Jobsheet 9

Oleh :

NAME : Maulana Dwi Cahyono

CLASS : 2I

NO.ABSENT: 14

Major : Information Technology

STUDY PROGRAM : Information Engineering

4. Latihan

```
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);
 }
}
```

4.1 Dari source coding diatas terletak dimanakah overloading?

1. void Perkalian(int a, int b, int c): Ini adalah metode pertama yang menerima tiga parameter a, b, dan c.
2. void Perkalian(int a, int b): Ini adalah metode kedua yang hanya menerima dua parameter a dan b.

Jadi, dalam contoh ini, overloading terjadi karena Anda memiliki dua versi metode Perkalian dengan jumlah parameter yang berbeda.

4.2 Jika terdapat overloading ada berapa jumlah parameter yang berbeda?

Dalam kode yang Anda berikan, terdapat dua versi metode Perkalian dengan overloading. Jumlah parameter yang berbeda dalam kedua versi metode adalah sebagai berikut:

Metode pertama memiliki tiga parameter: int a, int b, dan int c.

Metode kedua memiliki dua parameter: int a dan int b.

Jadi, terdapat dua jumlah parameter yang berbeda dalam kedua versi metode tersebut.

```
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);
 }
}
```

4.3 Dari source coding diatas terletak dimanakah overloading?

Dalam source code yang Anda berikan, overloading terjadi pada dua metode `Perkalian`. Overloading terjadi ketika Anda memiliki dua metode dengan nama yang sama, tetapi dengan parameter yang berbeda. Dalam kasus ini, Anda memiliki dua versi metode `Perkalian` dengan parameter yang berbeda:

1. Metode pertama `void Perkalian(int a, int b)` menerima dua parameter tipe data `int`.

2. Metode kedua `void Perkalian(double a, double b)` menerima dua parameter tipe data `double`.

Jadi, overloading terjadi karena Anda memiliki dua versi metode `Perkalian` dengan tipe data parameter yang berbeda.

4.4 Jika terdapat overloading ada berapa tipe parameter yang berbeda?

Dalam kode yang Anda berikan, terdapat dua tipe parameter yang berbeda dalam metode overloading `Perkalian`:

1. Metode pertama `void Perkalian(int a, int b)` menerima dua parameter dengan tipe data `int`.

2. Metode kedua `void Perkalian(double a, double b)` menerima dua parameter dengan tipe data `double`

Jadi, ada dua tipe parameter yang berbeda dalam metode overloading.

```
  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 Dari source coding diatas terletak dimanakah overriding?

Overriding adalah konsep dalam pemrograman berorientasi objek yang terjadi ketika sebuah subclass (turunan) mengganti (meng-override) implementasi metode yang telah didefinisikan dalam superclass (induk) dengan implementasi yang baru. Dalam kode yang Anda berikan, overriding terjadi pada metode `swim`. Berikut ini penjelasannya:

1. Kelas `ikan` memiliki metode `swim`, yang dicetak "ikan bisa berenang".

2. Kelas `piranha` adalah subclass dari `ikan` dan meng-override metode `swim`. Dalam kelas `piranha`, metode `swim` dicetak ulang menjadi "piranha bisa makan daging".

Ketika Anda membuat objek `b` dari kelas `piranha` dan memanggil `b.swim()` dalam method `main`, ini akan mencetak pesan "piranha bisa makan daging" daripada pesan "ikan bisa berenang" yang akan dicetak jika Anda memanggil `a.swim()`.

Jadi, overriding terjadi ketika metode dalam subclass mengganti implementasi metode yang sama dalam superclass.

4.6 Jabarkanlah apabila sourcoding diatas jika terdapat overriding?

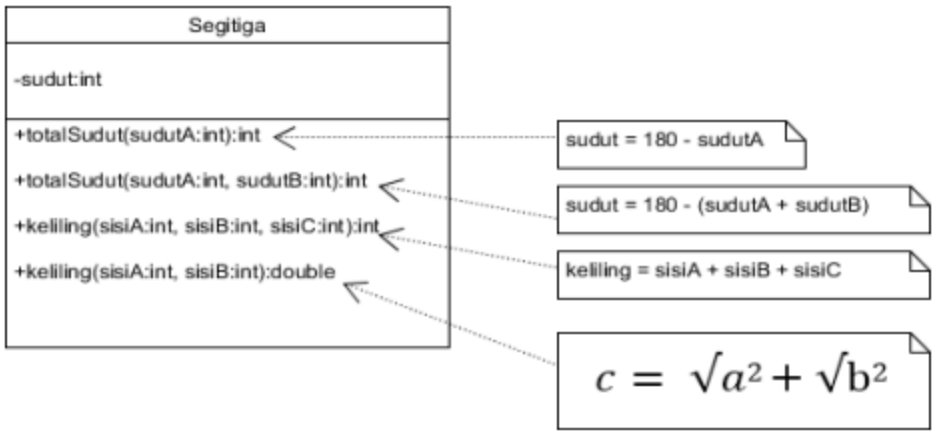
Pada source code yang Anda berikan, tidak ada overriding yang terjadi. Overriding terjadi ketika sebuah subclass (turunan) menyediakan implementasi yang berbeda untuk metode yang telah didefinisikan dalam superclass (induk) yang sama namanya. Overriding melibatkan hubungan antara kelas dan subkelas.

Namun, dalam source code yang Anda berikan, terdapat dua metode `swim` yang berbeda, tetapi mereka ada di dua kelas yang berbeda: `swim` dalam kelas `ikan` dan `swim` dalam kelas `piranha`. Ini bukan contoh overriding, tetapi lebih kepada penentuan implementasi yang berbeda di dua kelas yang berbeda. Dalam kasus ini, metode `swim` dalam kelas `piranha` tidak mengganti metode `swim` dalam kelas `ikan`, sehingga tidak ada overriding yang terjadi.

5. Tugas

5.1 Overloading

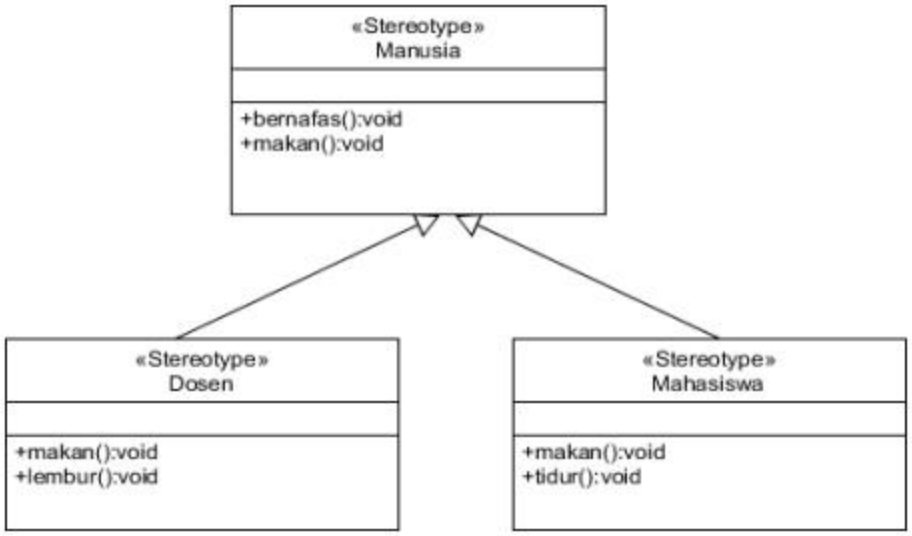
Implementasikan konsep overloading pada class diagram dibawah ini :




```java
package Jobsheet_9;


public class Segitiga {

    int sudut;

    void totalsudut(int sudut_a){
        sudut = 180 - sudut_a;
    }
    void totalsudut(int sudut_a,int sudut_b){
        sudut = 180 - (sudut_a - sudut_b);
    }
    void keliling(int sisi_a,int sisi_b,int sisi_c){
        System.out.println("keliling: " + (sisi_a + sisi_b + sisi_c));
    }
    void keliling(int sisi_a,int sisi_b){
        System.out.println("c: " + (Math.sqrt(Math.pow(sisi_a, sisi_a)) + Math.sqrt(Math.pow(sisi_b, sisi_b))));
    }

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

        // Contoh pemanggilan metode totalsudut dengan satu parameter
        segitiga.totalsudut(60);

        // Contoh pemanggilan metode totalsudut dengan dua parameter
        segitiga.totalsudut(45, 30);

        // Contoh pemanggilan metode keliling dengan tiga parameter
        segitiga.keliling(3, 4, 5);

        // Contoh pemanggilan metode keliling dengan dua parameter
        segitiga.keliling(3, 4);
    }


}

```

5.2 Overriding

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




```java
package Jobsheet_9;

class Manusia{
    void bernafas(){
        System.out.println("bernafas");
    }
    void makan(){
        System.out.println(" manusia bisa makan");
    }
}

class Mahasiswa extends Manusia{
    void makan(){
        System.out.println(" mahasiswa bisa makan");
    }
    void tidur(){
        System.out.println("tidur");
    }
}

class Dosen extends Manusia{
    void makan(){
        System.out.println(" dosen bisa makan");
    }
    void lembur(){
        System.out.println("lembur");
    }
}

public class Stereotype{
    public static void main(String[] args) {
        Manusia manusia1 = new Mahasiswa();
        Manusia manusia2 = new Dosen();

        // Dynamic method dispatch pada metode 'makan'
        manusia1.makan(); // Output: "mahasiswa bisa makan"
        manusia2.makan(); // Output: "dosen bisa makan"

        // Dynamic method dispatch pada metode 'bernafas'
        manusia1.bernafas(); // Output: "bernafas"
        manusia2.bernafas(); // Output: "bernafas"

        // Memanggil metode yang khusus untuk kelas Mahasiswa
        if (manusia1 instanceof Mahasiswa) {
            Mahasiswa mahasiswa = (Mahasiswa) manusia1;
            mahasiswa.tidur(); // Output: "tidur"
        }

        // Memanggil metode yang khusus untuk kelas Dosen
        if (manusia2 instanceof Dosen) {
            Dosen dosen = (Dosen) manusia2;
            dosen.lembur(); // Output: "lembur"
        }
    }
}

```