Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Laporan Tugas Minggu 9

Pemrograman Berorientasi Objek (PBO)

Percobaan 1

Class Karyawan

```java
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;
        }
    }
```

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Staff

```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;
    }

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

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Manager

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

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Utama

```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];

        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("Osman");
        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);

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

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Output

```
Program Testing Class Manager & Staff
Manager :Administrasi
NIP :101
Nama :Tedjo
Golongan :1
Tunjangan :5000000
Gaji :10000000
Bagian :Administrasi
--------------------
NIP :0003
Nama :Osman
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
```

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Latihan

Class ikan

```java
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);
    }
}
```

Output

```
1075
43792
PS C:\User
```

4.1 Dari source coding diatas dimanakah overloading?

= Overloading terjadi pada **dua method** yang bernama **perkalian**.

1. void perkalian(`int a, int b`): Method yang menerima **dua** parameter (`int`).
2. void perkalian(`int a, int b, int c`): Method yang menerima **tiga** parameter (`int`).

4.2 Jika terdapat overloading ada berapa parameter yang berbeda?

= Terdapat **2 (dua)** jenis daftar parameter yang berbeda, yaitu:

1. Daftar parameter dengan **2 parameter** (`int`, `int`).
2. Daftar parameter dengan **3 parameter** (`int`, `int`, `int`).

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Perkalianku

```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);
    }
}
```

Output

```
1075
819.072
PS C:\User
```

4.3 Dari source coding diatas dimanakah overloading?

1. void perkalian(`int a, int b`): Method yang menerima dua parameter bertipe **integer (`int`)**.
2. void perkalian(`double a, double b`): Method yang menerima dua parameter bertipe **floating-point (`double`)**.

4.4 Jika terdapat overloading ada berapa tipe parameter yang berbeda?

= Terdapat **2** jenis daftar tipe parameter yang berbeda:

1. Daftar tipe parameter **(`int`, `int`)**.
2. Daftar tipe parameter **(`double`, `double`)**.

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Ikan, Piranha,, Fish

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

Output

```
Ikan bisa berenang
Piranha bisa makan daging
PS C:\Users\fdlpr\Polinema\Ser
```

4.5 Dari source coding diatas dimanakah overriding?

= Overriding terjadi pada **method swim()** di dalam kelas **Piranha**.

- Kelas induk (**`Ikan`**) memiliki method: `public void swim()`.
- Kelas anak (**`Piranha`**) mendefinisikan ulang method yang sama persis: `public void swim()`.

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

4.6 Jabarkanlah apabila source coding diatas jika terdapat overriding?

= Terdapat *Method Overriding* pada public void swim() karena:

1. **Hubungan Pewarisan:** Kelas Piranha adalah *subclass* dari kelas Ikan (Piranha extends Ikan).
2. **Definisi Ulang:** Kelas Piranha **mendefinisikan ulang** (override) *method* swim() yang diwarisi dari Ikan.
    - *Method* swim() di Ikan mencetak: "Ikan bisa berenang".
    - *Method* swim() di Piranha mencetak: "Piranha bisa makan daging".

Bagaimana Overriding Bekerja (Dilihat dari main):

Ketika program dijalankan:

- **a.swim()**: Variabel a bertipe Ikan dan mengacu pada objek Ikan. Maka, *method* swim() dari kelas **Ikan** dipanggil, menghasilkan *output*: Ikan bisa berenang.
- **b.swim()**: Variabel b bertipe Ikan, tetapi mengacu pada objek **Piranha**. Karena *method* swim() telah di-*override* di Piranha, maka *method* swim() dari kelas **Piranha** yang akan dipanggil, menghasilkan *output*: Piranha bisa makan daging.

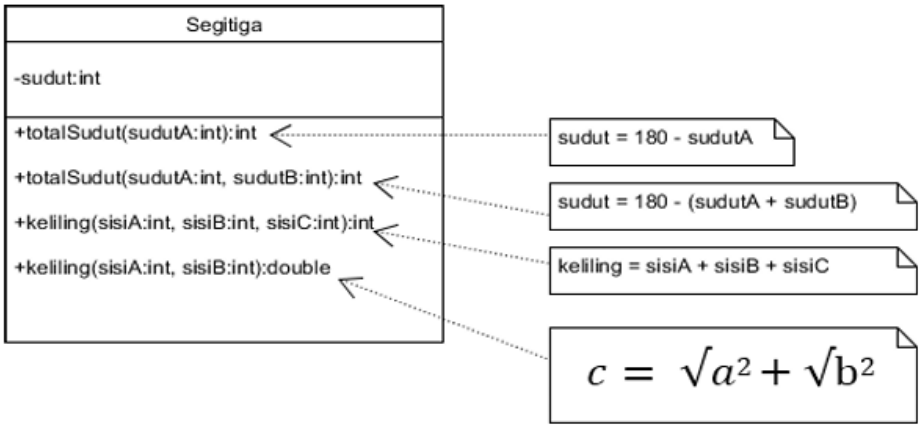
Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

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

```
Total sudut (1 parameter): 120
Total sudut (2 parameter): 80
Keliling (3 sisi): 12
Sisi miring (2 sisi): 5.0
PS C:\Users\fdlpr\Polinema\Semest
```

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

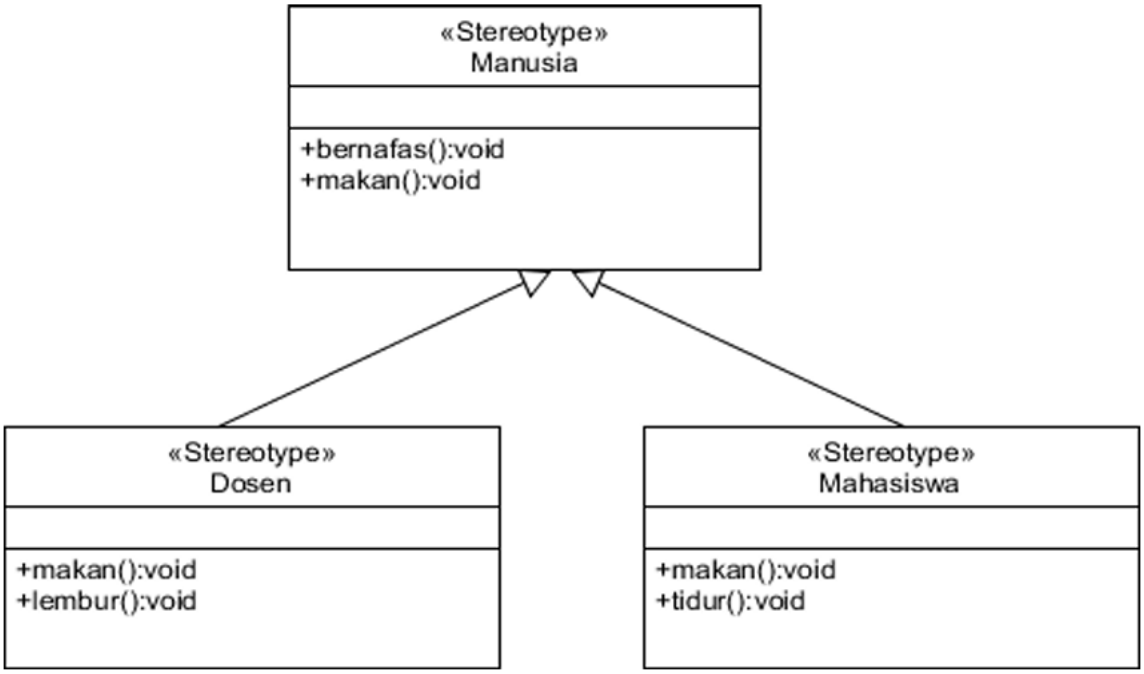
5.2 Overriding

Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

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.