Name : Faricha Aulia
Class : 2I / D-IV Informatics Engineering
NIM : 2141720155

Jobsheet 9
Overloading and Overriding

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. Pendahuluan

2.1 Overloading

Overloading adalah menuliskan kembali method dengan nama yang sama pada suatu class. Tujuannya dapat memudahkan penggunaan/pemanggilan method dengan fungsionalitas yang mirip. Untuk aturan pendeklarasian method Overloading sebagai berikut:

- Nama method harus sama.
- Daftar parameter harus berbeda.
- Return type boleh sama, juga boleh berbeda.

Ada beberapa daftar parameter pada overloading dapat dilihat sebagai berikut:

- Perbedaan daftar parameter bukan hanya terjadi pada perbedaan banyaknya parameter, tetapi juga urutan darai parameter tersebut. Misalnya saja dua buah parameter berikut:
  - Function_member (int x, string n)
  - Function_member (String n, int x)
- Dua parameter tersebut juga di anggap berbeda daftar parameternya.
- Daftar parameter tidak terkait dengan penamaan variabel yang ada dalam parameter. Misalnya saja 2 daftar parameter berikut :
  - function_member(int x)
  - function_member(int y)
- Dua daftar parameter diatas dianggap sama karena yang berbeda hanya penamaan variable parameternya saja.

Overloading juga bisa terjadi antara parent class dengan subclass-nya jika memenuhi ketiga syarat overload. Ada beberapa aturan overloading yaitu:

- Primitive widening conversion didahulukan dalam overloading dibandingkan boxing dan var args.
- Kita tidak dapat melakukan proses widening dari tipe wrapper ke tipe wrapper lainnya (mengubah Integer ke Long).
- Kita tidak dapat melakukan proses widening dilanjutkan boxing (dari int menjadi Long)
- Kita dapat melakukan boxing dilanjutkan dengan widening (int dapat menjadi Object melalui Integer)
- Kita dapat menggabungkan var args dengan salah satu yaitu widening atau boxing

2.2 Overriding

Overriding adalah Subclass yang berusaha memodifkasi tingkah laku yang diwarisi dari superclass. Tujuannya subclass dapat memiliki tingkah laku yang lebih spesifik sehingga dapat dilakukan dengan cara mendeklarasikan kembali method milik parent class di

subclass. Deklarasi method pada subclass harus sama dengan yang terdapat di super class. Kesamaan pada:

- Nama
- Return type (untuk return type: class A atau merupakan subclass dari class A)
- Daftar parameter (jumlah, tipe dan urutan)

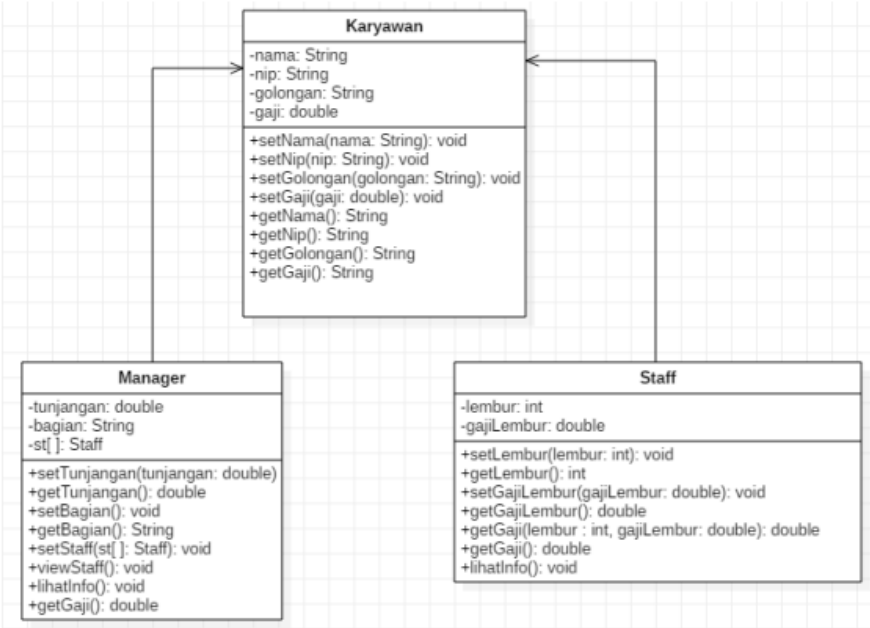
Sehingga method pada parent class disebut overridden method dan method pada subclass disebut overriding method. Ada beberapa aturan method didalam overriding:

- Mode akses overriding method harus sama atau lebih luas dari pada overridden method.
- Subclass hanya boleh meng-override method superclass satu kali saja, tidak boleh ada lebih dari satu method pada kelas yang sama persis.
- Overriding method tidak boleh throw checked exceptions yang tidak dideklarasikan oleh overridden method.
-

3. Praktikum

3.1 Class Diagram

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.



3.2 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(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;
        }
    }
```

3.3 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(int 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());
    }
}
```

3.4 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("================================");
            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("Gaji :"+this.getBagian());
        this.viewStaff();
    }

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

```
}
```

3.5 Main

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Program Testing Class Manager & Class");
        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("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);

        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();
    }
}
```

4. 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);
    }
}
```

4.1 Dari source coding diatas terletak dimanakah overloading?

Jawab: Pada kode program diatas overloading terletak pada method dengan nama perkalian dimana pada program tersebut terdapat dua method dengan nama yang sama tetapi parameter dan isi method yang berbeda serta terletak pada satu class yang sama. Hal tersebut berarti method tersebut merupakan overloading

4.2 Jika terdapat overloading ada berapa jumlah parameter yang berbeda?

Jawab:

Pada method overloading pada program tersebut terdapat dua parameter yang berbeda. Dimana masing masing method dengan nama yang sama yaitu perkalian memiliki parameter yang berbeda meskipun tipe data dari kedua parameter tersebut sama sama menggunakan tipe data integer. Perbedaan kedua parameter tersebut adalah jika di dalam method perkalian yang pertama hanya ada dua variabel yang digunakan untuk parameter sedangkan untuk method perkalian yang kedua menggunakan tiga buah variabel yang digunakan pada parameter tersebut.

```
public class Perkaliankuu {
    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) {
        Perkaliankuu objek= new Perkaliankuu();
        objek.Perkalian(25, 43);
        objek.Perkalian(34.56,23.7);
    }
}
```

4.3 Dari source coding diatas terletak dimanakah overloading?

Jawab: Pada kode program diatas overloading terletak pada method dengan nama perkalian dimana pada program tersebut terdapat dua method dengan nama yang sama tetapi parameter dan isi method yang berbeda serta terletak pada satu class yang sama. Hal tersebut berarti method tersebut merupakan overloading.

4.4 Jika terdapat overloading ada berapa tipe parameter yang berbeda?

Jawab: Pada method overloading pada program tersebut terdapat dua parameter yang berbeda. Dimana masing masing method dengan nama yang sama yaitu perkalian memiliki yang sama yaitu adalah jumlah parameter yakni sama-sama terdapat dua

parameter dan yang membedakan yakni kedua method tersebut memiliki 2 tipe data yang berbeda yakni yang satu bertipe data integer dan satunya lagi bertipe data double

```
public class Ikan {
    public void swim(){
        System.out.println("Ikan bisa berenang");
    }
}
public class Piranha extends Ikan{
    public void swim(){
        System.out.println("Piranha bisa makan daging");
    }
}
public class MainIkan {
    public static void main(String[] args) {
        Ikan a = new Ikan();
        Ikan b = new Piranha();
        a.swim();
        b.swim();
    }
}
```

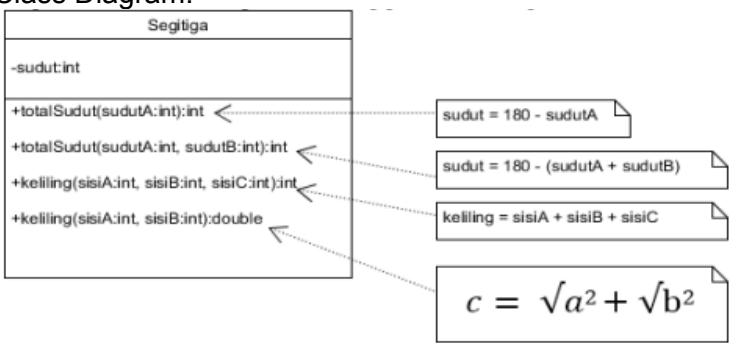
4.5 Dari source coding diatas terletak dimanakah overriding?
Jawab: Terletak pada method swim() baik pada class ikan dan class piranha bisa dikatakn yakni overidding

4.6 Jabarkanlah apabila sourcoding diatas jika terdapat overriding?
Jawab: Berdasarkan kode program diatas class Ikan adalah superclass dari class Piranha. Di dalam class Ikan terdapat method swim() dengan parameter default dan di class Piranha juga terdapat method yang sama seperti yang ada di class Ikan yaitu method swim() dengan parameter default. Dengan adanya hal tersebut maka method swim() pada class Piranha merupakan overriding karena terdapat method dengan nama dan parameter yang sama dengan method yang berada pada superclass

5. Tugas

5.1 Overloading

Class Diagram:



Source Code:

```
public class Segitiga {
    private int sudut;
    public int totalSudut(int sudutA) {
        return sudut = 180-sudutA;
    }
    public int totalSudut(int sudutA, int sudutB) {
        return sudut = 180 - (sudutA + sudutB);
    }
    public int keliling(int sisiA, int sisiB, int sisiC) {
        return sisiA + sisiB + sisiC;
    }
```

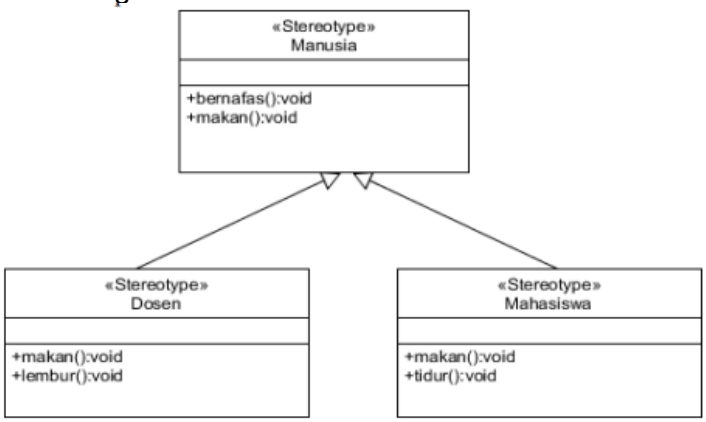
```
    public double keliling(int sisiA, int sisiB) {
        return Math.sqrt(sisiA*sisiA) + Math.sqrt(sisiB*sisiB);
    }
}
public class Main {
    public static void main(String[] args) {
        Segitiga sgt = new Segitiga();
        System.out.println("Total Sudut Segitiga 1 : " + sgt.totalSudut(90));
        System.out.println("Total Sudut Segitiga 2 : " + sgt.totalSudut(30, 30));
        System.out.println("Keliling Segitiga 1 : " + sgt.keliling(12, 12, 12));
        System.out.println("Keliling Segitiga 2 : " + sgt.keliling(8, 8));
    }
}
```

Output:

```
Total Sudut Segitiga 1 : 90
Total Sudut Segitiga 2 : 120
Keliling Segitiga 1 : 36
Keliling Segitiga 2 : 16.0
```

5.2 Overriding

Class Diagram:



Source Code:

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

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

}
public class Mahasiswa extends Manusia{
    public void makan(){
        System.out.println("Mahasiswa Sedang Makan");
    }

    public void Tidur(){
        System.out.println("Mahasiswa Sedang Tidur");
    }
}
public class Dosen extends Manusia{
    public void makan(){
        System.out.println("Dosen Sedang Makan");
    }

    public void lembur(){
```

```
        System.out.println("Dosen Sedang Lembur");
    }
}
public class Main {
    public static void main(String[] args) {
        // membuat objek dari Class Manusia
        Manusia man = new Manusia();
        // Membuat Objek dari class Dosen
        Dosen lecturer = new Dosen();
        // Membuat Objek dari class Mahasiswa
        Mahasiswa students = new Mahasiswa();

        // Menampilkan Seluruh Method
        System.out.println("==============================");
        //menampilkan method dari class Manusia
        man.bernafas();
        man.makan();
        System.out.println("==============================");
        //menampilkan method dari class Dosen
        lecturer.lembur();
        lecturer.makan();
        System.out.println("==============================");
        //menampilkan method dari class Mahasiswa
        students.Tidur();
        students.makan();
        System.out.println("==============================");
        System.out.println("==============================");
        //dynamic method dispatch
        //membuat referensi
        System.out.println("==============================");
        System.out.println(" dynamic method dispatch");
        // dapatkan referensi tipe Manusia
        Manusia ref;
        //referensi mengacu pada objek Manusia
        ref = man;
        // memanggil Method Makan Dari ref
        ref.makan();
        // mengubah referensi tipe Dosen
        ref = lecturer;
        // memanggil Method Makan Dari ref
        ref.makan();
        // mengubah referensi tipe Mahasiswa
        ref = students;
        // memanggil Method Makan Dari ref
        ref.makan();
        System.out.println("==============================");
        System.out.println("==============================");
    }
}
```

Output:

```
============================
Manusia Sedang Bernafas
Manusia Sedang Makan
============================
Dosen Sedang Lembur
Dosen Sedang Makan
============================
Mahasiswa Sedang Tidur
Mahasiswa Sedang Makan
============================

============================
    Dynamic Method Dispatch
============================
Manusia Sedang Makan
Dosen Sedang Makan
Mahasiswa Sedang Makan
============================
```