Pemrograman Berbasis Objek

Jobsheet 7 Overloading dan Overriding

Dosen Pengampu : Endah Septa Sintiya, S.Pd., M.Kom.

Nama : Annisa

NIM : 2341760032

Kelas : SIB 2C

Prodi : D-IV Sistem Informasi Bisnis

JURUSAN TEKNOLOGI INFORMASI

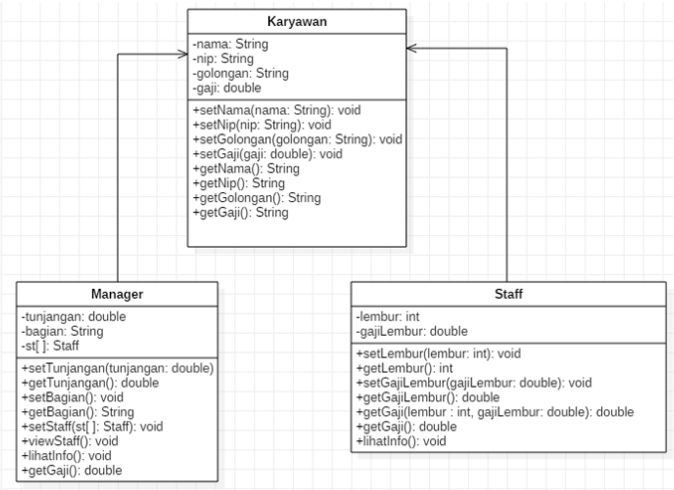
POLITEKNIK NEGERI MALANG

2024/2025

Praktikum

Percobaan 1

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.



Berikut merupakan kode program dari class Karyawan, Manager, Staff, Utama

```java
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(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;
    }
}
```

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

```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: %.2f\n", this.getTunjangan()); // Ganti %.Of dengan %.2f
        System.out.printf("Gaji    : %.2f\n", this.getGaji());       // Ganti %.Of dengan %.2f
        System.out.println("Bagian: " + this.getBagian());
        this.viewStaff();
    }

    public double getGaji() {
        return super.getGaji() + tunjangan;
    }
}
```

```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];

        // Pembuatan Manager
        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");

        // Pembuatan Staff untuk man[0]
        staff1[0] = new Staff();
        staff1[0].setNama("Usman");
        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); // Mengaitkan staff dengan man[0]

```

```
41          // Pembuatan Staff untuk man[1]
42          staff2[0] = new Staff();
43          staff2[0].setNama("Hendra");
44          staff2[0].setNip("0004");
45          staff2[0].setGolongan("3");
46          staff2[0].setLembur(15);
47          staff2[0].setGajiLembur(5500);
48
49          staff2[1] = new Staff();
50          staff2[1].setNama("Arie");
51          staff2[1].setNip("0006");
52          staff2[1].setGolongan("4");
53          staff2[1].setLembur(5);
54          staff2[1].setGajiLembur(100000);
55
56          staff2[2] = new Staff(); // Menggunakan staff2[2] untuk staff ketiga
57          staff2[2].setNama("Mentari");
58          staff2[2].setNip("0007");
59          staff2[2].setGolongan("3");
60          staff2[2].setLembur(6);
61          staff2[2].setGajiLembur(200000);
62
63          man[1].setStaff(staff2); // Mengaitkan staff dengan man[1]
64
65          // Cetak informasi dari manager + staffnya
66          man[0].lihatInfo();
67          man[1].lihatInfo();
68      }
69  }
```

Berikut merupakan output dari kode program di atas

```
Manager: Administrasi
NIP: 101
Nama    : Tedjo
Golongan: 1
Tunjangan: 5000000.00
Gaji    : 10000000.00
Bagian: Administrasi
--------------------
NIP :0003
Nama :Usman
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: 2500000.00
Gaji    : 7500000.00
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
Gaji : 1500000
NIP :0007
Nama :Mentari
Golongan :3
Jml Lembur :6
Gaji Lembur : 200000
Gaji : 3200000
---------------------
```

Latihan

```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);
    }

}
```

Berikut merupakan output dari kode program di atas

```
1075
43792
```

1. **Dari source coding diatas terletak dimanakah overloading?**
   Overloading terletak pada bagian ini

   ```java
   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?**
   Dalam kode tersebut ada 2 parameter yang berbeda, dimana parameter pertama **memiliki 2 parameter**, terletak pada perkalian **(int a, int b)**
   Dan parameter yang kedua memiliki **3 parameter** yang terletak pada perkalian **(int a, int b, int c)**

   Berikut merupakan kode program yang telah dimodifikasi

```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);
    }

}
```

Berikut merupakan output dari kode program di atas

```
1075
819.072
```

1. **Dari source coding diatas terletak dimanakah overloading?**

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

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

2. **Jika terdapat overloading ada berapa tipe parameter yang berbeda?**
   Terdapat dua tipe parameter yang berbeda:
   Tipe Parameter int: perkalian(int a, int b)
   Tipe Parameter double: perkalian(double a, double b)

```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");
    }
}
```

```java
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

D:\Jobsheet 7>
```

1. **Dari source coding diatas terletak dimanakah overriding?**

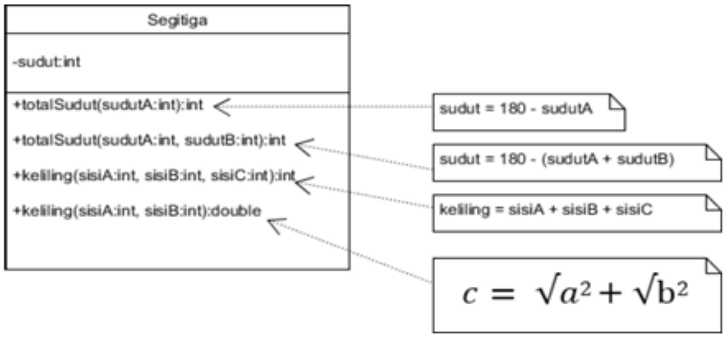
```java
public void swim() {
        System.out.println("Ikan bisa berenang");
    }
}
```

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

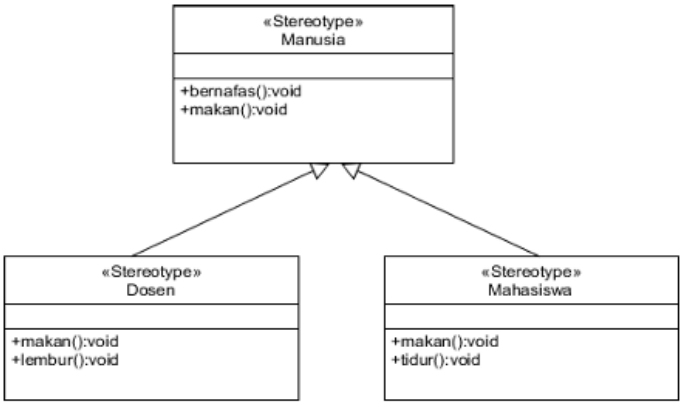
2. **Jabarkanlah apabila sourcoding diatas jika terdapat overriding?**
   kelas Piranha mengganti metode swim() dari kelas Ikan. Kelas Ikan mencetak "Ikan bisa berenang", sementara Piranha mencetak "Piranha bisa makan daging".
   Ketika objek b dideklarasikan sebagai Ikan, tetapi sebenarnya adalah objek Piranha, pemanggilan b.swim() akan menjalankan metode dari kelas Piranha.

Tugas

1. Overloading
   Implementasikan konsep overloading pada class diagram dibawah ini :



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



1. Berikut merupakan kode program dari case tersebut
   Disini saya menggunakan 2 class yaitu Segitiga dan SegitigaMain

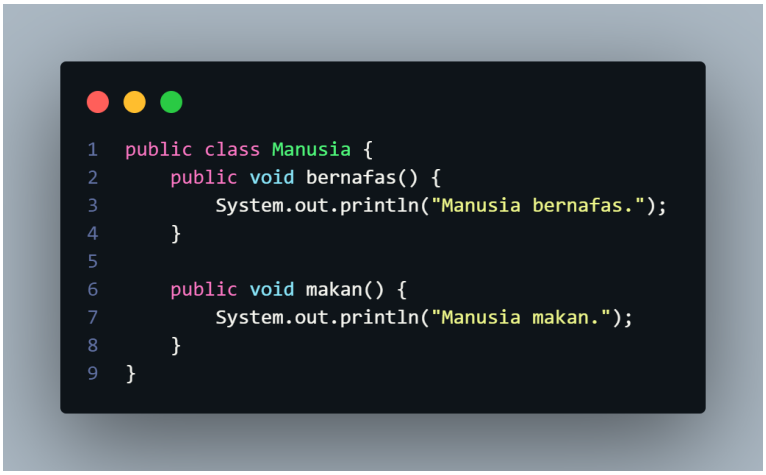
```java
public class Segitiga {

    // Method untuk menghitung total sudut segitiga
    public int totalSudut(int sudutA, int sudutB) {
        return 180 - (sudutA + sudutB);
    }

    // Overloaded untuk menghitung total sudut dengan satu parameter
    public int totalSudut(int sudut) {
        return 180 - sudut;
    }

    // Method untuk menghitung keliling segitiga
    public double keliling(double sisiA, double sisiB, double sisiC) {
        return sisiA + sisiB + sisiC;
    }

    // Overloaded untuk menghitung keliling segitiga dengan dua sisi
    public double keliling(double sisiA, double sisiB) {
        return sisiA + sisiB + Math.sqrt(sisiA * sisiA + sisiB * sisiB);
    }
}
```

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

        int sudutC = segitiga.totalSudut(60, 30);
        System.out.println("Total Sudut C: " + sudutC);

        double kelilingSegitiga = segitiga.keliling(5.0, 6.0, 7.0);
        System.out.println("Keliling Segitiga: " + kelilingSegitiga);

        double kelilingDuaSisi = segitiga.keliling(3.0, 4.0);
        System.out.println("Keliling dengan dua sisi: " + kelilingDuaSisi);
    }
}
```

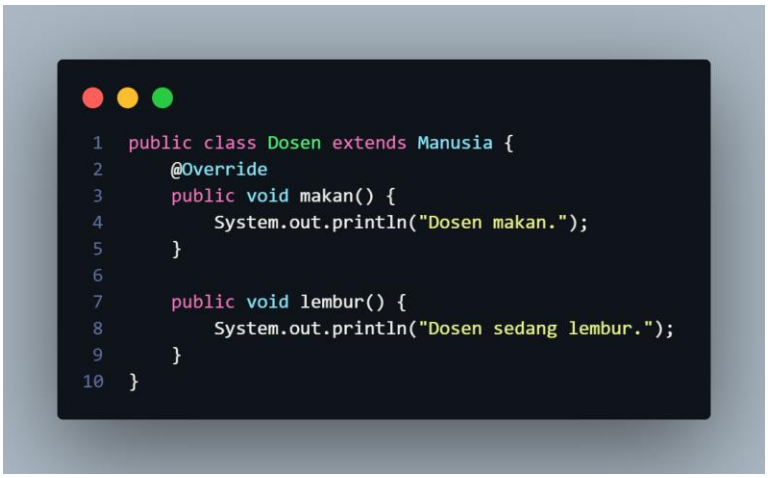
Berikut merupakan outputnya

```
Total Sudut C: 90
Keliling Segitiga: 18.0
Keliling dengan dua sisi: 12.0
```

2. Berikut merupakan kode program yang memiliki 4 class berupa Manusia, Dosen, Mahasiswa, dan Main



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

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



```java
public class Dosen extends Manusia {
    @Override
    public void makan() {
        System.out.println("Dosen makan.");
    }

    public void lembur() {
        System.out.println("Dosen sedang lembur.");
    }
}
```

```java
public class Mahasiswa extends Manusia {
    @Override
    public void makan() {
        System.out.println("Mahasiswa makan.");
    }

    public void tidur() {
        System.out.println("Mahasiswa sedang tidur.");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Manusia manusia = new Manusia();
        Manusia dosen = new Dosen();
        Manusia mahasiswa = new Mahasiswa();

        // Memanggil method dari class Manusia
        manusia.bernafas();
        manusia.makan();

        // Memanggil method dengan dynamic method dispatch
        dosen.makan();
        ((Dosen) dosen).lembur(); // Memanggil lembur() dari Dosen

        mahasiswa.makan();
        ((Mahasiswa) mahasiswa).tidur(); // Memanggil tidur() dari Mahasiswa
    }
}
```

Berikut merupakan outputnya

```
Manusia bernafas.
Manusia makan.
Dosen makan.
Dosen sedang lembur.
Mahasiswa makan.
Mahasiswa sedang tidur.
```