Overloading & Overriding

@ Name	Davis Maulana Hermanto
Class	TI 2i
# NIM	2241720255
Subject	Object Oriented Programming
Type	Assignment
Semester	Semester 3
Time	@November 7, 2023

Practice

```
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
Jml Lembur :10
Gaji Lembur :10000
Gaji :3100000
NIP :0005
Nama :Anugrah
Golongan :2
Jml 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
Jml Lembur :15
Gaji Lembur :5500
Gaji :2082500
NIP :0006
Nama :Arie
Golongan :4
Jml Lembur :5
Gaji Lembur :100000
Gaji :1500000
NIP :0007
Nama :Mentari
Golongan :3
Jml Lembur :6
Gaji Lembur :20000
Gaji :2120000
----------------
```

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. From the source coding above, where is the overloading located?
    a. The overloading is located at the “perkalian” method.

```
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);

}
```

2. If there is overloading, how many different parameters are there?
    a. There are 2 number of different parameters (the first method has 2 parameters and the second method has 3 parameters)

```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);
 }
}
```

3. From the source coding above, where is the overloading located?

```java
void perkalian(int a, int b){

 System.out.println(a * b);

}

void perkalian(double a, double b){

 System.out.println(a * b);

}
```

4. If there is overloading, how many different types of parameters are there?
    a. There are 2 different types of parameters (the first method has a Int parameter and the second method has a Double parameter)

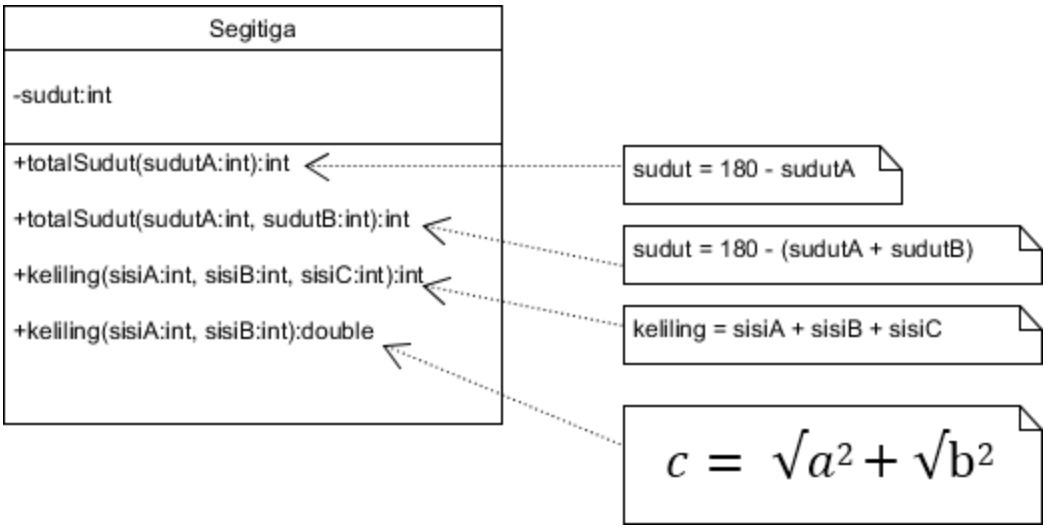
```
class Ikan{
 public void swim(){
    System.out.println("Ikan bisa berenang");
 }
}
class Piranha extends Ikan{
 public void swim(){
    System.out.println("Piranha bisa makan daging");
 }
}
public class Fish {
  public static void main(String[] args) {
     Ikan a = new Ikan();
    Ikan b = new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

5. Where is the overriding located in the source coding above?
    a. at the swim method in Piranha class.
6. Explain if the source coding above if there is overriding?
    a. The swim method is actually overriding because it takes from the parent method.

```
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");
 }
}
```

Tasks

1. Implement the concept of overloading in the class diagram below:

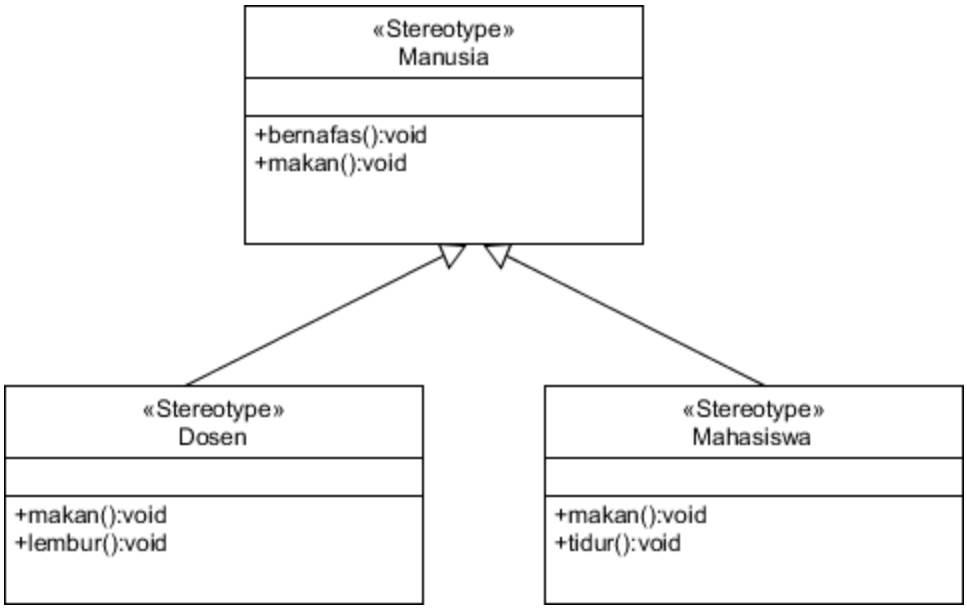



```java
package Tasks.Overloading;

public class Segitiga {
    private int sudut;

    Codeium: Refactor | Explain | Generate Javadoc
    public int totalSudut(int sudutA){
        sudut = 180 - sudutA;
        return sudut;
    }
    Codeium: Refactor | Explain | Generate Javadoc
    public int totalSudut(int sudutA, int sudutB){
        sudut = 180 - (sudutA + sudutB);
        return sudut;
    }
    Codeium: Refactor | Explain | Generate Javadoc
    public int keliling(int sisiA, int sisiB, int sisiC){
        return sisiA + sisiB + sisiC;
    }
    Codeium: Refactor | Explain | Generate Javadoc
    public double keliling(int sisiA, int sisiB){
        return sisiA + sisiB;
    }
}
```

2. Implement the class diagram below using dynamic techniques method dispatch technique:



- Manusia class

```java
package Tasks.Overriding;

public class Manusia {
    Codeium: Refactor | Explain | Generate Javadoc
    public void bernafas(){
        System.out.println(x:"Bernafas");
    }
    Codeium: Refactor | Explain | Generate Javadoc
    public void makan(){
        System.out.println(x:"Makan");
    }
}
```

- Dosen class

```java
package Tasks.Overriding;

public class Dosen extends Manusia{
    Codeium: Refactor | Explain | Generate Javadoc
    public void makan(){
        System.out.println(x:"Dosen makan");
    }
    Codeium: Refactor | Explain | Generate Javadoc
    public void lembur(){
        System.out.println(x:"Dosen lembur");
    }
}
```

- Mahasiswa class

```java
package Tasks.Overriding;

public class Mahasiswa extends Manusia{
    Codeium: Refactor | Explain | Generate Javadoc
    public void makan(){
        System.out.println(x:"Mahasiswa makan");
    }
    Codeium: Refactor | Explain | Generate Javadoc
    public void tidur(){
        System.out.println(x:"Mahasiswa tidur");
    }
}
```