

BAB IV KONSEP PEMROGRAMAN BERORIENTASI OBJEK

Untuk dapat menguasai pemrograman Java, harus mengerti dengan baik konsep pemrograman berorientasi objek, karena Java merupakan bahasa pemrograman berorientasi objek. Pada bagian ini akan dibahas konsep-konsep penting dalam pemrograman berorientasi objek, sehingga diharapkan kita akan lebih mudah dalam mempelajari bahasa Java.

A. Objek

Pada dasarnya semua benda yang ada di dunia nyata dapat dianggap sebagai sebuah objek. Jika perhatikan lebih lanjut, pada dasarnya ada dua karakteristik yang utama pada sebuah objek, yaitu :

- Setiap objek memiliki atribut sebagai status yang kemudian akan disebut sebagai *state*.
- Setiap objek memiliki tingkah laku yang kemudian akan disebut sebagai *behaviour*.

Contoh sederhananya adalah : objek sepeda

- Sepeda memiliki atribut (*state*) : pedal, roda, jeruji, dan warna.
- Sepeda memiliki tingkah laku (*behaviour*) : kecepatannya menaik, kecepatannya menurun, dan perpindahan gigi sepeda.

Dalam pengembangan perangkat lunak berorientasi objek, objek dalam perangkat lunak akan menyimpan *state*-nya dalam variabel dan menyimpan informasi tingkah laku (*behaviour*) dalam *method-method* atau fungsi-fungsi/prosedur.

B. Class

Class berbeda dengan objek. Class merupakan prototipe yang mendefinisikan variabel-variabel dan method-method secara umum. Sedangkan objek pada sisi yang lain merupakan instansiasi dari suatu kelas.

Latihan 10. Class.java

```
class Buku {  
    String pengarang;  
    String judul;  
    void Isi(String isi1,String isi2) {  
        judul    = isi1;  
        pengarang = isi2;  
    }  
}
```

```

void CetakKeLayar() {
    if(judul==null && pengarang==null) return;
    System.out.println("Judul : " + judul +
        ", pengarang : " + pengarang);
}

class Karangan {
    public static void main(String[] args) {
        Buku a,b,c,d;
        a = b = c = d = new Buku();
        a.Isi("Pemrograman Java","Asep Herman Suyanto");
        a.CetakKeLayar();
        b.Isi(null,null);
        b.CetakKeLayar();
        c.Isi(null,"Johan Prasetyo Hendriyanto");
        c.CetakKeLayar();
        d.Isi("Pemrograman Web",null);
        d.CetakKeLayar();
    }
}

```

```

C:\> Command Prompt

G:\JAVAA>javac Class.java

G:\JAVAA>java Karangan
Judul : Pemrograman Java, pengarang : Asep Herman Suyanto
Judul : null, pengarang : Johan Prasetyo Hendriyanto
Judul : Pemrograman Web, pengarang : null

```

C. Enkapsulasi

Dalam sebuah objek yang mengandung variabel-variabel dan method-method, dapat ditentukan hak akses pada sebuah variabel atau method dari objek. Pembungkusan variabel dan method dalam sebuah objek dalam bagian yang terlindungi inilah yang disebut dengan *enkapsulasi*. Jadi, *enkapsulasi* dapat diartikan sebagai bungkusan (wrapper) pelindung program dan data yang sedang diolah. Pembungkus ini mendefinisikan perilaku dan melindungi program dan data yang sedang diolah agar tidak diakses sembarangan oleh program lain.

Manfaat dari proses enkapsulasi adalah :

- **Modularitas**
Kode sumber dari sebuah objek dapat dikelola secara independen dari kode sumber objek yang lain.
- **Information Hiding**
Karena kita dapat menentukan hak akses sebuah variabel/method dari objek, dengan demikian kita bisa menyembunyikan informasi yang tidak perlu diketahui objek lain.

D. Inheritance

Class dapat didefinisikan dengan referensi pada class yang lain yang telah terdefinisi. *Inheritance* merupakan pewarisan atribut dan method pada sebuah class yang diperoleh dari class yang telah terdefinisi tersebut. Setiap *subclass* akan mewarisi *state* (variabel-variabel) dan *behaviour* (method-method) dari *superclass*-nya. *Subclass* kemudian dapat menambahkan *state* dan *behaviour* baru yang spesifik dan dapat pula memodifikasi (*override*) *state* dan *behaviour* yang diturunkan oleh *superclass*-nya.

Keuntungan dari inheritance adalah :

- *Subclass* menyediakan *state/behaviour* yang spesifik yang membedakannya dengan *superclass*, hal ini akan memungkinkan programmer Java untuk menggunakan ulang *source code* dari *superclass* yang telah ada.
- Programmer Java dapat mendefinisikan *superclass* khusus yang bersifat generik, yang disebut *abstract class*, untuk mendefinisikan *class* dengan *behaviour* dan *state* secara umum.

Istilah dalam *inheritance* yang perlu diperhatikan :

- **Extends**
Keyword ini harus kita tambahkan pada definisi class yang menjadi subclass.
- **Superclass**
Superclass digunakan untuk menunjukkan hirarki class yang berarti class dasar dari subclass/class anak.
- **Subclass**
Subclass adalah class anak atau turunan secara hirarki dari superclass.

- ***Super***
Keyword ini digunakan untuk memanggil konstruktor dari superclass atau menjadi variabel yang mengacu pada superclass.
- **Method *Overriding***
Pendefinisian ulang method yang sama pada subclass.

Dalam ***inheritance***, method ***overriding*** berbeda dengan method ***overloading***. Kalau method ***overriding*** adalah mendefinisikan kembali method yang sama, baik nama method maupun signature atau parameter yang diperlukan dalam subclass, kalau method ***overloading*** adalah mendefinisikan method yang memiliki nama yang sama, tetapi dengan signature yang berbeda dalam definisi class yang sama.

Latihan 11. Inheritance.java

```
class A {
    int x;
    int y;
    void TampilkanNilaixy() {
        System.out.println("Nilai x : " + x + ", y : " + y);
    }
}

class B extends A {
    int z;
    void TampilkanJumlah() {
        //subclass dapat mengakses member dari superclass
        System.out.println("Jumlah : " + (x+y+z));
    }
}

class Inheritance {
    public static void main(String[] args) {
        A VarsuperOb = new A();
        B VarsubOb = new B();
        System.out.println("SuperClass");
        VarsuperOb.x = 10;
        VarsuperOb.y = 20;
        VarsuperOb.TampilkanNilaixy();
    }
}
```

```

System.out.println("SubClass");
//member superclass dapat diakses dari subclass nya
VarsubOb.x = 5;
VarsubOb.y = 4;
VarsubOb.TampilkanNilaixy();

System.out.println("SubClass Jumlah");
//member tambahan yang hanya ada dalam subclass
VarsubOb.z = 30;
VarsubOb.TampilkanJumlah();

System.out.println("SubClass");
VarsubOb.x = 15;
VarsubOb.y = 14;
VarsubOb.TampilkanNilaixy();

System.out.println("SuperClass");
VarsuperOb.x = 10;
VarsuperOb.y = 20;
//super.x = 100;    error
//super.y = 200;    error
VarsuperOb.TampilkanNilaixy();

System.out.println("SubClass Jumlah");
VarsubOb.z = 60;
VarsubOb.TampilkanJumlah();
}
}

```

```

C:\> Command Prompt
G:\JAVAA>javac Inheritance.java
G:\JAVAA>java Inheritance
SuperClass
Nilai x : 10, y : 20
SubClass
Nilai x : 5, y : 4
SubClass Jumlah
Jumlah : 39
SubClass
Nilai x : 15, y : 14
SuperClass
Nilai x : 10, y : 20
SubClass Jumlah
Jumlah : 89

```

E. Polimorfisme

Kata *polimorfisme* yang berarti satu objek dengan banyak bentuk yang berbeda, adalah konsep sederhana dalam bahasa pemrograman berorientasi objek yang berarti kemampuan dari suatu variabel referensi objek untuk memiliki aksi berbeda bila method yang sama dipanggil, dimana aksi method tergantung dari tipe objeknya. Kondisi yang harus dipenuhi supaya *polimorfisme* dapat diimplementasikan adalah :

- Method yang dipanggil harus melalui variabel dari basis class atau superclass.
- Method yang dipanggil harus juga menjadi method dari basis class.
- Signature method harus sama baik pada superclass maupun subclass.
- Method access attribute pada subclass tidak boleh lebih terbatas dari basis class.

Latihan 12. Polimorfisme.java

```
abstract class Bentuk {
    protected int panjang;
    protected int lebar;
    public String getBentuk() {
        return "Bentuk Dasar";
    }
    public abstract int hitungLuas();
}

class BujurSangkar extends Bentuk {
    public BujurSangkar(int panjang1, int lebar1) {
        this.panjang = panjang1;
        this.lebar = lebar1;
    }
    public String getBentuk() {
        return "Bentuk Bujur Sangkar";
    }
    public int hitungLuas() {
        return panjang*lebar;
    }
}
```

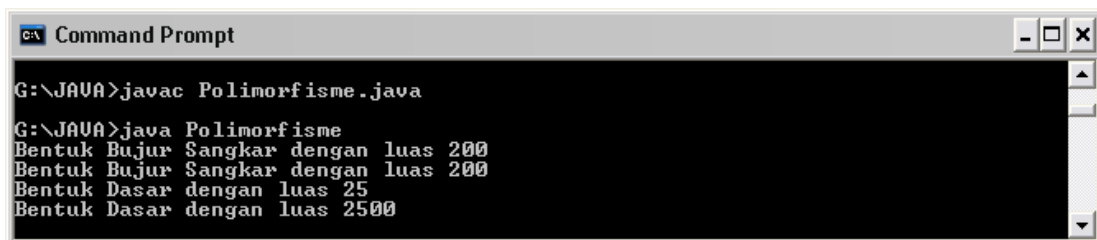
```

class SegiTiga extends Bentuk {
    public SegiTiga(int panjang2, int lebar2) {
        this.panjang = panjang2;
        this.lebar = lebar2;
    }
    //public String getBentuk() {
        //return "Bentuk Segi Tiga";
        //return "";
    //}
    public int hitungLuas() {
        return this.panjang*this.lebar/2;
    }
}

class Polimorfisme {
    public static void cetakLuasBentuk(Bentuk btk) {
        System.out.println(btk.getBentuk() + " dengan luas " +
btk.hitungLuas());
    }
    public static void main(String[] args) {
        BujurSangkar bs = new BujurSangkar(10,20);
        BujurSangkar bs1 = new BujurSangkar(10,20);
        SegiTiga st = new SegiTiga(5,10);
        SegiTiga st1 = new SegiTiga(50,100);

        cetakLuasBentuk(bs);
        cetakLuasBentuk(bs1);
        cetakLuasBentuk(st);
        cetakLuasBentuk(st1);
    }
}

```



```

G:\JAVAA>javac Polimorfisme.java
G:\JAVAA>java Polimorfisme
Bentuk Bujur Sangkar dengan luas 200
Bentuk Bujur Sangkar dengan luas 200
Bentuk Dasar dengan luas 25
Bentuk Dasar dengan luas 2500

```

F. Interface

Pada Java juga dikenal konsep *interface*, yang merupakan device yang digunakan untuk komunikasi antar objek berbeda yang tidak memiliki hubungan apapun. *Interface* bisa dikatakan sebagai protokol komunikasi antar objek tersebut.

Latihan 13. Interface.java

```
interface Control {  
    public void pindahChannel(int channel);  
    public void PerbesarVolume(int intensitas);  
    public void PerkecilVolume(int intensitas);  
}  
  
class TelevisiA implements Control {  
    String[] channelTv = {"RCTI", "SCTV", "INDOSIAR", "TRANS TV", "TPI"};  
    public void pindahChannel(int channel) {  
        System.out.println("Pindah channel pada tv A ke : " + channelTv[channel]);  
    }  
    public void PerbesarVolume(int intensitas) {  
        System.out.println("Perbesar intensitas volume pada tv A sebanyak : " +  
intensitas);  
    }  
    public void PerkecilVolume(int intensitas) {  
        System.out.println("Perkecil intensitas volume pada tv A sebanyak : " +  
intensitas);  
    }  
}  
  
class TelevisiB implements Control {  
    String[] chanTv = {"TVRI", "LA TV", "TV 7", "RCTI", "SCTV"};  
    public void pindahChannel(int channel) {  
        System.out.println("Perintah pindah channel pada tv B ke : " +  
chanTv[channel]);  
    }  
    public void PerbesarVolume(int intensitas) {  
        System.out.println("Perbesar intensitas volume pada tv B sebanyak : " +  
intensitas);  
    }  
}
```



```

    public void PerkecilVolume(int intensitas) {
        System.out.println("Perkecil intensitas volume pada tv B sebanyak : " +
intensitas);
    }
}

```

```

class RemoteControl {
    public static final int PINDAH_CHANNEL = 1;
    public static final int PERBESAR_VOLUME = 2;
    public static final int PERKECIL_VOLUME = 3;
    public void kirimPerintahKeTv(int aksi,Control tv,int tombol) {
        switch(aksi) {
            case PINDAH_CHANNEL:
                tv.pindahChannel(tombol);
                break;
            case PERBESAR_VOLUME:
                tv.PerbesarVolume(tombol);
                break;
            case PERKECIL_VOLUME:
                tv.PerkecilVolume(tombol);
        }
    }
}

```

```

class Interface {
    public static void main(String[] args) {
        TelevisiA tvA = new TelevisiA();
        TelevisiB tvB = new TelevisiB();
        RemoteControl rc = new RemoteControl();
        //Kirim perintah ke tvA
        rc.kirimPerintahKeTv(RemoteControl.PINDAH_CHANNEL,tvA,2);
        rc.kirimPerintahKeTv(RemoteControl.PERBESAR_VOLUME,tvA,5);
        //Kirim perintah ke tvB
        rc.kirimPerintahKeTv(RemoteControl.PINDAH_CHANNEL,tvB,1);
        rc.kirimPerintahKeTv(RemoteControl.PERKECIL_VOLUME,tvB,3);
    }
}

```



```
G:\JAWA>javac Interface.java
G:\JAWA>java Interface
Pindah channel pada tv A ke : INDOSIAR
Perbesar intensitas volume pada tv A sebanyak : 5
Perintah pindah channel pada tv B ke : LA TU
Perkecil intensitas volume pada tv B sebanyak : 3
```

LATIHAN

1. Berikan contoh program yang dapat menggambarkan/menjelaskan tentang state dan behaviour ?
2. Berikan contoh program yang dapat mewakili penjelasan tentang Class dan jelaskan program tersebut ?
3. Berikan contoh program yang dapat mewakili penjelasan tentang Enkapsulasi dan jelaskan program tersebut ?
4. Keuntungan apa yang diperoleh dengan melakukan pewarisan dan kapan digunakan ?
5. Berikan contoh program yang dapat mewakili penjelasan tentang Inheritance dan jelaskan program tersebut ?
6. Berikan contoh program yang dapat mewakili penjelasan tentang Polimorfisme dan jelaskan program tersebut ?
7. Berikan contoh program yang dapat mewakili penjelasan tentang Interface dan jelaskan program tersebut ?