JOBSHEET 7

OVERLOADING AND OVERRIDING

Name	**: Hanin Mariam Abiyyah Hendrik**
Class	**: 2G - SIB**
NIM	**: 2341760154**

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:
    - o Function_member (int x, string n)
    - o 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:
    - o function_member(int x)
    - o 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)

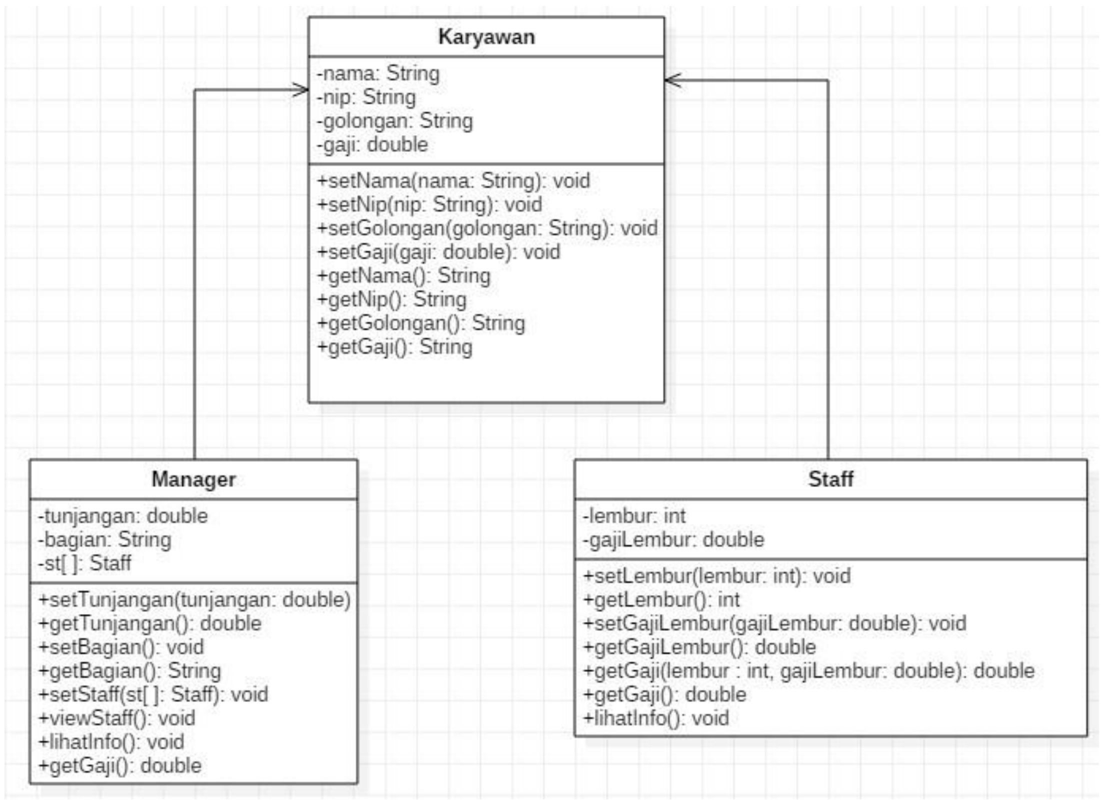
So that the method in the parent class is called the overridden method and the method in the subclass is called the overriding method. There are several method rules in overriding:

- 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.

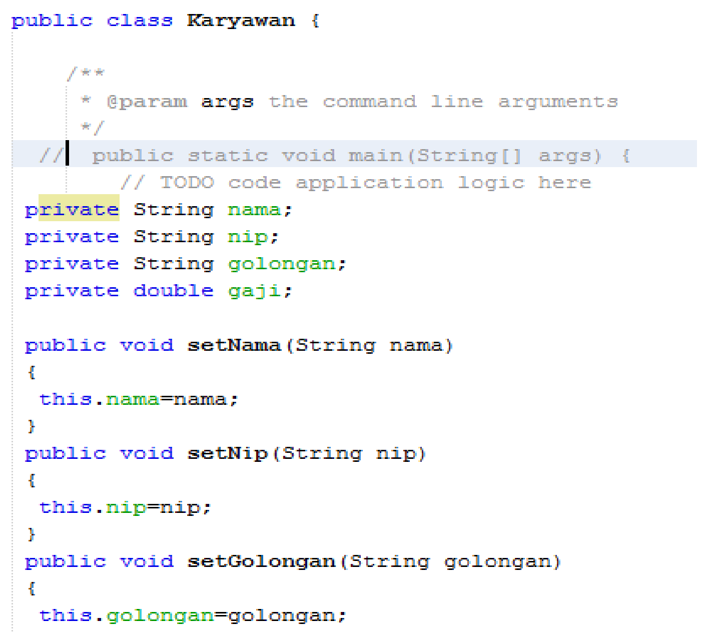
3. Practicum

3.1 Experiment 1

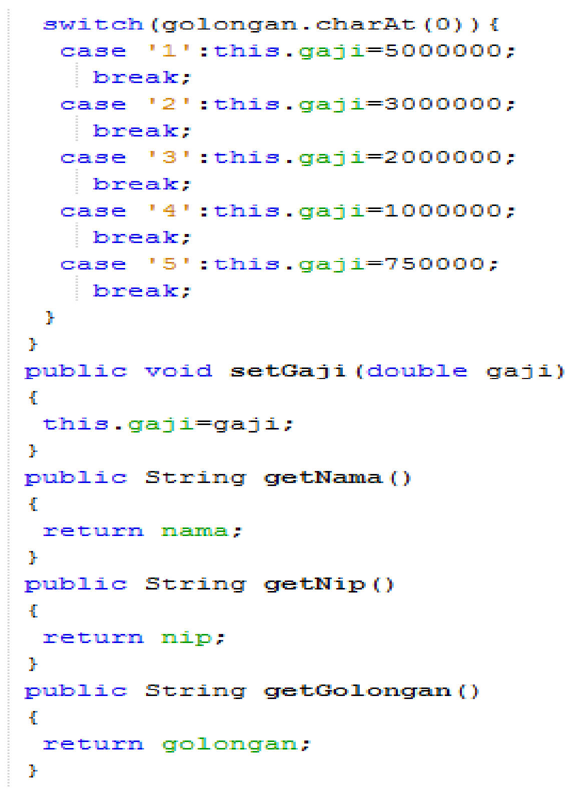
For the following example case, there are three classes, namely Karyawan, Manager, and Staff. Employee Class is a superclass of Manager and Staff where the Manager and Staff subclasses have different methods for calculating salaries.



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;
}
```

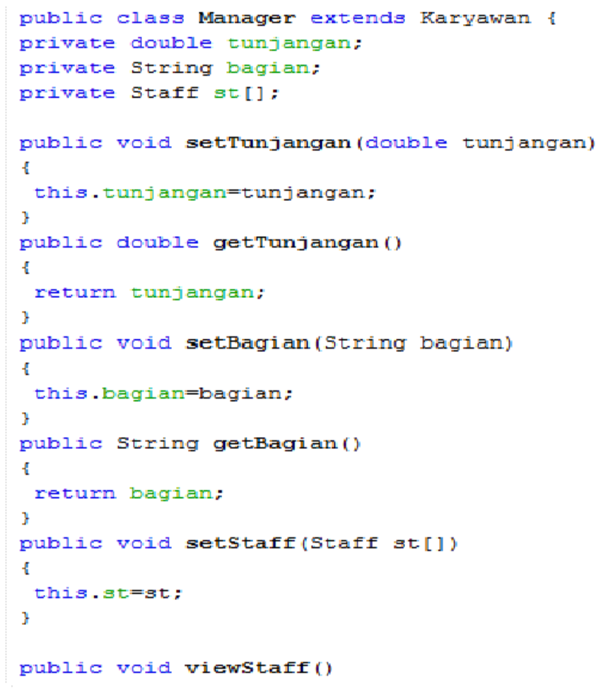
Overloading

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

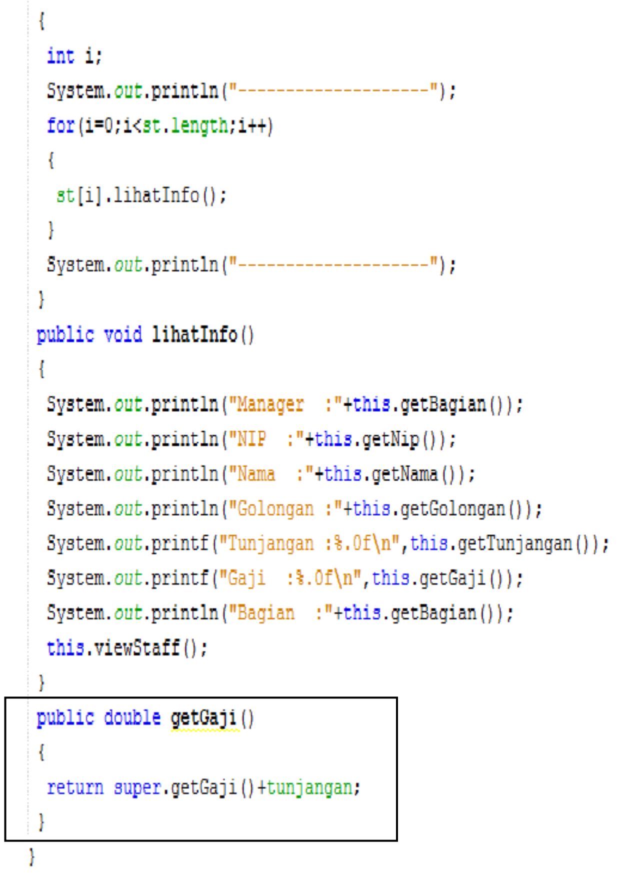
Overriding

```
 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());
 }
}
```

3.4 Manager

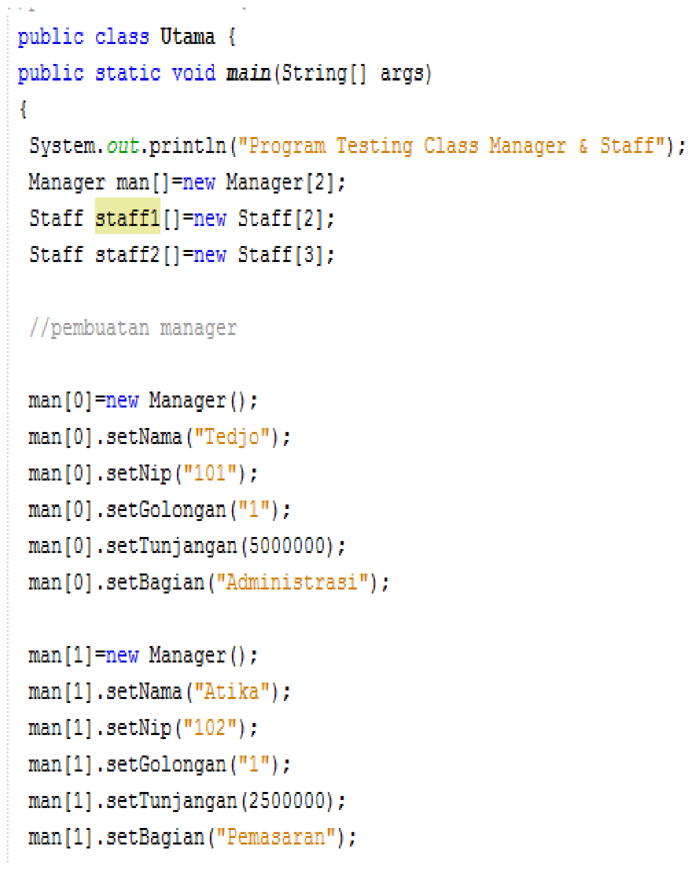


```
public class Manager extends Karyawan {
private double tunjangan;
private String bagian;
private Staff st[];

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()
```

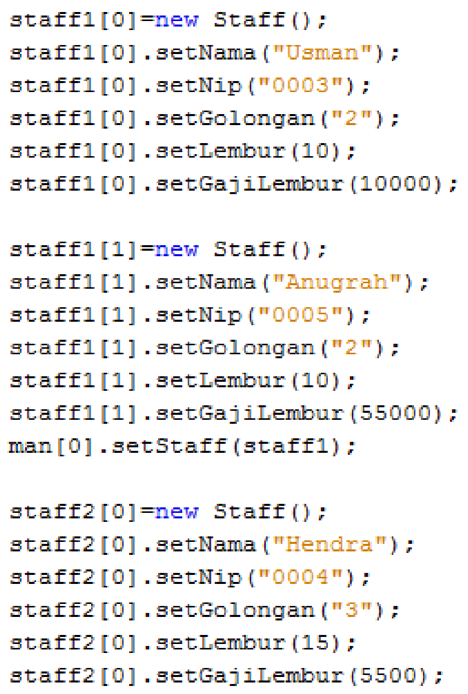


```
{
 int i;
 System.out.println("--------------------");
 for(i=0;i<st.length;i++)
 {
  st[i].lihatInfo();
 }
 System.out.println("--------------------");
}
public void lihatInfo()
{
 System.out.println("Manager  :"+this.getBagian());
 System.out.println("NIP  :"+this.getNip());
 System.out.println("Nama  :"+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

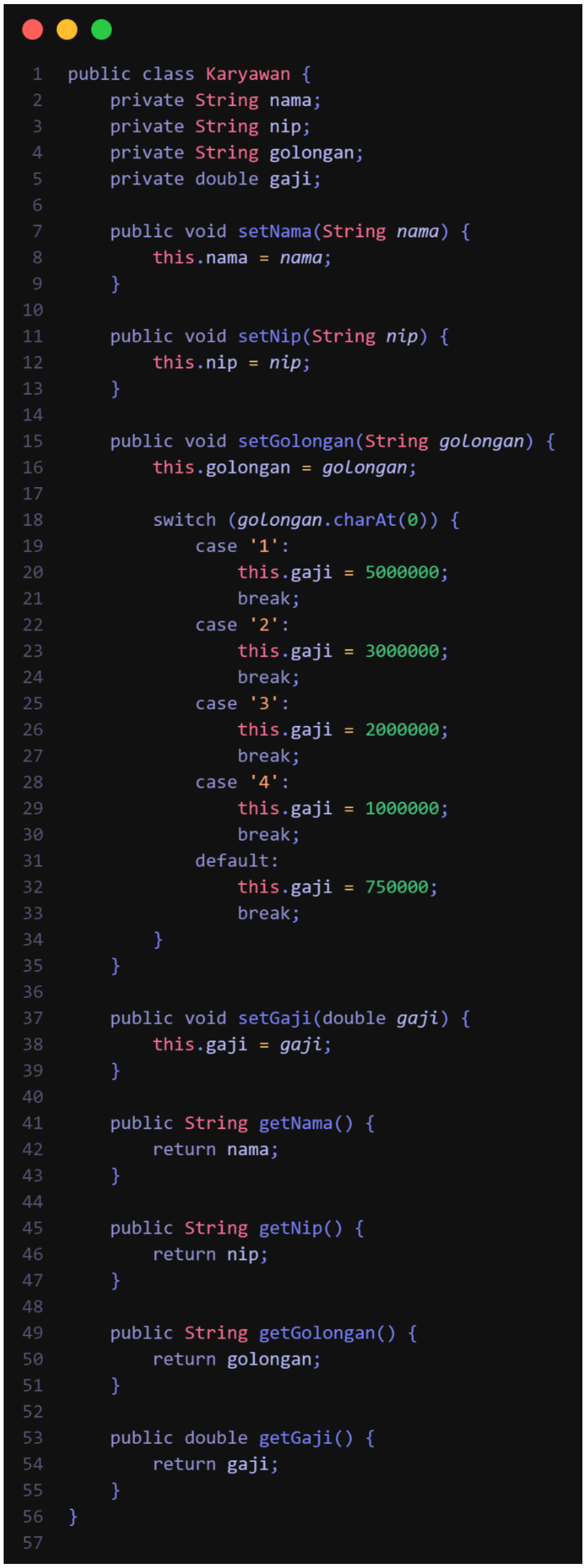


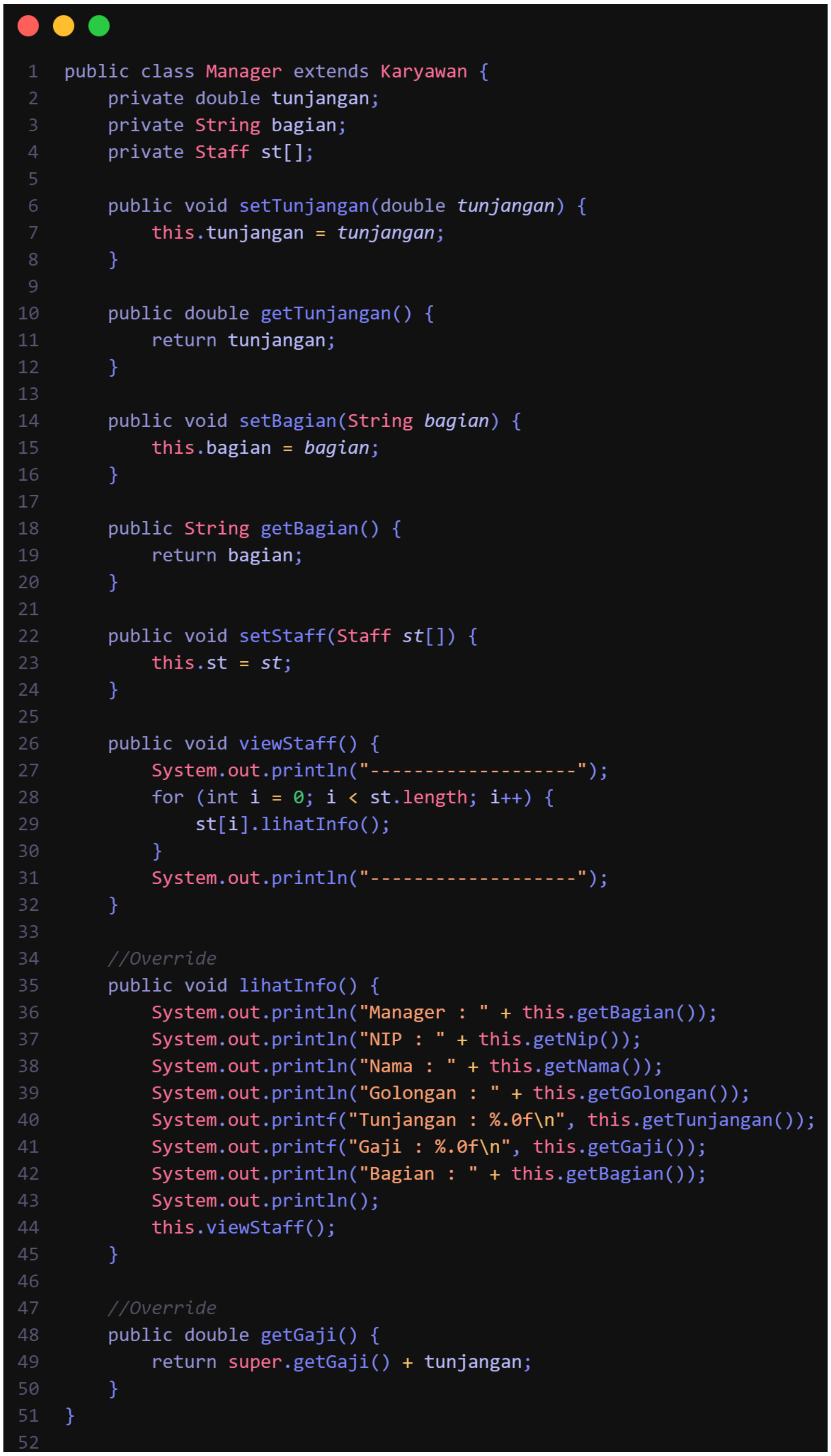
```java
public class Utama {
public static void main(String[] args)
{
 System.out.println("Program Testing Class Manager & Staff");
 Manager man[]=new Manager[2];
 Staff staff1[]=new Staff[2];
 Staff staff2[]=new Staff[3];

 //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");
```



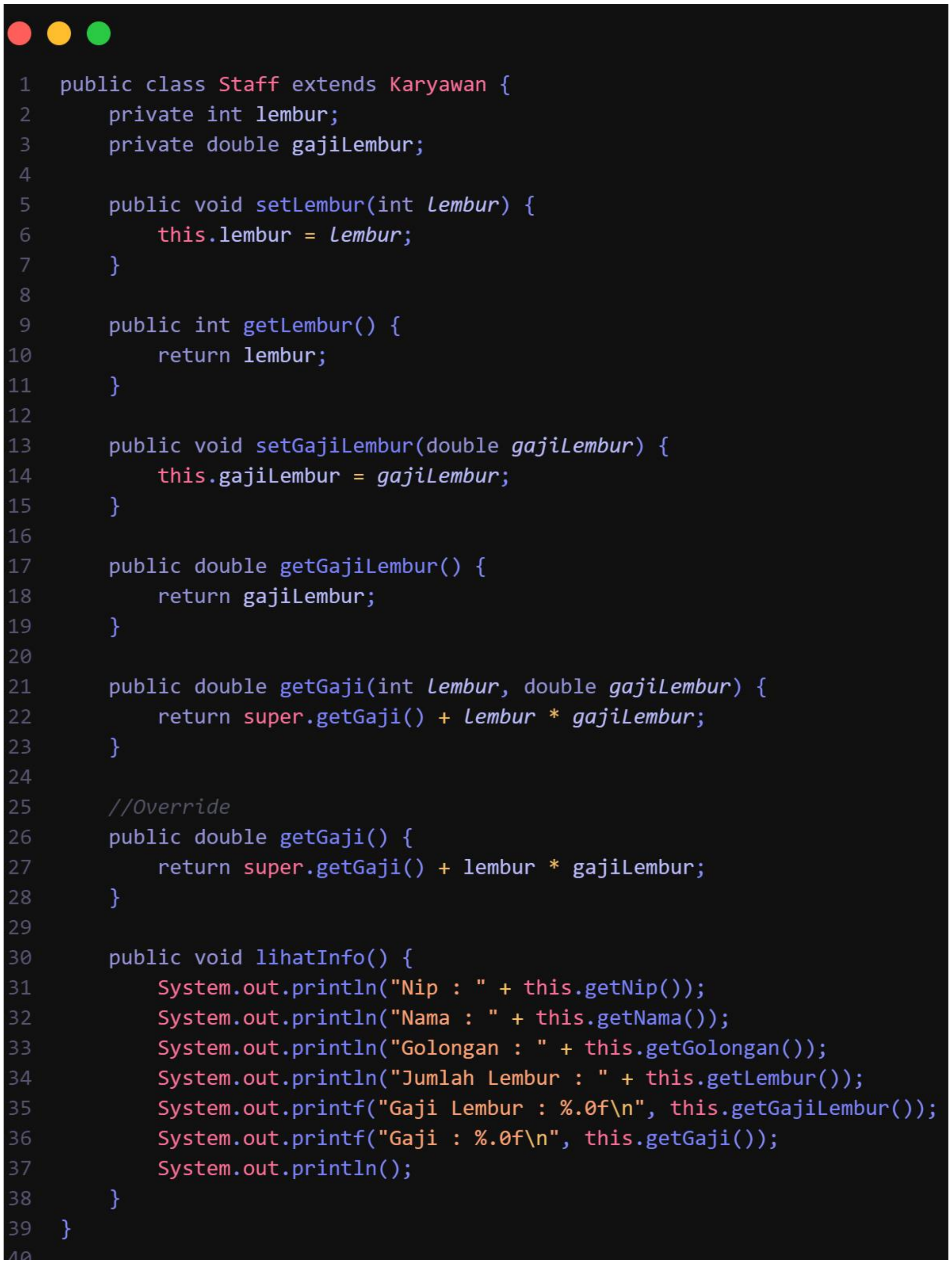
```java
 staff1[0]=new Staff();
 staff1[0].setNama("Usman");
 staff1[0].setNip("0003");
 staff1[0].setGolongan("2");
 staff1[0].setLembur(10);
 staff1[0].setGajiLembur(10000);

 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();
}
```

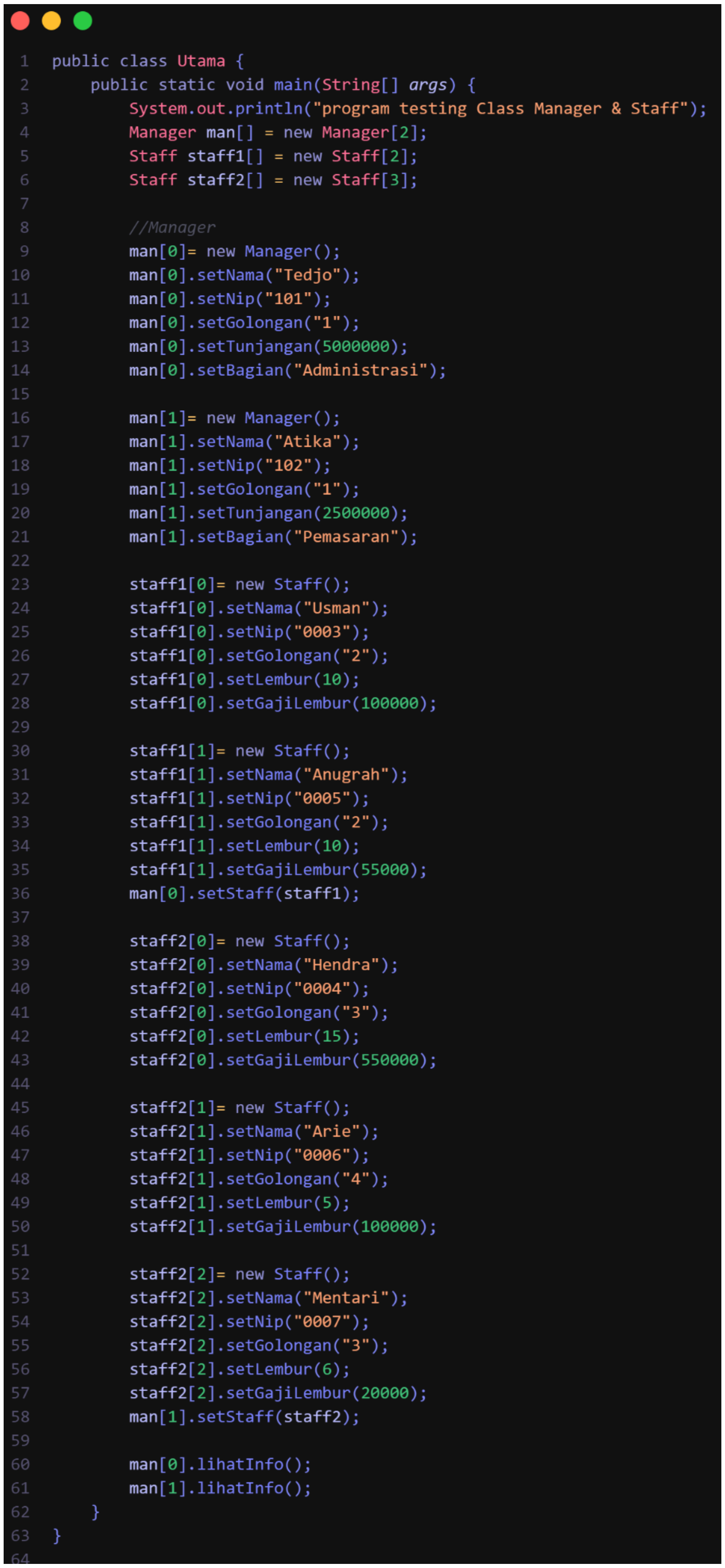

```java
public class Karyawan {
    private String nama;
    private String nip;
    private String golongan;
    private double gaji;

    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;
            default:
                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;
    }
}

```



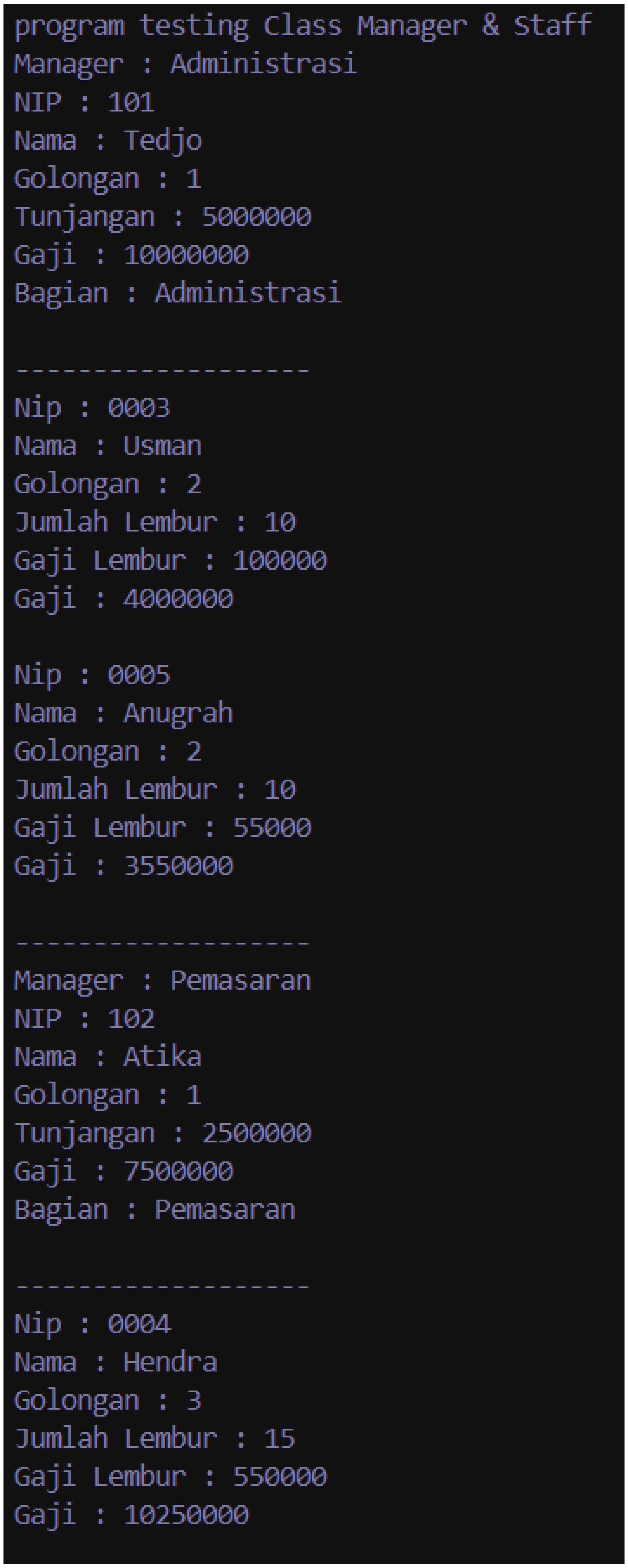
```java
public class Manager extends Karyawan {
    private double tunjangan;
    private String bagian;
    private Staff st[];

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

    //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("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());
        System.out.println();
        this.viewStaff();
    }

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

```



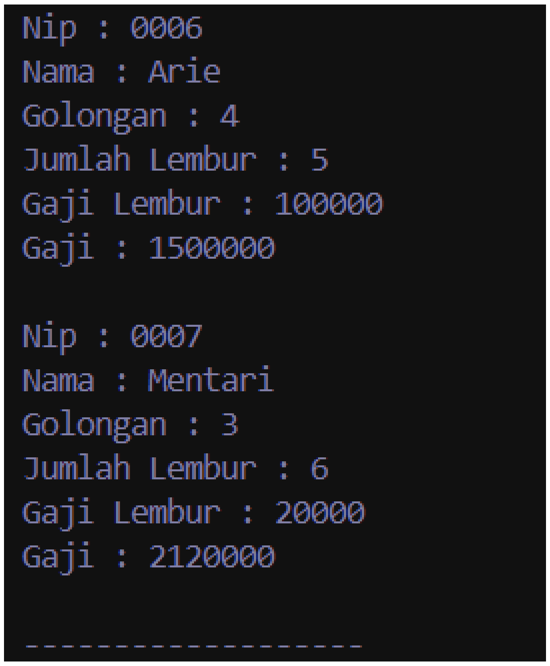
```java
public class Staff extends Karyawan {
    private int lembur;
    private double gajiLembur;

    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;
    }

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



```java
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];

        //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(100000);

        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(550000);

        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);

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



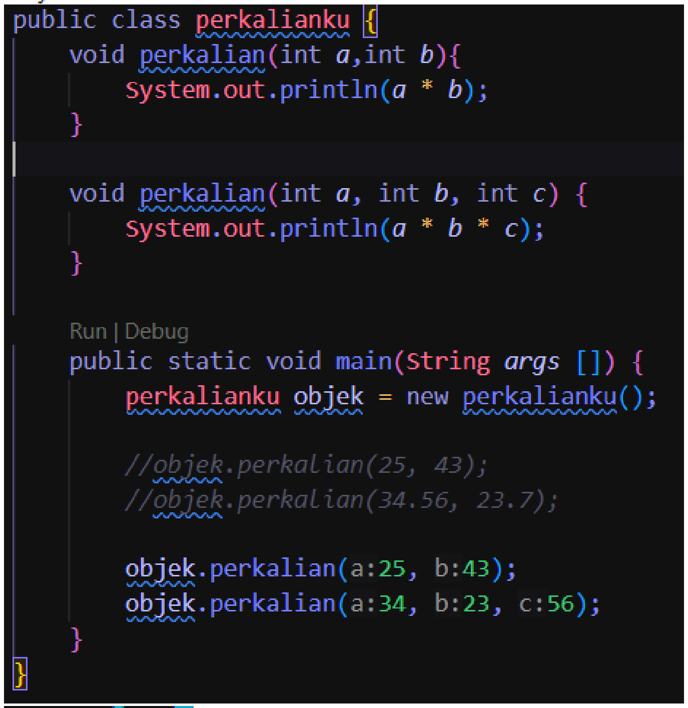
```
program testing Class Manager & Staff
Manager : Administrasi
NIP : 101
Nama : Tedjo
Golongan : 1
Tunjangan : 5000000
Gaji : 10000000
Bagian : Administrasi

-------------------
Nip : 0003
Nama : Usman
Golongan : 2
Jumlah Lembur : 10
Gaji Lembur : 100000
Gaji : 4000000

Nip : 0005
Nama : Anugrah
Golongan : 2
Jumlah Lembur : 10
Gaji Lembur : 55000
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 : 550000
Gaji : 10250000
```



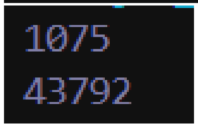
```
Nip : 0006
Nama : Arie
Golongan : 4
Jumlah Lembur : 5
Gaji Lembur : 100000
Gaji : 1500000

Nip : 0007
Nama : Mentari
Golongan : 3
Jumlah Lembur : 6
Gaji Lembur : 20000
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);
 }
}
```



```
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);
    }

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

        //objek.perkalian(25, 43);
        //objek.perkalian(34.56, 23.7);

        objek.perkalian(a:25, b:43);
        objek.perkalian(a:34, b:23, c:56);
    }
}
```

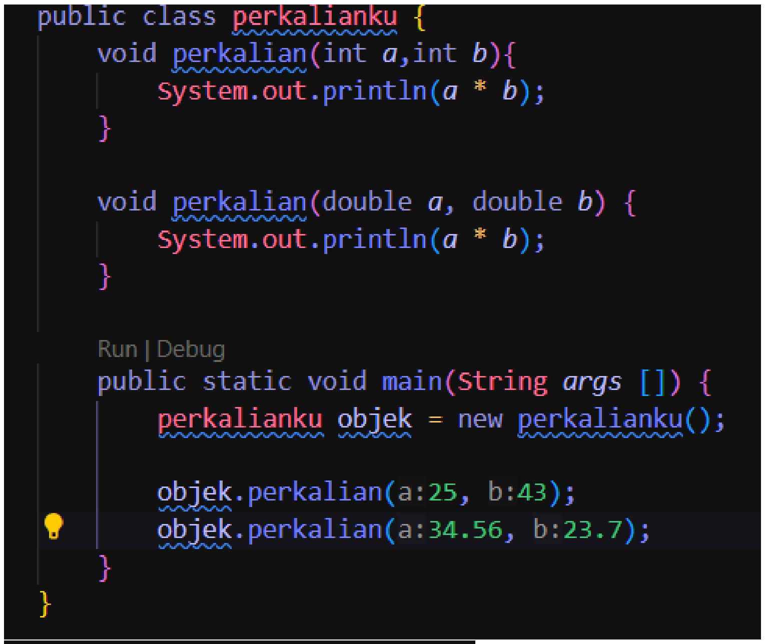


```
1075
43792
```

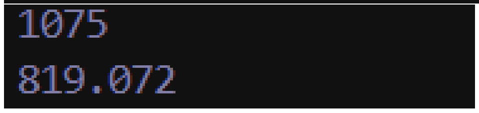
4.1 From the source coding above, where is the overloading?
Overloading occurs in the Perkalianku class. Specifically, it's demonstrated with the two perkalian() methods.

4.2 If there is overloading, how many different parameters are there?
There are two distinct parameter lists for the perkalian() methods:
- Method 1: perkalian(int a, int b) takes two integer parameters.
- Method 2: perkalian(int a, int b, int c) takes three integer 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);
 }
}
```



```
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);
    }

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

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



```
1075
819.072
```

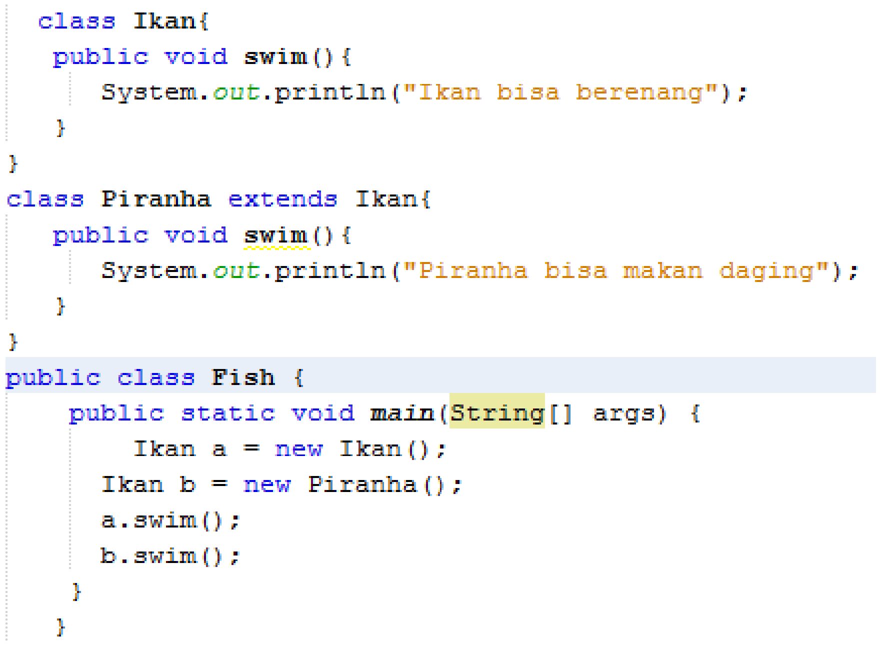
4.3 From the source coding above, where is the overloading?

Method overloading occurs within the Perkalianku class. Specifically, it's demonstrated with the two perkalian() methods.

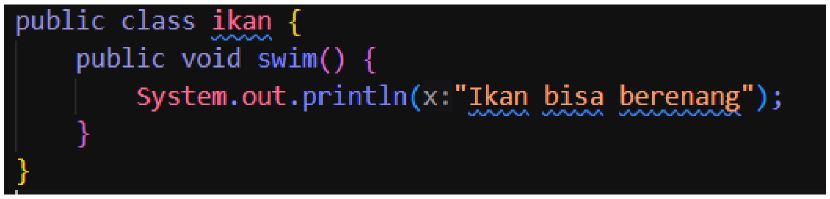
4.4 If there is overloading, how many different types of parameters are there?

There are two distinct parameter lists for the perkalian() methods:

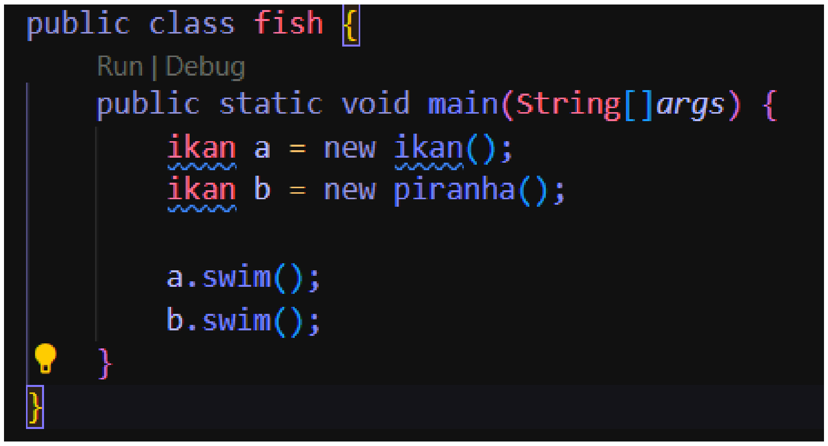
- Method 1: perkalian(int a, int b) takes two integer parameters.
- Method 2: perkalian(double a, double b) takes two double parameters.



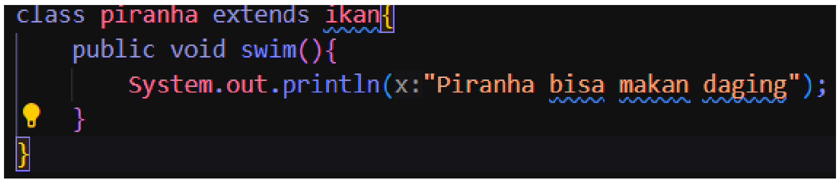
```
  class Ikan{
   public void swim(){
      System.out.println("Ikan bisa berenang");
   }
}
class Piranha extends Ikan{
   public void swim(){
      System.out.println("Piranha bisa makan daging");
   }
}
public class Fish {
    public static void main(String[] args) {
        Ikan a = new Ikan();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



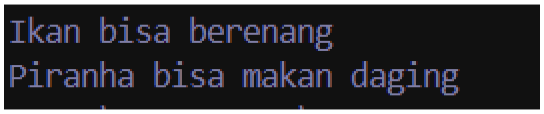
```
public class ikan {
    public void swim() {
        System.out.println(x:"Ikan bisa berenang");
    }
}
```



```
public class fish {
    public static void main(String[]args) {
        ikan a = new ikan();
        ikan b = new piranha();

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



```
class piranha extends ikan{
    public void swim(){
        System.out.println(x:"Piranha bisa makan daging");
    }
}
```



```
Ikan bisa berenang
Piranha bisa makan daging
```

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

Overriding occurs in the swim() method of the Piranha class. This is because the Piranha class inherits the swim() method from the Ikan class, but it provides its own implementation for the method.

4.6 Describe when sourcoding above if there is overriding?

In this code, overriding happens when the Piranha class changes the way the swim() method works compared to how it was originally written in the `Ikan` class.

- The Ikan class has a method called swim() that says, "Ikan bisa berenang" (Fish can swim).
- The Piranha class is a type of Ikan, but it changes the swim() method to say, "Piranha bisa makan daging" (Piranhas can eat meat).

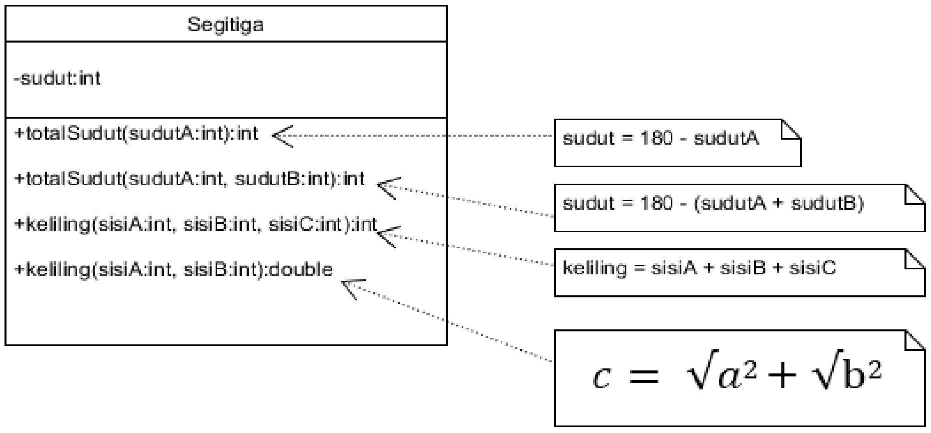
In the main method:

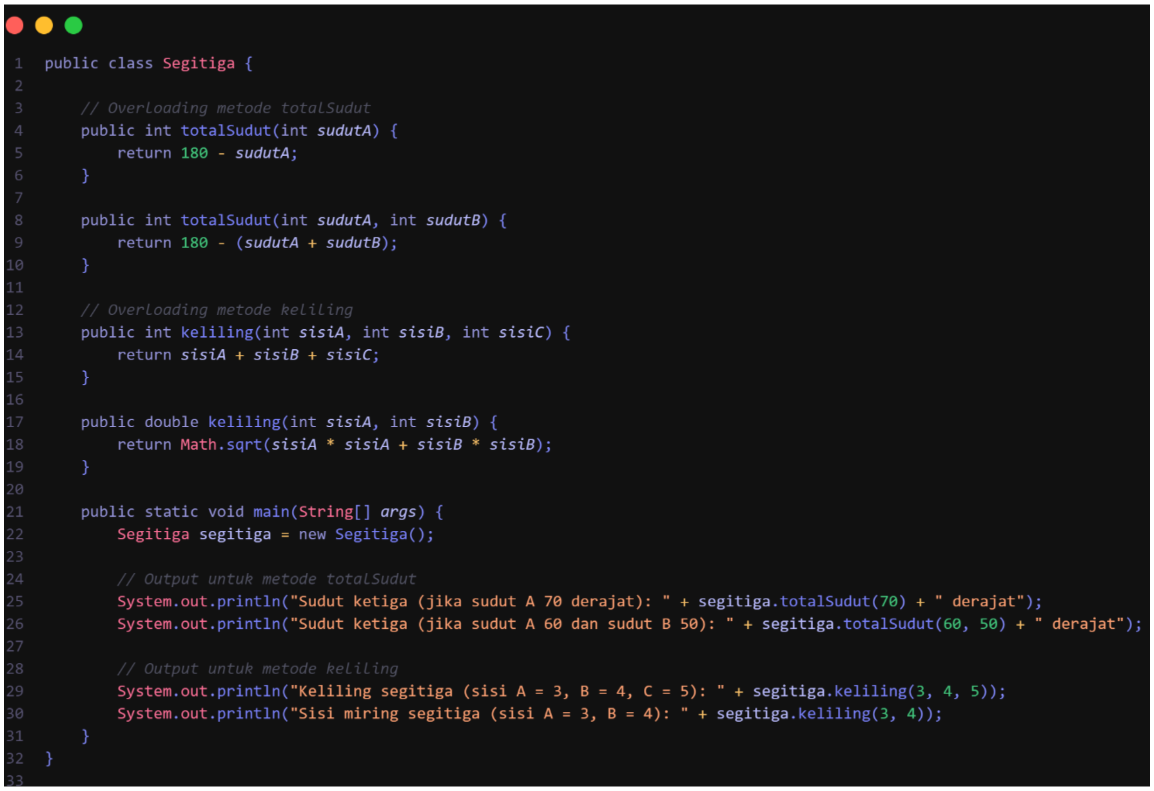
- When a.swim() is called on an Ikan object, it uses the original method from the Ikan class and prints, "Ikan bisa berenang."
- When `b.swim()` is called on a `Piranha` object, even though it's still referred to as an `Ikan`, it uses the `Piranha` version of the `swim()` method and prints, "Piranha bisa makan daging."

5. Tasks

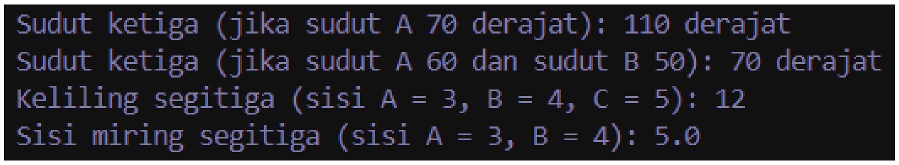
5.1 Overloading

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
$c = \sqrt{a^2} + \sqrt{b^2}$



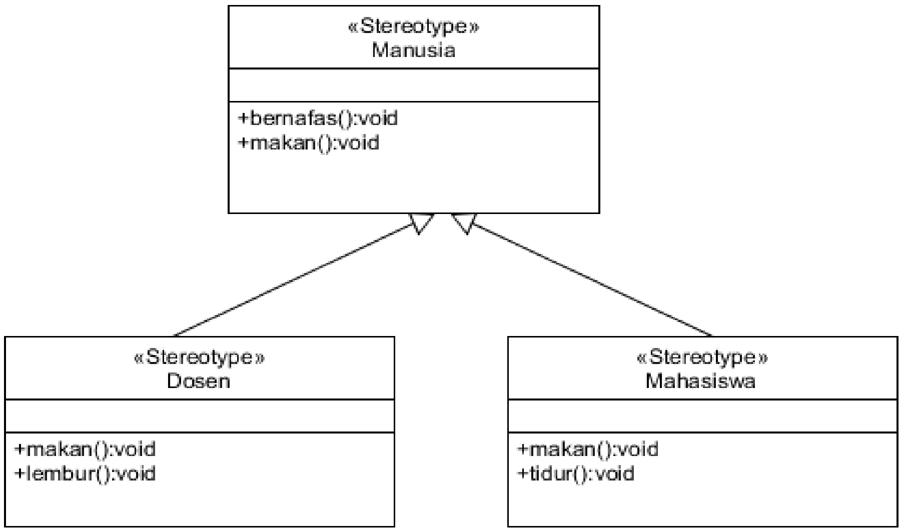
```java
public class Segitiga {

    // Overloading metode totalSudut
    public int totalSudut(int sudutA) {
        return 180 - sudutA;
    }

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

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

    public double keliling(int sisiA, int sisiB) {
        return Math.sqrt(sisiA * sisiA + sisiB * sisiB);
    }

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

        // Output untuk metode totalSudut
        System.out.println("Sudut ketiga (jika sudut A 70 derajat): " + segitiga.totalSudut(70) + " derajat");
        System.out.println("Sudut ketiga (jika sudut A 60 dan sudut B 50): " + segitiga.totalSudut(60, 50) + " derajat");

        // Output untuk metode keliling
        System.out.println("Keliling segitiga (sisi A = 3, B = 4, C = 5): " + segitiga.keliling(3, 4, 5));
        System.out.println("Sisi miring segitiga (sisi A = 3, B = 4): " + segitiga.keliling(3, 4));
    }
}
```

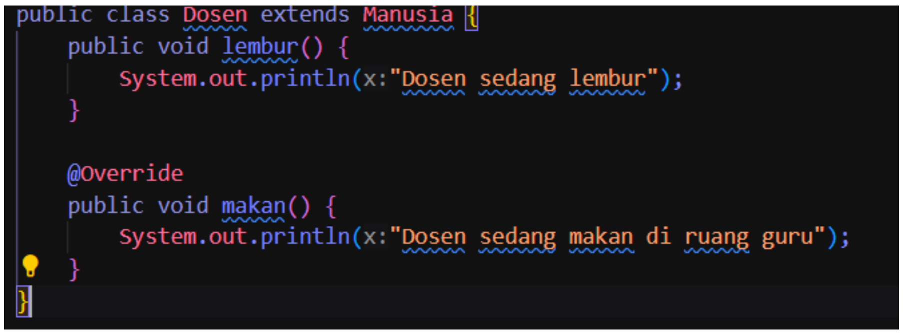


```
Sudut ketiga (jika sudut A 70 derajat): 110 derajat
Sudut ketiga (jika sudut A 60 dan sudut B 50): 70 derajat
Keliling segitiga (sisi A = 3, B = 4, C = 5): 12
Sisi miring segitiga (sisi A = 3, B = 4): 5.0
```

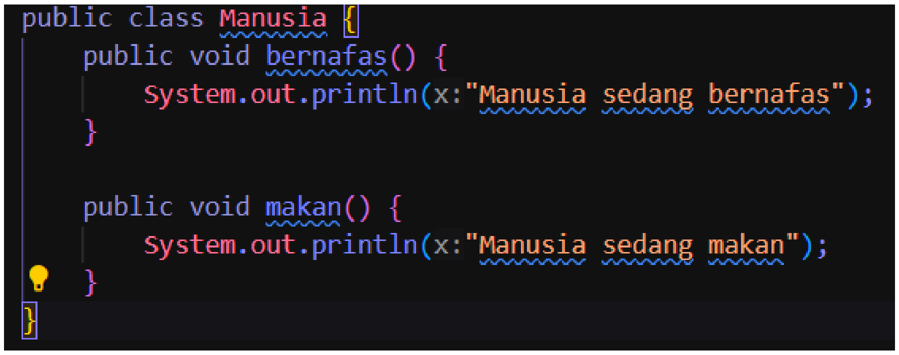
5.2 Overriding

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

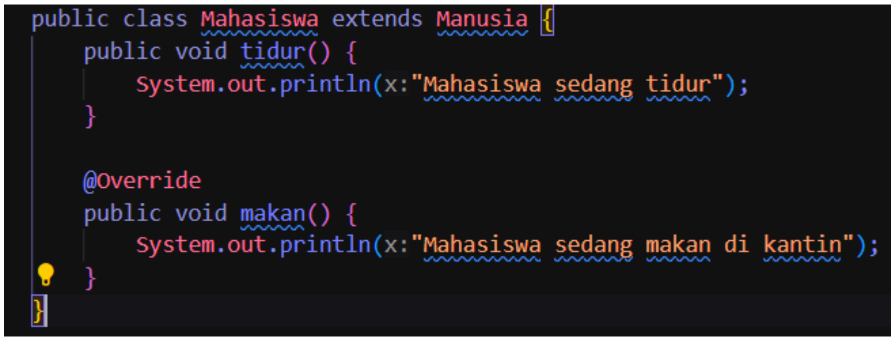




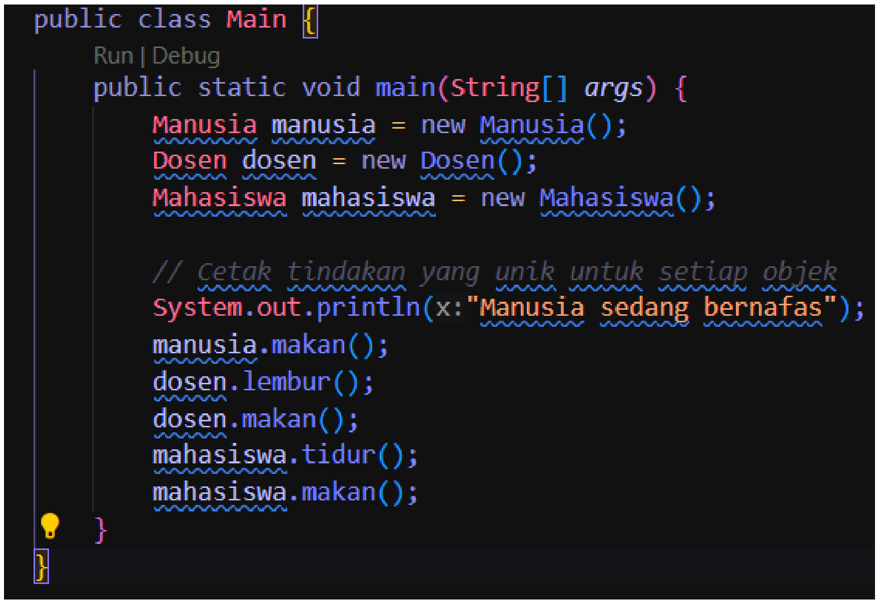
```java
public class Dosen extends Manusia {
    public void lembur() {
        System.out.println(x:"Dosen sedang lembur");
    }

    @Override
    public void makan() {
        System.out.println(x:"Dosen sedang makan di ruang guru");
    }
}
```



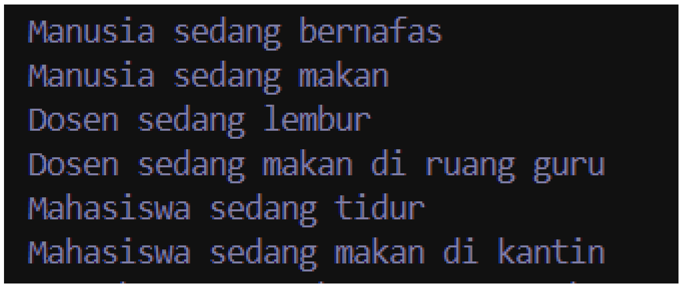
```java
public class Manusia {
    public void bernafas() {
        System.out.println(x:"Manusia sedang bernafas");
    }

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



```java
public class Mahasiswa extends Manusia {
    public void tidur() {
        System.out.println(x:"Mahasiswa sedang tidur");
    }

    @Override
    public void makan() {
        System.out.println(x:"Mahasiswa sedang makan di kantin");
    }
}
```



```
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Manusia manusia = new Manusia();
        Dosen dosen = new Dosen();
        Mahasiswa mahasiswa = new Mahasiswa();

        // Cetak tindakan yang unik untuk setiap objek
        System.out.println(x:"Manusia sedang bernafas");
        manusia.makan();
        dosen.lembur();
        dosen.makan();
        mahasiswa.tidur();
        mahasiswa.makan();
    }
}
```



```
Manusia sedang bernafas
Manusia sedang makan
Dosen sedang lembur
Dosen sedang makan di ruang guru
Mahasiswa sedang tidur
Mahasiswa sedang makan di kantin
```