Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-9 : Overloading and Overriding

Mata Kuliah Object Oriented Programming

Pengampu: Tim Ajar Object Oriented Programming

Desember 2023

Nama	:	MAULIA BALQIS ANSYA AULIA
Nim	:	2241720246
Kelas	:	TI – 2I

LEMBAR JAWABAN JOBSHEET-9

Experiment 1

1

Percobaan1 > J Karyawan.java > ...

```java
package Percobaan1;
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;
    }
}
```

Percobaan1 > Staff.java > ...

```java
package Percobaan1;
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;
    }

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

Percobaan1 > J Manager.java > ...

```java
package Percobaan1;
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.printf(format:"Tunjangan: %.0f\n", this.getTunjangan());
        System.out.printf(format:"Gaji: %.0f\n", this.getGaji());
        System.out.println("Bagian: " + this.getBagian());
        this.viewStaff();
    }

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

```
e\bin' 'Percobaan1.Utama'
Program Testing Class Manager & Staff
Manager: Administrasi
NIP: 101
Nama: Tedjo
Golongan: 1
Tunjangan: 5000000
Gaji: 10000000
Bagian: Administrasi
---------------------------
NIP: 0003
Nama: Usman
Golongan: 2
Jumlah Lembur: 10
Gaji Lembur: 10000
Gaji: 3100000
NIP: 0005
Nama: Anugrah
Golongan: 2
Jumlah 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
Jumlah Lembur: 15
Gaji Lembur: 5500
Gaji: 2082500
NIP: 0006
Nama: Arie
Golongan: 4
Jumlah Lembur: 5
Gaji Lembur: 100000
Gaji: 1500000
NIP: 0007
Nama: Mentari
Golongan: 3
Jumlah Lembur: 6
Gaji Lembur: 20000
Gaji: 2120000
---------------------------
```

Exercise	
4.1	Overloading terjadi ketika ada beberapa metode dalam satu kelas dengan nama yang sama tetapi parameter yang berbeda. Pada soal tersebut, overloading terdapat pada kelas **Perkalianku** dengan metode **perkalian** yang memiliki parameter yang berbeda.
4.2	Ada dua metode yang di-overload dengan daftar parameter yang berbeda: • **perkalian(int a, int b)** • **perkalian(int a, int b, int c)** Jadi, terdapat dua set parameter yang berbeda, menunjukkan adanya dua metode yang di-overload.
4.3	Overloading terjadi ketika terdapat dua metode dengan nama yang sama, tetapi memiliki tipe parameter yang berbeda. Pada soal tersebut, overloading terdapat pada kelas **Perkalianku** dengan metode **perkalian** yang memiliki satu versi dengan parameter **int** dan satu versi dengan parameter **double**.
4.4	Terdapat dua tipe parameter yang berbeda: • **int** • **double** Jadi, ada dua tipe parameter yang berbeda, menunjukkan adanya dua metode yang di-overload.
4.5	Overriding terjadi pada metode **swim** di kelas **Piranha**. Overriding adalah ketika sebuah subclass memberikan implementasi yang berbeda untuk metode yang sudah didefinisikan di superclass.
4.6	Overriding terjadi pada metode **swim** di kelas **Piranha** yang menggantikan implementasi metode yang sama di kelas **Ikan**. • Metode **swim** dari kelas **Ikan** didefinisikan sebagai "Ikan bisa berenang".

	• Kemudian, kelas **Piranha** meng-override metode **swim** tersebut dengan implementasi yang berbeda, yaitu "Piranha bisa makan daging". • Saat objek **Piranha** diinisialisasi menggunakan tipe referensi **Ikan** (dengan **Ikan b = new Piranha();**), pemanggilan **b.swim();** akan menggunakan implementasi **swim** dari kelas **Piranha** karena objek yang sebenarnya adalah objek **Piranha**.
Assignment	

5.1

Assignment1 > Segitiga.java > ...

```java
package Assignment1;
public class Segitiga {
    private int sudut;

    // Metode overloading 1
    public int totalSudut(int sudutA) {
        sudut = 180 - sudutA;
        return sudut;
    }

    // Metode overloading 2
    public int totalSudut(int sudutA, int sudutB) {
        sudut = 180 - sudutA - sudutB;
        return sudut;
    }

    // Metode overloading 3
    public int keliling(int sisiA, int sisiB, int sisiC) {
        return sisiA + sisiB + sisiC;
    }

    // Metode overloading 4
    public double keliling(int sisiA, int sisiB) {
        // Menghitung panjang sisi c sesuai dengan rumus Pythagoras
        double sisiC = Math.sqrt(Math.pow(sisiA, b:2) + Math.pow(sisiB, b:2));
        return sisiA + sisiB + sisiC;
    }

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

        // Memanggil metode overloading 1
        System.out.println("Total Sudut 1: " + segitiga.totalSudut(sudutA:90));

        // Memanggil metode overloading 2
        System.out.println("Total Sudut 2: " + segitiga.totalSudut(sudutA:45, sudutB:45));

        // Memanggil metode overloading 3
        System.out.println("Keliling 1: " + segitiga.keliling(sisiA:3, sisiB:4, sisiC:5));

        // Memanggil metode overloading 4
        System.out.println("Keliling 2: " + segitiga.keliling(sisiA:6, sisiB:8));
    }
}
```

```
f4a619b\redhat.java\j
Total Sudut 1: 90
Total Sudut 2: 90
Keliling 1: 12
Keliling 2: 24.0
PS C:\OOP\Week9>
```

Pada contoh di atas, kita menggunakan overloading untuk metode **totalSudut** dan **keliling** dengan jumlah dan tipe parameter yang berbeda. Ini memungkinkan kita menggunakan metode yang sesuai dengan kebutuhan pada saat pemanggilan metode.

5.2

Assignment2 > Manusia.java > ...

```java
package Assignment2;
class Manusia {
    public void bernafas() {
        System.out.println(x:"Manusia bernafas");
    }

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

Assignment2 > Dosen.java > ...

```java
package Assignment2;
class Dosen extends Manusia {
    @Override
    public void makan() {
        System.out.println(x:"Dosen makan");
    }

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

Assignment2 > Mahasiswa.java > ...

```java
package Assignment2;
class Mahasiswa extends Manusia {
    @Override
    public void makan() {
        System.out.println(x:"Mahasiswa makan");
    }

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

Assignment2 > DynamicDispatchExample.java > ...

```java
package Assignment2;
public class DynamicDispatchExample {
    Run | Debug
    public static void main(String[] args) {
        // Contoh dynamic method dispatch
        Manusia manusia1 = new Dosen();
        Manusia manusia2 = new Mahasiswa();

        manusia1.bernafas();
        manusia1.makan();

        manusia2.bernafas();
        manusia2.makan();
    }
}
```

```
Manusia bernafas
Dosen makan
Manusia bernafas
Mahasiswa makan
PS C:\OOP\Week9>
```

Masing-masing kelas memiliki metode **makan** yang di-override sesuai dengan class diagram. Pada method **main**, kita menggunakan tipe referensi **Manusia**

	untuk objek **Dosen** dan **Mahasiswa**, sehingga memungkinkan dynamic method dispatch untuk memanggil metode yang sesuai dengan jenis objeknya.