

MODUL 1

INHERITANCE, POLYMORFISME, DAN VIRTUAL METHOD INOVATION

TUJUAN :

1. Memahami konsep *inheritance* dan dapat menerapkan *inheritance* pada pemrograman berorientasi objek,
2. Memahami konsep polimorfisme dan dapat menerapkan polimorfisme pada pemrograman berorientasi objek,
3. Memahami konsep *virtual method invocation* dan dapat menerapkan *virtual method invocation* pada pemrograman berorientasi objek.

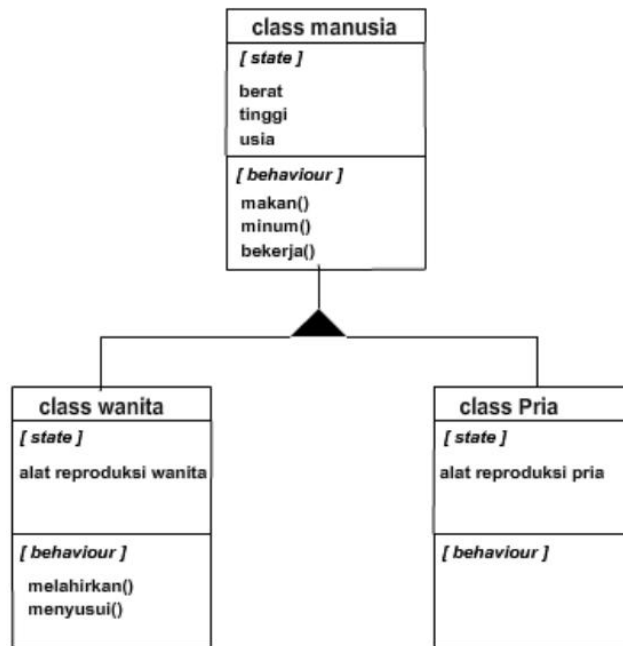
DASAR TEORI :

Pada modul ini kita akan membahas beberapa bentuk / konsep pemrograman yang mendasari pemrograman berorientasi objek, diantaranya yaitu :

1. *Inheritance*
2. Polimorfisme (*Overloading & Overriding Method*)
3. *Virtual Method Invocation* (VMI)

INHERITANCE

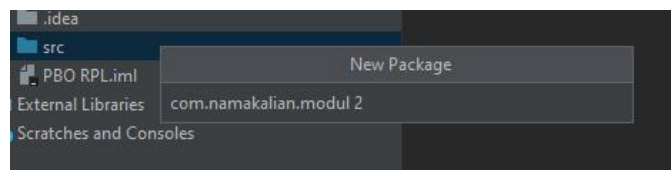
Inheritance adalah salah satu ciri utama dari bahasa program yang berorientasi pada objek, merupakan suatu cara untuk menurunkan atau mewarisi suatu kelas yang lebih umum menjadi suatu kelas yang lebih spesifik. Kelas yang mewariskan biasa disebut dengan ***super class*** / ***parent class***, sedangkan kelas yang diwariskan disebut dengan ***sub class*** / ***child class***. Jadi, pada konsep inheritance, setiap *sub class* dapat mengakses data-data dari kelas utamanya yang bertindak sebagai *super class*.



Ilustrasi *Inheritance*

CODELAB INHERITANCE

Buatlah project baru dan buatlah sebuah package baru sesuai format dibawah ini :



Lalu buatlah sebuah kelas baru dengan nama “A” dan tulis kode program seperti di bawah ini :

```

public class A{
    public int x;
    public int y;

    public void printXy() {
        System.out.println("Nilai x : " + x +
            "\n" +
            "Nilai y : " + y);
    }
}
  
```

Setelah itu, buat kelas baru lagi dengan nama “B” dan tulis kode program seperti di bawah ini :

```
public class B extends A{
    public int z;

    public void sumValue() {
        System.out.println("Jumlah : " +
                           (x + y + z));
    }
}
```

Setelah itu, buat kelas baru dengan nama “DemoInheritance”, kemudian tuliskan kode nya seperti dibawah ini :

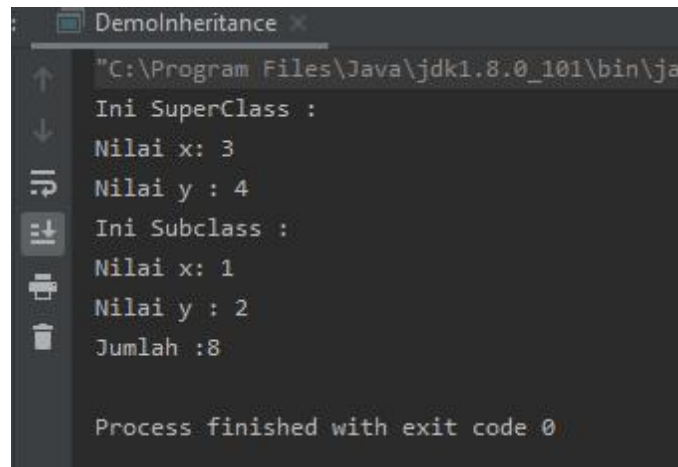
```
/**
this code was written by Ajeng Fitria
*/
public class DemoInheritance{
    public static void main(String[] args) {
        A superclass = new A();
        B subclass = new B();

        System.out.println("Ini SuperClass :");
        superclass.x = 3;
        superclass.y = 4;
        superclass.printXy();

        //member super class dapat diakses oleh
        //sub class nya
        System.out.println("Ini SubClass :");
        subclass.x = 1;
        subclass.y = 2;
        subclass.printXy();

        //member tambahan pada sub class
        subclass.z = 5;
        subclass.sumValue();
    }
}
```

Setelah melengkapinya lalu jalankan programnya, maka output dari program tersebut akan seperti ini.



```
Demolnheritance x
"C:\Program Files\Java\jdk1.8.0_101\bin\ja
Ini SuperClass :
Nilai x: 3
Nilai y : 4
Ini Subclass :
Nilai x: 1
Nilai y : 2
Jumlah :8
Process finished with exit code 0
```

POLIMORFISME

Polymorphism berasal dari bahasa Yunani yang berarti banyak bentuk. Dalam PBO, konsep ini memungkinkan digunakannya suatu interface yang sama untuk memerintah obyek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda.

OVERLOADING METHOD

Di dalam java, kita diperbolehkan membuat dua atau lebih method dengan nama yang sama dalam satu kelas, tetapi jumlah dan tipe data argumen masing – masing method haruslah berbeda satu dengan yang lainnya. Hal itu yang dinamakan dengan Overloading Method.

CODELAB OVERLOADING METHOD

Buatlah sebuah kelas baru dengan nama “Lagu” dan tulis kode program seperti dibawah ini :

```
/**
this code was written by Ajeng Fitria
*/
public class Lagu{
    public String judul;
    public String pencipta;

    public Lagu(String judul) {
        this.judul = judul;
        this.pencipta = "Tidak di ketahui";
    }

    public Lagu(String judul, String pencipta){
        this.judul = judul;
        this.pencipta = pencipta;
    }

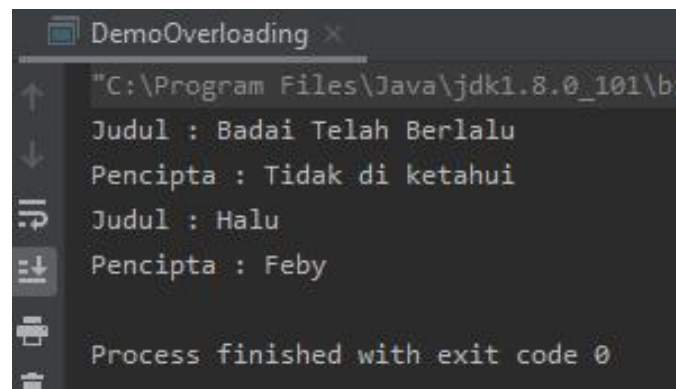
    public void getDataLagu() {
        System.out.println("Judul : " +
                           judul + "\n" +
                           "Pencipta : " +
                           pencipta);
    }
}
```

Lalu buatlah sebuah class baru dengan nama “DemoOverloading”, dan tulis kode program seperti dibawah ini.

```
/**
this code was written by Ajeng Fitria
*/
public class DemoOverLoading{
    public static void main(String[] args){
        Lagu lagu1 = new Lagu("Badai Telah Berlalu");
        Lagu lagu2 = new Lagu("Halu","Feby");

        lagu1.getDataLagu();
        lagu2.getDataLagu();
    }
}
```

Setelah melengkapinya lalu jalankan programnya, maka output dari program tersebut akan seperti ini :



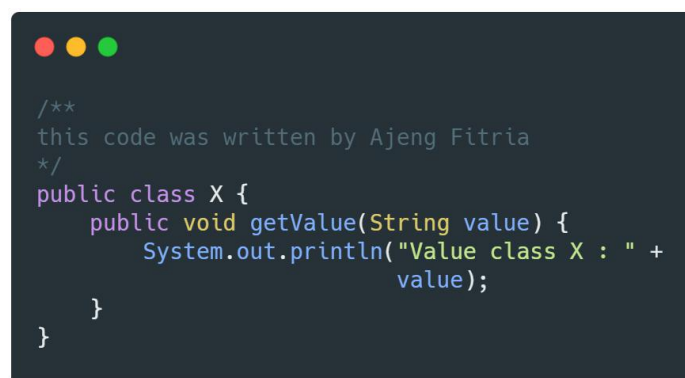
```
DemoOverloading x
"C:\Program Files\Java\jdk1.8.0_101\bin
Judul : Badai Telah Berlalu
Pencipta : Tidak di ketahui
Judul : Halu
Pencipta : Feby
Process finished with exit code 0
```

OVERRIDING METHOD

Di dalam java, jika dalam suatu subclass mendefinisikan sebuah method yang sama dengan yang dimiliki oleh superclass, maka method yang dibuat dalam subclass tersebut dikatakan meng-override superclass-nya. Sehingga jika kita mencoba memanggil method tersebut dari instance subclass yang kita buat, maka method milik subclass-lah yang akan dipanggil, bukan lagi method milik superclass-nya.

CODELAB OVERRIDING METHOD

Buatlah sebuah class baru dengan nama “X”, dan tulis kode program seperti dibawah ini.



```
/**
 * this code was written by Ajeng Fitria
 */
public class X {
    public void getValue(String value) {
        System.out.println("Value class X : " +
                           value);
    }
}
```

Lalu buatlah sebuah kelas baru dengan nama “Y”, dan tulis kode program seperti di bawah ini.

```

/**
this code was written by Ajeng Fitria
*/
public class Y extends X {
    @Override
    public void getValue(String value) {
        super.getValue(value);
        System.out.println("Value class Y : " +
                           value);
    }
}

```

Kemudian, buat kelas baru dengan nama “DemoOverriding” dan tuliskan kodenya seperti di bawah ini :

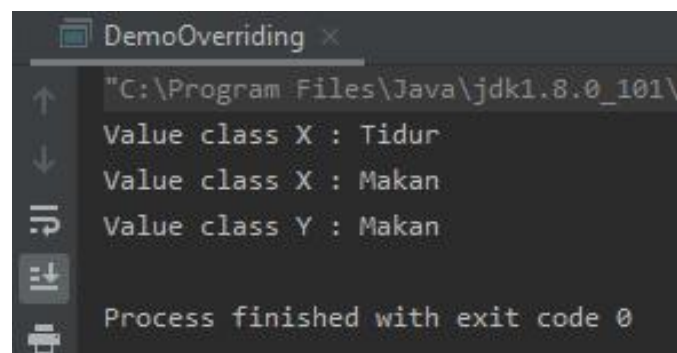
```

/**
this code was written by Ajeng Fitria
*/
public class DemoOverriding{
    public static void main(String[] args) {
        X superClass = new X();
        Y subClass = new Y();

        superClass.getValue("Tidur");
        subClass.getValue("Makan");
    }
}

```

Setelah melengkapinya lalu jalankan programnya, maka output dari program tersebut akan seperti ini.



```

DemoOverriding x
"C:\Program Files\Java\jdk1.8.0_101\
Value class X : Tidur
Value class X : Makan
Value class Y : Makan
Process finished with exit code 0

```

Terlihat pada contoh di atas, kelas Y yang merupakan turunan dari kelas X, mengoverride method `getValue()`, sehingga pada waktu kita memanggil method tersebut dari variable instance kelas Y (variable `subClass`), yang terpanggil adalah method yang sama yang ada pada kelas Y.

VIRTUAL METHOD INVOCATION (VMI)

Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan overriding. Pada saat objek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass, dimana yang seharusnya dipanggil adalah overridden method.

CODELAB VIRTUAL METHOD INVOCATION (VMI)

Buatlah sebuah class baru dengan nama “Parent”, dan tulis kode program seperti dibawah ini.

```
/**
this code was written by Ajeng Fitria
*/
public class Parent{
    public int x = 5;

    public void getData() {
        System.out.println("Data parent : " + x);
    }
}
```

Lalu buatlah sebuah kelas baru dengan nama “Child”, dan tulis kode program seperti di bawah ini.

```
/**
this code was written by Ajeng Fitria
*/
public class Child extends Parent{
    public int z = 7;

    public void getData(){
        System.out.println("Data child : " + z);
    }
}
```

Kemudian buat sebuah kelas baru dengtan nama “DemoVMI” dan tuliskan kodenya seperti di bawah ini :


```

/**
this code was written by Ajeng Fitria
*/
public class DemoVMI{
    public static void main(String[] args) {
        Child child = new Child();
        Parent parentChild = new Child();

        child.getData();
        parentChild.getData();
    }
}

```

Setelah melengkapinya lalu jalankan programnya, maka output dari program tersebut akan seperti ini.

```

DemoVMI x
"C:\Program Files\Java\jdk1.8.0_101\bin\java.exe" -Djava.class.path=
Data child 7
Data child 7
Process finished with exit code 0

```

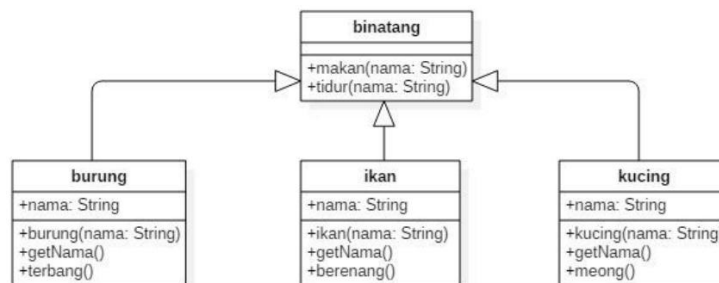
LATIHAN

club
+nama : String +tahunBerdiri : int +stadion :String +juaraUcl : int +deskripsi : String
+club() +club(nama : String) +club(nama : String, deskripsi : String) +club(nama : String, tahunBerdiri : int, stadion :String) +club(nama : String, tahunBerdiri : int, stadion :String, juaraUcl : int, deskripsi : String) +getTeam()

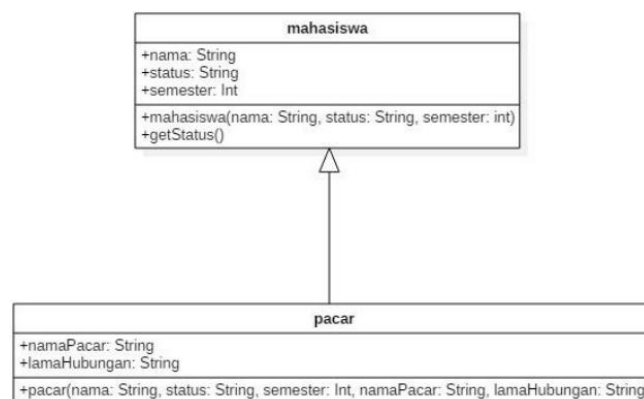
1. Pada class club di atas, kita akan melakukan percobaan overloading dimana kita akan mencoba semua constructure yang ada. Disini kita membutuhkan 5 objek club berbeda, setiap objek club diberikan perilaku overloading yang berbeda.

- Club pertama berikan perilaku overloading dengan constructure kosong
- Club kedua berikan perilaku overloading dengan constructure 1 parameter
- Club ketiga berikan perilaku overloading dengan constructure 2 parameter
- Club keempat berikan perilaku overloading dengan constructure 3 parameter
- Club kelima berikan perilaku overloading dengan constructure 5 parameter

Setelah itu panggil method getTeam yang akan mencetak semua data dalam objek kelas club. Disini akan terjadi null variable jika objek club tidak memiliki beberapa data dari hasil overloading, oleh karena itu berikanlah default value jika hal itu terjadi



2. Pada diagram class diatas, terdapat class binatang sebagai parent yang mewarisi sifat pada kelas pewaris yaitu burung, ikan, dan kucing. Masing-masing kelas pewaris memiliki sifat dari kelas parent nya, tetapi kelas pewaris juga memiliki sifat uniknya tersendiri. Buatlah ketiga kelas pewaris dan panggil semua method yang ada di setiap kelas yang ada.



3. Pada class diagram diatas, terdapat kelas mahasiswa dan pacar dimana kelas pacar mewarisi kelas mahasiswa. Buatlah objek kelas mahasiswa dan isi datanya lalu panggil method getStatus, setelah itu buatlah kelas objek mahasiwa dengan perilaku VMI ke kelas pacar dan isi data setelah itu panggil getStatus.