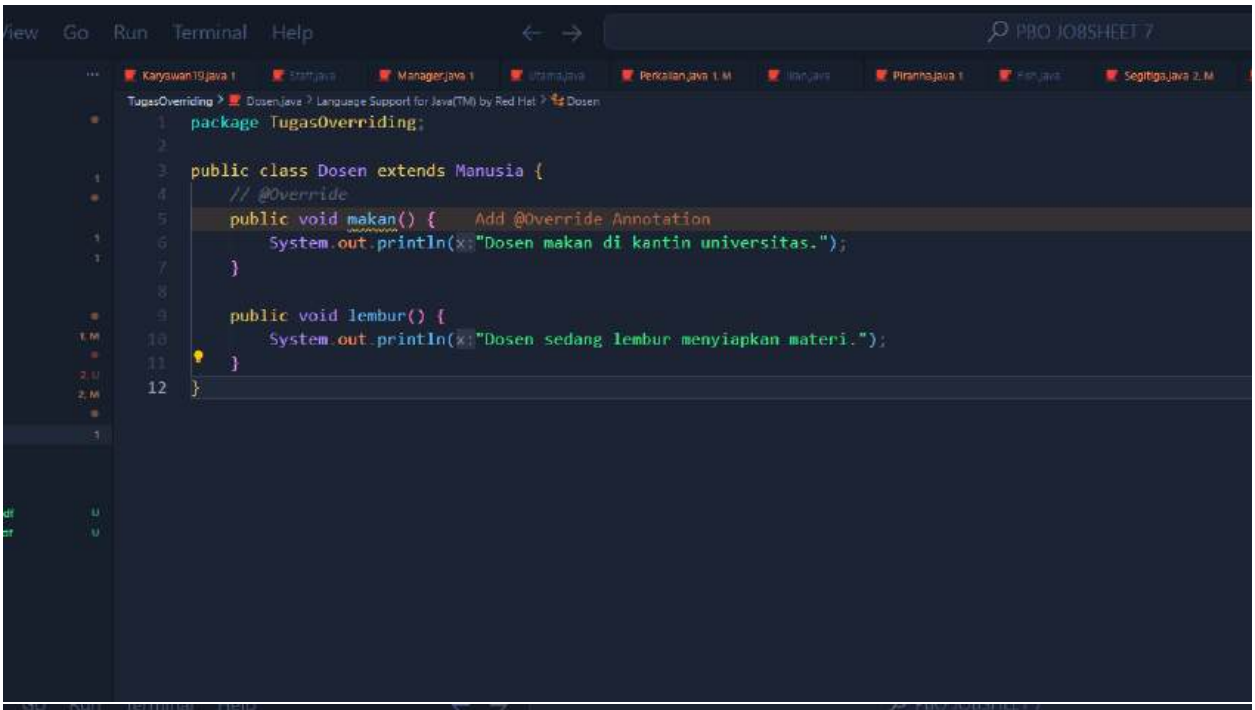
```
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIDELANTEV\PBO JOBSHEET 7> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SAFRIZAL RAMAN\AppData\Roaming\Code\User\workspaceStorage\15ad9588689f892549dced824
Total sudut (1 sudut): 120
Total sudut (2 sudut): 90
Keliling (3 sisi): 12
Keliling (2 sisi): 5.0
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIDELANTEV\PBO JOBSHEET 7>
```



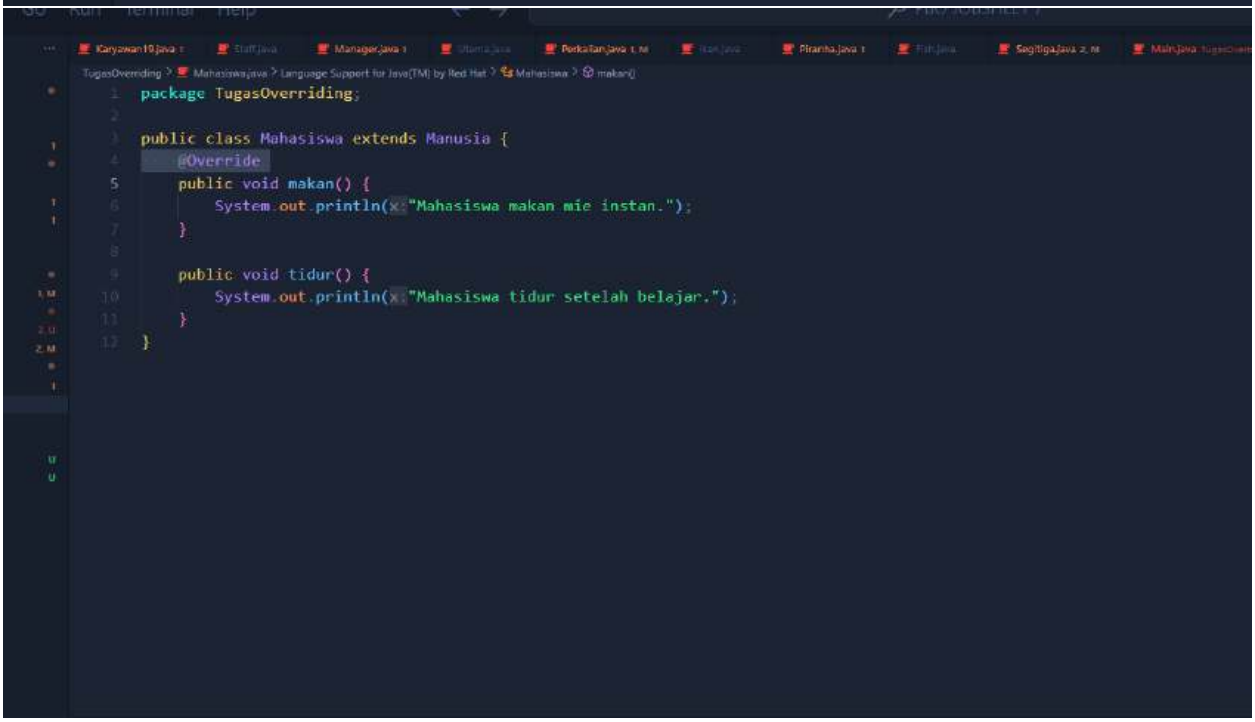
5.2 Overriding

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

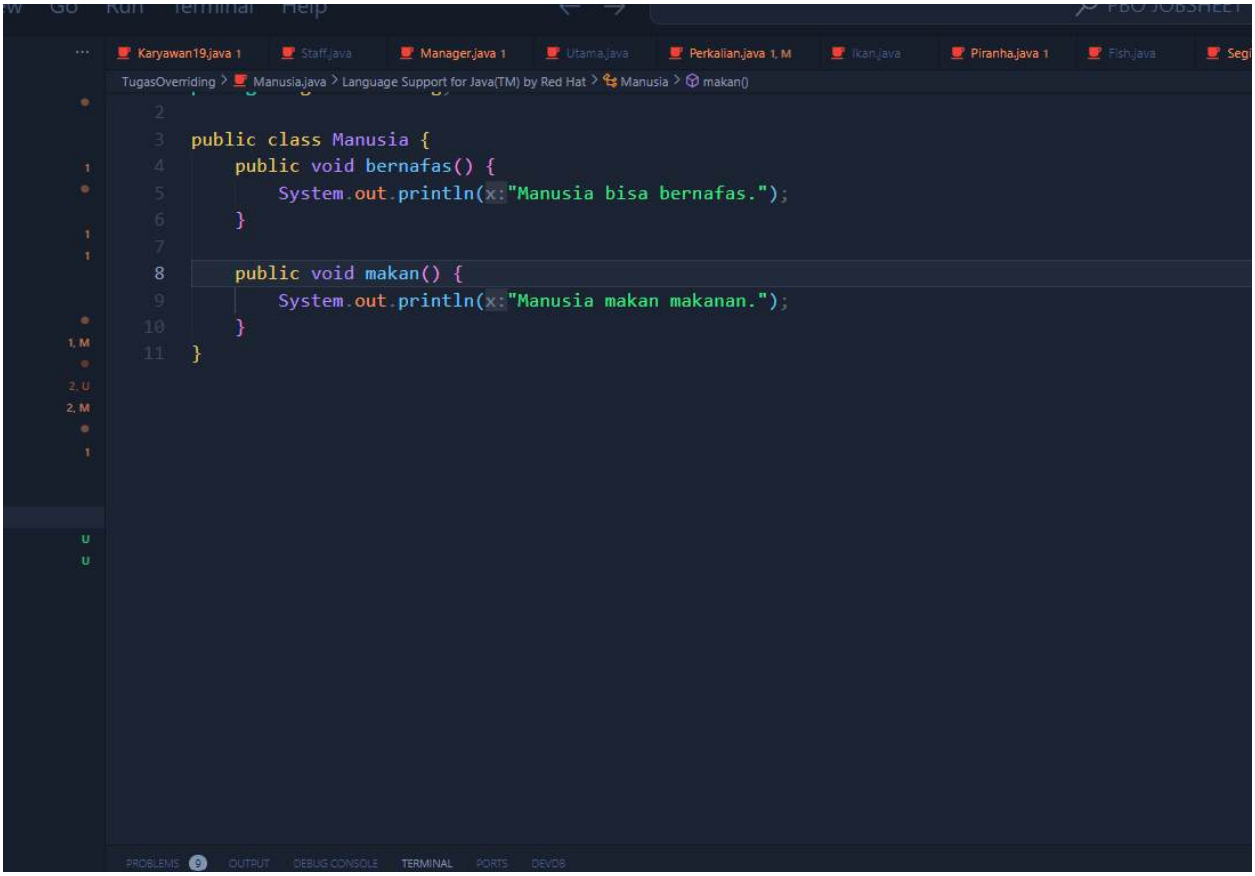


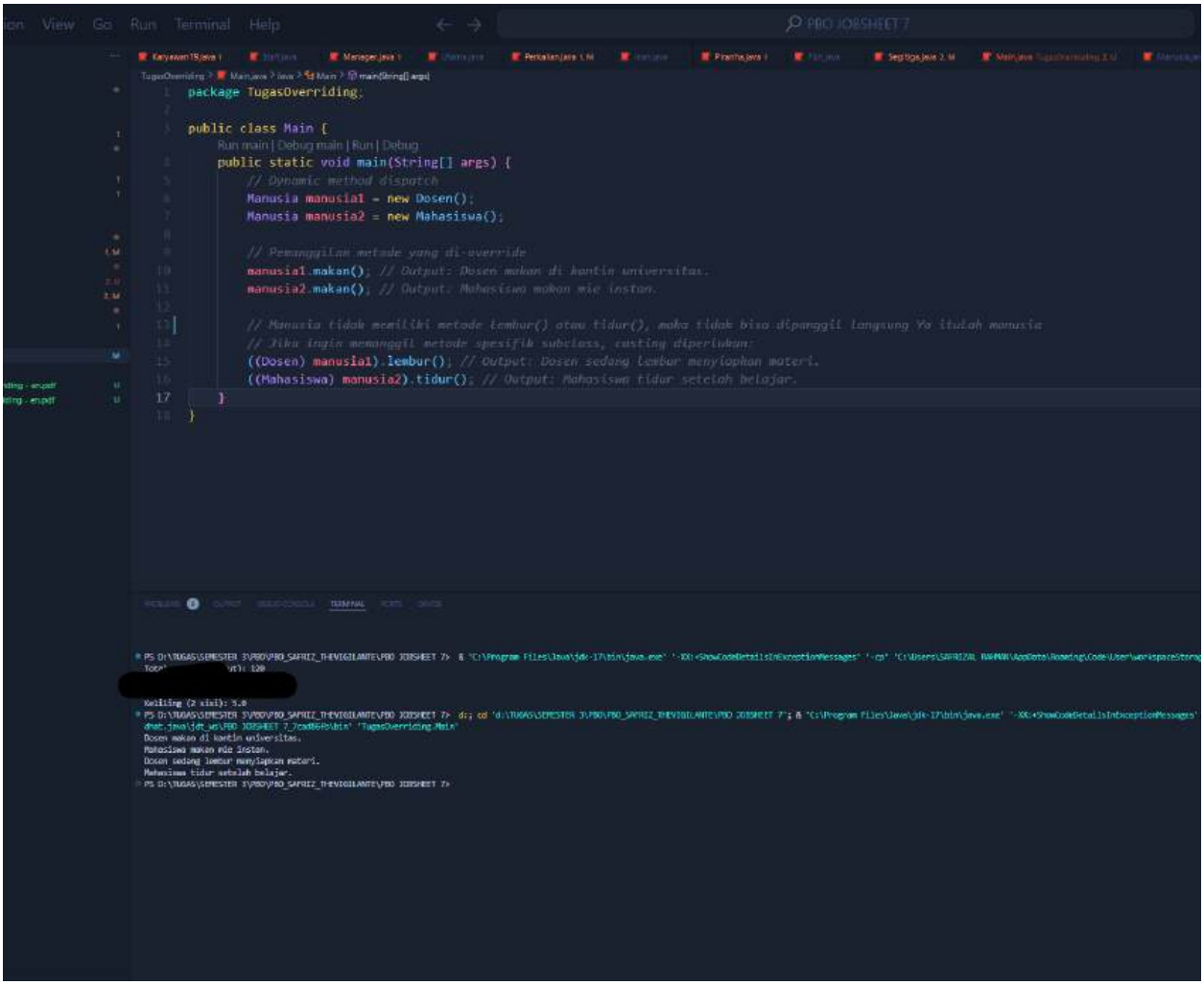


```java
package TugasOverriding;

public class Dosen extends Manusia {
    // @Override
    public void makan() {    Add @Override Annotation
        System.out.println("Dosen makan di kantin universitas.");
    }

    public void lembur() {
        System.out.println("Dosen sedang lembur menyiapkan materi.");
    }
}
```



```java
package TugasOverriding;

public class Mahasiswa extends Manusia {
    @Override
    public void makan() {
        System.out.println("Mahasiswa makan mie instan.");
    }

    public void tidur() {
        System.out.println("Mahasiswa tidur setelah belajar.");
    }
}
```


Karyawan19.java 1
Staff.java
Manager.java 1
Utama.java
Perkalian.java 1, M
Ikan.java
Piranha.java 1
Fish.java
TugasOverriding > Manusia.java > Language Support for Java(TM) by Red Hat > Manusia > makan()
public class Manusia {
public void bernafas() {
System.out.println(x:"Manusia bisa bernafas.");
}
public void makan() {
System.out.println(x:"Manusia makan makanan.");
}
}
PROBLEMS 9
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
DEVDB


View Go Run Terminal Help
PBO JOBSHEET 7
TugasOverriding > Main.java > Main > main(String[] args)
package TugasOverriding;
public class Main {
Run main | Debug main | Run | Debug
public static void main(String[] args) {
// Dynamic method dispatch
Manusia manusia1 = new Dosen();
Manusia manusia2 = new Mahasiswa();
// Pemanggilan metode yang di-override
manusia1.makan(); // Output: Dosen makan di kantin universitas.
manusia2.makan(); // Output: Mahasiswa makan mie instan.
// Manusia tidak memiliki metode lembur() atau tidur(), maka tidak bisa dipanggil langsung Ya itulah manusia
// Jika ingin memanggil metode spesifik subclass, casting diperlukan:
((Dosen) manusia1).lembur(); // Output: Dosen sedang lembur menyiapkan materi.
((Mahasiswa) manusia2).tidur(); // Output: Mahasiswa tidur setelah belajar.
}
}
Keliling (2 sisi): 5.0
Dosen makan di kantin universitas.
Mahasiswa makan mie instan.
Dosen sedang lembur menyiapkan materi.
Mahasiswa tidur setelah belajar.