Name : Faatihurrizki Prasojo

NIM : 244107020142

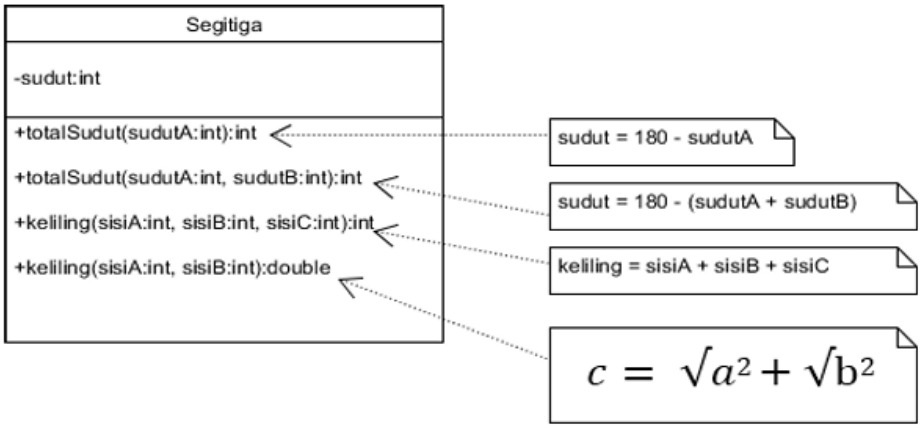
Class : TI_1H

Laporan Tugas Minggu 9

Pemrograman Berorientasi Objek (PBO)

5.1 Overloading

Implementasikan konsep overloading pada class diagram dibawah ini :



Class Hitung

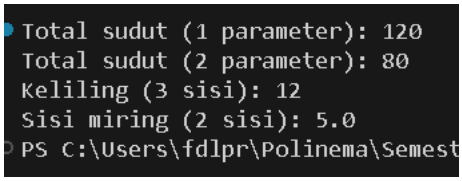
```java
public class Segitiga {
    private int sudut;

    public int totalSudut(int sudutA) {
        sudut = 180 - sudutA;
        return sudut;
    }

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

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

    public double keliling(int sisiA, int sisiB) {
        double c = Math.sqrt(Math.pow(sisiA, 2) + Math.pow(sisiB, 2));
        return c;
    }

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

        System.out.println("Total sudut (1 parameter): " + segitiga.totalSudut(60));
        System.out.println("Total sudut (2 parameter): " + segitiga.totalSudut(60, 40));

        System.out.println("Keliling (3 sisi): " + segitiga.keliling(3, 4, 5));
        System.out.println("Sisi miring (2 sisi): " + segitiga.keliling(3, 4));
    }
}
```

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Output



Penjelasan Singkat :

totalSudut(int sudutA)

- Parameter: 1
- Fungsi: Menghitung sisa sudut segitiga dari 1 sudut
- Return: int

totalSudut(int sudutA, int sudutB)

- Parameter: 2
- Fungsi: Menghitung sisa sudut segitiga dari 2 sudut
- Return: int

keliling(int sisiA, int sisiB, int sisiC)

- Parameter: 3
- Fungsi: Menghitung keliling segitiga
- Return: int

keliling(int sisiA, int sisiB)

- Parameter: 2
- Fungsi: Menghitung sisi miring segitiga siku-siku (rumus Pythagoras)
- Return: double

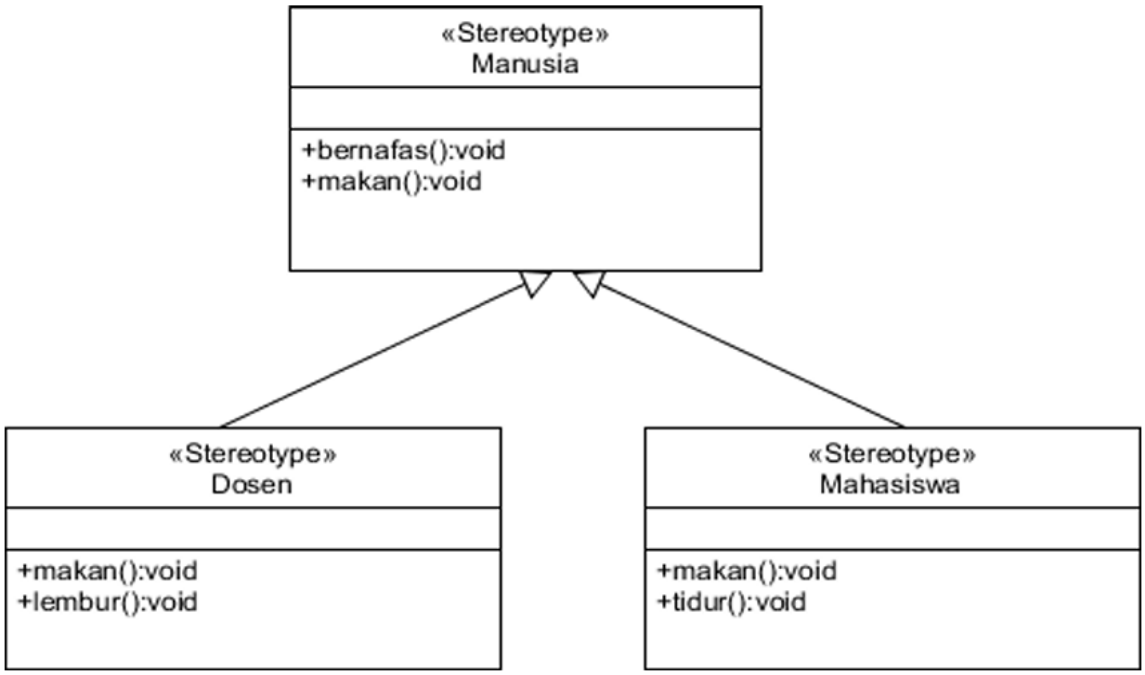
Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

5.2 Overriding

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



class DemoPolymorphism

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

    void makan() {
        System.out.println("Manusia sedang makan...");
    }
}

class Dosen extends Manusia {
    @Override
    void makan() {
        System.out.println("Dosen makan di ruang dosen.");
    }

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

class Mahasiswa extends Manusia {
    @Override
    void makan() {
        System.out.println("Mahasiswa makan di kantin kampus.");
    }

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

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

```java
public class DemoPolymorphism {
    public static void main(String[] args) {
        Manusia manusia;

        manusia = new Dosen();
        manusia.bernafas();
        manusia.makan();

        System.out.println();

        manusia = new Mahasiswa();
        manusia.bernafas();
        manusia.makan();

        System.out.println();

        ((Dosen) new Dosen()).lembur();
        ((Mahasiswa) new Mahasiswa()).tidur();
    }
}
```

Output

```
Manusia sedang bernafas...
Dosen makan di ruang dosen.

Manusia sedang bernafas...
Mahasiswa makan di kantin kampus.

Dosen sedang lembur menyiapkan materi kuliah.
Mahasiswa tidur setelah belajar semalaman.
PS C:\Users\fdlpr\Polinema\Semester 3\PBO\Teori
```

Dynamic Method Dispatch terjadi ketika referensi dari kelas induk menunjuk ke objek kelas turunan, dan method yang dipanggil ditentukan pada **runtime**.

Pada diagram:

- Manusia → superclass
- Dosen dan Mahasiswa → subclass dari Manusia
- Method makan() di-*override* oleh masing-masing subclass.