LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
PERTEMUAN 9

2141762056
Margaretha Violina Putri Purnomo
SIB 2F /11

DAFTAR ISI

1. Kompetensi

Setelah menempuh pokok bahasan ini, mahasiswa mampu :

a. Memahami konsep overloading dan overriding,

b. Memahami perbedaan overloading dan overriding,

c. Ketepatan dalam mengidentifikasi method overriding dan overloading

d. Ketepatan dalam mempraktekkan instruksi pada jobsheet

e. Mengimplementasikan method overloading dan overriding.

2. Link Github

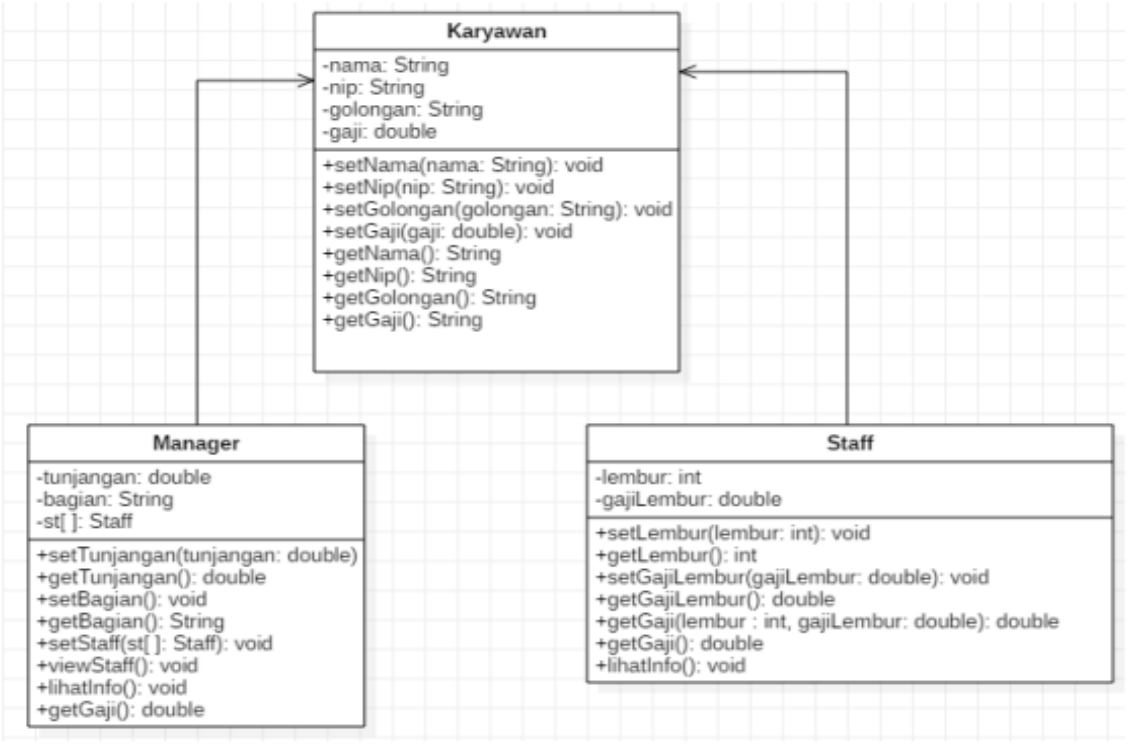
https://github.com/MargarethaViolinaPutri/PBO-

3. Praktikum

2.1 Percobaan 1

2.1.1 Langkah-Langkah Percobaan

Untuk kasus contoh berikut ini, terdapat tiga kelas, yaitu Karyawan, Manager, dan Staff. Class Karyawan merupakan superclass dari Manager dan Staff dimana subclass Manager dan Staff memiliki method untuk menghitung gaji yang berbeda.



Karyawan

```
public class Karyawan {
    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( index: 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;
    }
}

 public void setGaji(double gaji){
    this.gaji=gaji;
 }
 public String getNama(){
     return nama;
 }
 public String getNip(){
     return nip;
 }
 public String getGolongan(){
     return golongan;
 }
 public double getGaji(){
     return gaji;
 }
 }
```

Staff

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

Manager

```
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( x: "---------------");
        for(i=0;i<st.length;i++){
            st[i].lihatInfo();
        }
        System.out.println( x: "---------------");
    }
    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.println("Tunjangan :"+this.getTunjangan());
        System.out.println("Gaji :"+this.getGaji());
        System.out.println("Bagian :"+this.getBagian());
    this.viewStaff();
    }
    public double getGaji(){
        return super.getGaji()+tunjangan;
    }
}
```

Utama

```
public class Utama {
    public static void main(String[]args){
        System.out.println( x: "Program Testing Class Manager & Staff");
        Manager man[]=new Manager[2];
        Staff staff1[]=new Staff[2];
        Staff staff2[]=new Staff[3];

        //Pembuatan Manager
        man[0]=new Manager();
        man[0].setNama( nama: "Tedjo");
        man[0].setNip( nip: "101");
        man[0].setGolongan( golongan: "1");
        man[0].setTunjangan( tunjangan: 5000000);
        man[0].setBagian( bagian: "Administrasii");

        man[1]=new Manager();
        man[1].setNama( nama: "Atika");
        man[1].setNip( nip: "102");
        man[1].setGolongan( golongan: "1");
        man[1].setTunjangan( tunjangan: 2500000);
        man[1].setBagian( bagian: "Pemasaran");

        staff1[0]=new Staff();
        staff1[0].setNama( nama: "Usman");
        staff1[0].setNip( nip: "0003");
        staff1[0].setGolongan( golongan: "2");
        staff1[0].setLembur( lembur: 10);
        staff1[0].setGajiLembur( gajiLembur: 10000);
```

```
staff1[1]=new Staff();
staff1[1].setNama( nama: "Anugrah");
staff1[1].setNip( nip: "0005");
staff1[1].setGolongan( golongan: "2");
staff1[1].setLembur( lembur: 10);
staff1[1].setGajiLembur( gajiLembur: 55000);
man[0].setStaff( st: staff1);

staff2[0]=new Staff();
staff2[0].setNama( nama: "Hendra");
staff2[0].setNip( nip: "0004");
staff2[0].setGolongan( golongan: "3");
staff2[0].setLembur( lembur: 15);
staff2[0].setGajiLembur( gajiLembur: 5500);

staff2[1]=new Staff();
staff2[1].setNama( nama: "Arie");
staff2[1].setNip( nip: "0006");
staff2[1].setGolongan( golongan: "4");
staff2[1].setLembur( lembur: 5);
staff2[1].setGajiLembur( gajiLembur: 100000);

staff2[2]=new Staff();
staff2[2].setNama( nama: "Mentari");
staff2[2].setNip( nip: "0007");
staff2[2].setGolongan( golongan: "3");
staff2[2].setLembur( lembur: 6);
staff2[2].setGajiLembur( gajiLembur: 20000);
man[1].setStaff( st: staff2);

        //cetak informaso dari manager + Staffnya
        man[0].lihatInfo();
        man[1].lihatInfo();
    }
}
```

2.1.2 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);
 }
}
```

1. Dari source coding diatas terletak dimanakah overloading?

Jawab : Terletak dibaris kode berikut ini, karena overloading metode dengan nama yang sama, tetapi memiliki parameter berbeda, dan metode ini berada di kelas yang sama

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

Jawab : Ada dua parameter yang berbeda. parameter pertama yaitu int a, int b dan parameter kedua yaitu int a, int b, int c

```
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. Dari source coding diatas terletak dimanakah overloading?

Jawab : Terletak dibaris kode berikut ini, karena overloading metode dengan nama yang sama, tetapi memiliki parameter berbeda, dan metode ini berada di kelas yang sama

```
void perkalian(int a, int b){
    System.out.println(a*b);
}

void perkalian(double a, double b){
    System.out.println(a*b);
}
```

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

Jawab : Ada dua. Pada method yang pertama terdapat dua parameter dengan tipe data int yaitu int a dan int b. Sedangkan pada method kedua terdapat dua parameter dengan tipe data double yaitu double a dan double b

```
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. Dari source coding diatas terletak dimanakah overriding?

Jawab :

```
public void swim(){
    System.out.println( x: "Ikan bisa berenang");
}

public void swim(){
    System.out.println( x: "Piranha bisa makan daging");
}
```

6. Jabarkanlah apabila sourcoding diatas jika terdapat overriding?

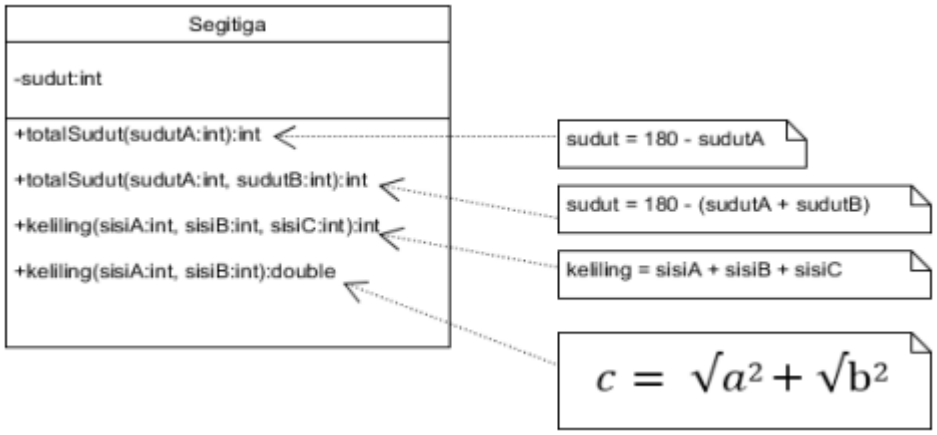
Jawab : Pada class Ikan yang mempunyai method swim() tanpa parameter begitu pula sama dengan class Piranha yang merupakan extends dari class Ikan. Method yang sama yaitu swim() tanpa parameter berisi keterangan “Ikan bisa berenang” di class ikan dan “Piranha bisa makan daging” di class Piranha. Method swim() tersebut adalah overriding

karena nama method sama, terjadi dalam sub kelas/turunan dan memiliki signature/parameter yang sama

TUGAS

Overloading

Implementasikan konsep overloading pada class diagram dibawah ini :



Jawab :

Class Segitiga

```
public class Segitiga {
    private int sudut;

   public int totalSudut(int sudutA){
        this.sudut = 180-sudutA;
        return sudut;
    }
   public int totalSudut(int sudutA, int sudutB){
        this.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(((double)sisiA), b: 2)
                  +Math.pow(((double)sisiB), b: 2));
    return sisiA+sisiB+c;
    }
}
```

Class SegitigaMain

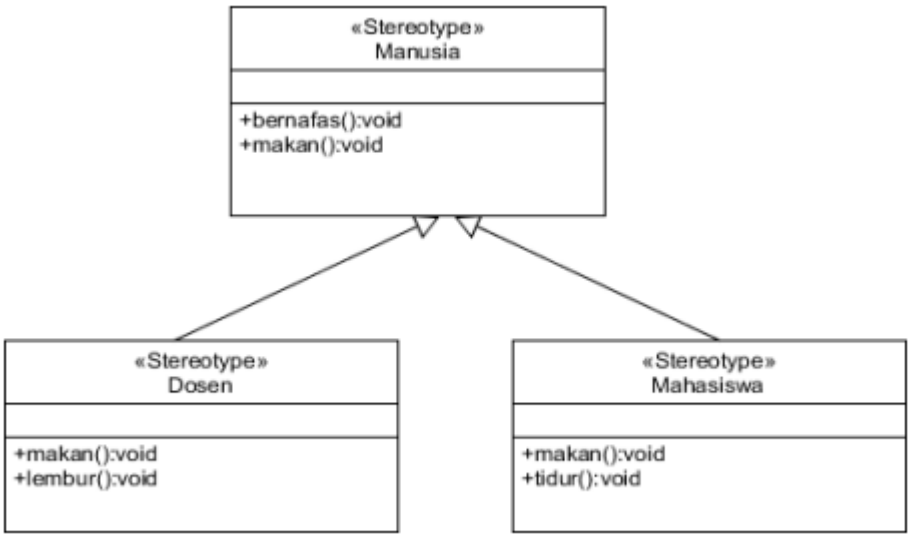
```
public class SegitigaMain {
    public static void main(String[] args){
        Segitiga stg = new Segitiga();
        System.out.println( x: "==== Tugas Overloading Segitiga ====");
        System.out.println("Total sudut 1 parameter:  "+stg.totalSudut( sudutA: 60));

        System.out.println("Total sudut 2 parameter: "+stg.totalSudut( sudutA: 60,  sudutB: 70));

        System.out.println("Keliling 2 parameter   : "+stg.keliling( sisiA: 6,  sisiB: 8));
        System.out.println("Keliling 3 parameter   : "+stg.keliling( sisiA: 6,  sisiB: 8,  sisiC: 10));
    }
}
```

Output

```
--- exec-maven-plugin:3.0.0:exec (default
==== Tugas Overloading Segitiga ====
Total sudut 1 parameter:  120
Total sudut 2 parameter: 50
Keliling 2 parameter   : 24.0
Keliling 3 parameter   : 24
------------------------------------------
BUILD SUCCESS
------------------------------------------
```

Overriding

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



Jawab :

Class Manusia

```
public class Manusia {
    public void bernafas(){
        System.out.println( x: "Manusia selalu bernafas");
    }
    public void makan(){
        System.out.println( x: "Manusia butuh makan");
    }
}
```

Class Dosen

```
public class Dosen extends Manusia{
    public void makan(){
        System.out.println( x: "Dosen sedang makan siang");
    }
    public void lembur(){
        System.out.println( x: "Dosen suka lembur kerja");
    }
}
```

Class Mahasiswa

```
public class Mahasiswa extends Manusia{
    public void makan(){
        System.out.println( x: "Mahasiswa butuh makan");
    }
    public void tidur(){
        System.out.println( x: "Mahasiswa sering tidur di kelas");
    }
}
```

Class Stereotype

```
--- exec-maven-plugin:3.0.0:exec (default-
Manusia selalu bernafas
Manusia butuh makan
Mahasiswa butuh makan
Mahasiswa sering tidur di kelas
Dosen sedang makan siang
Dosen suka lembur kerja
------------------------------------------
BUILD SUCCESS
------------------------------------------
Total time:  2.819 s
Finished at: 2022-11-12T00:15:52+07:00
------------------------------------------
```