Tugas Praktikum 9

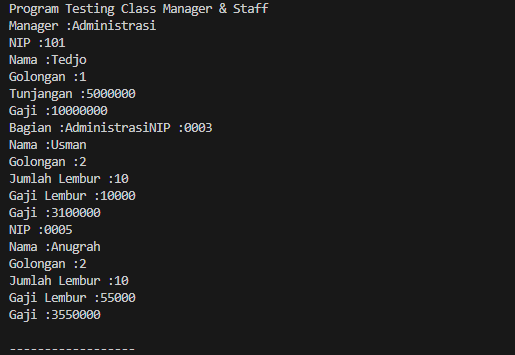
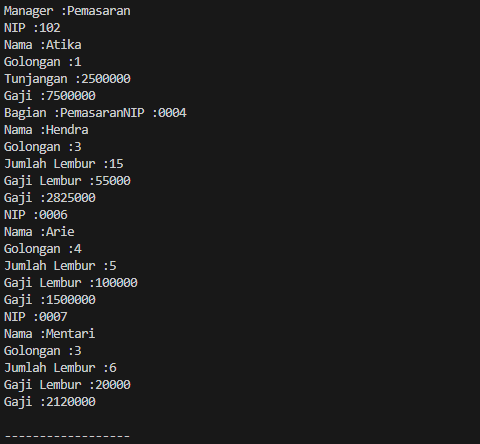
Overriding & Overloading



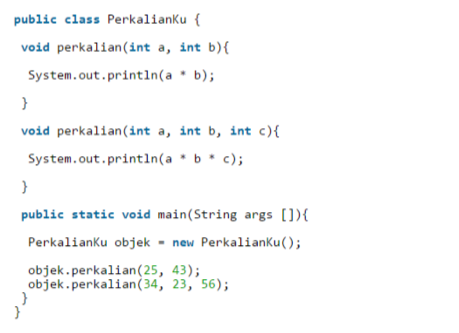
Rafi Ody Prasetyo  
(2341720180)

D-IV Teknik Informatika  
Politeknik Negeri Malang  
Semester 3  
2024

**Percobaan 1**

**** ****

**Latihan**

****

1. Dari source coding diatas terletak dimanakah overloading?

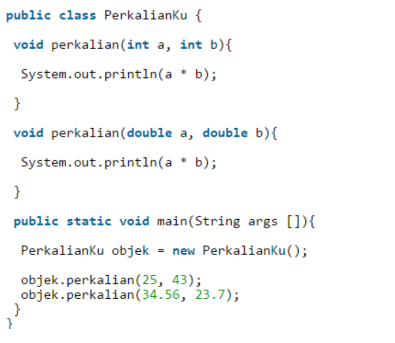
**Jawab:**

Pada kode di atas overloading terletak pada dua method perkalian pada class PerkalianKu. Walaupun dua method tersebut memiliki nama yang sama, namun keduanya memiliki parameter berbeda.

1. Jika terdapat overloading ada berapa jumlah parameter yang berbeda?

**Jawab:**

Terdapat satu parameter yang berbeda, yaitu parameter C.



1. Dari source coding diatas terletak dimanakah overloading?

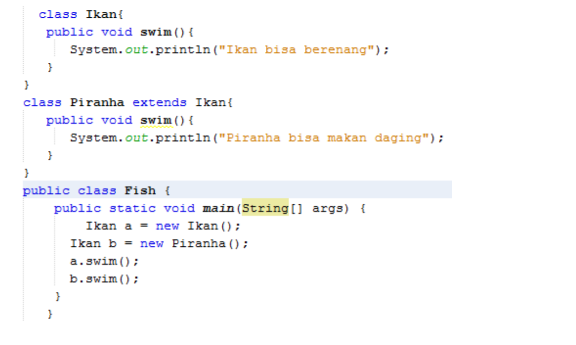
**Jawab:**

Pada kode di atas overloading terletak pada tipe data parameter. Method pertama memiliki parameter dengan tipe data int sedangkan method kedua memiliki parameter dengan tipe data double.

1. Jika terdapat overloading ada berapa tipe parameter yang berbeda?

**Jawab:**

Terdapat dua tipe data parameter yang berbeda int dan double.



1. Dari source coding diatas terletak dimanakah overriding?

**Jawab:**

Overriding pada kode di atas terletak pada method swim yang berada dalam class Piranha.

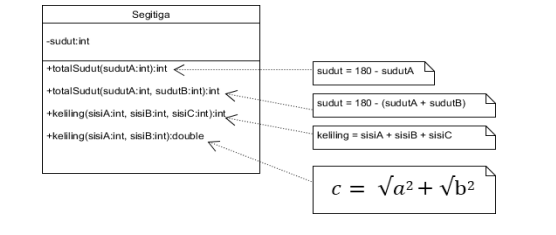
1. Jabarkanlah apabila sourcoding diatas jika terdapat overriding?

**Jawab:**

Dalam class Piranha terdapat method turunan dari class Ikan. Namun, pada class Piranha method tersebut dimodifikasi. Jadi, class Piranha dapat memberikan behavior yang berbeda terhadap method swim() dibandingkan dengan class Ikan.

**Tugas**

1. Implementasikan konsep overloading pada class diagram dibawah ini :



Code:

package Tugas.Overloading;

public class Segitiga {

private int sudut;

public int totalSudut(int sudutA) {

return 180 - sudutA;

}

public int totalSudut(int sudutA, int sudutB) {

return 180 - (sudutA + sudutB);

}

public int keliling(int sisiA, int sisiB, int sisiC) {

return sisiA + sisiB + sisiC;

}

public double keliling(int sisiA, int sisiB) {

double c = Math.sqrt(sisiA \* sisiA) + Math.sqrt(sisiB \* sisiB);

return sisiA + sisiB + c;

}

public void tampil() {

System.out.println("Total sudut 1: " + totalSudut(20));

System.out.println("Total sudut 2: " + totalSudut(15, 10));

System.out.println("Keliling 1: " + keliling(5, 10, 15));

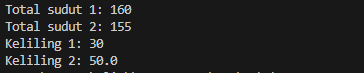
System.out.println("Keliling 2: " + keliling(10, 15));

}

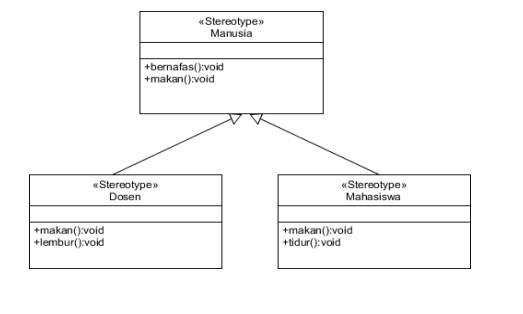
}

Pada kode di atas, overloading terletak pada method totalSudut dan keliling. Masing – masing method memiliki parameter yang berbeda. Pada method keliling ke 2 terdapat fungsi Math.sqrt yang digunakan untuk mencari akar dari nilai sisiA dan sisiB kuadrat.

Output:



1. Implementasikan class diagram dibawah ini dengan menggunakan teknik dynamic method dispatch :



Code:

Manusia.java

package Tugas.Overriding;

public class Manusia {

public void bernafas() {

System.out.println("Manusia bernafas melalui hidung.");

}

public void makan() {

System.out.println("Manusia makan melalui mulut");

}

}

Dosen.java

package Tugas.Overriding;

public class Dosen extends Manusia{

public void makan() {

System.out.println("Dosen sedang makan");

}

public void lembur() {

System.out.println("Dosen sedang lembur");

}

}

Mahasiswa.java

package Tugas.Overriding;

public class Mahasiswa extends Manusia{

public void makan() {

System.out.println("Mahasiswa sedang makan");

}

public void tidur() {

System.out.println("Mahasiswa sedang tidur");

}

}

Main.java

package Tugas.Overriding;

public class MainManusia {

public static void main(String[] args) {

Manusia mns = new Manusia();

Dosen dsn = new Dosen();

Mahasiswa mhs = new Mahasiswa();

mns.bernafas();

dsn.lembur();

mhs.tidur();

mns.makan();

dsn.makan();

mhs.makan();

}

}

Class Dosen dan Mahasiswa merupakan turunan dari class Manusia. Method makan() pada class Dosen dan Mahasiswa memiliki behavior yang berbeda. Karena pada 2 class tersebut dilakukan modifikasi. Apabila tidak ada perubahan pada method makan() dalam 2 class tersebut, maka ketika di panggil hasilnya akan sama dengan behavior yang ada pada superclass.

Output:

