BAB 4
ENCAPSULATION

Tujuan

1. Praktikan mampu memahami konsep encapsulation (enkapsulasi) yang ada di java
2. Mampu memahami dan mengimplementasikan encapsulation

Ringkasan Materi

A. Encapsulation

Enkapsulasi adalah suatu cara untuk menyembunyikan informasi detail dari suatu class. Dalam enkapsulasi terdapat hak akses *public, protected, dan private*. Hak akses *public* memungkinkan semua kelas dapat mengakses meskipun berada pada paket yang berbeda, hak akses *protected* hanya diberikan kepada kelasnya sendiri dan turunannya, serta kelas-kelas dalam satu paket. Sedangkan *private* hanya boleh diakses oleh kelasnya sendiri.

Access Modifier	Class tersebut	Package	Subclass	Root / Network
Private	v			
Default	v	v		
Protected	v	v	v	
Public	v	v	v	v

Enkapsulasi bertujuan untuk menjaga suatu proses program agar tidak dapat diakses secara sembarangan atau di intervensi oleh program lain. Konsep enkapsulasi sangat penting dilakukan untuk menjaga kebutuhan program agar dapat diakses sewaktu-waktu, sekaligus menjaga program tersebut. Dua hal yang mendasar dalam enkapsulasi yakni :

A.1 Information Hiding

Sebelumnya, kita dapat mengakses anggota class baik berupa atribut maupun method secara langsung dengan menggunakan objek yang telah kita buat. Hal ini dikarenakan akses kontrol yang diberikan kepada atribut maupun method yang ada di dalam class tersebut adalah 'public'. Kita dapat menyembunyikan informasi dari suatu class sehingga anggota class tersebut tidak dapat diakses dari luar, caranya adalah hanya dengan memberikan akses kontrol 'private' ketika mendeklarasikan atribut atau method. Proses ini disebut dengan information hiding.

A.2 Interface to Access Data

Jika kita telah melakukan information hiding terhadap suatu atribut pada suatu class, lalu bagaimana melakukan perubahan terhadap atribut yang kita sembunyikan tersebut. Caranya adalah dengan membuat suatu interface berupa method untuk menginisialisasi atau merubah nilai dari suatu atribut tersebut. Manfaat utama teknik encapsulation adalah kita mampu memodifikasi kode tanpa merusak kode yang telah digunakan pada class lain. Enkapsulasi memiliki manfaat sebagai berikut:

- Modularitas

Source code dari sebuah class dapat dikelola secara independen dari source code class yang lain. Perubahan internal pada sebuah class tidak akan berpengaruh bagi class yang menggunakannya.

- Information Hiding
Penyembunyian informasi yang tidak perlu diketahui objek lain.

B. **Accessor**

Untuk mengimplementasikan enkapsulasi, kita tidak menginginkan sembarang object dapat mengakses data kapan saja. Untuk itu, kita deklarasikan atribut dari class sebagai private. Namun, ada kalanya dimana kita menginginkan object lain untuk dapat mengakses data private. Dalam hal ini kita gunakan accessor methods.

Accessor Methods digunakan untuk membaca nilai variabel pada class, baik berupa instance maupun static. Sebuah accessor method umumnya dimulai dengan penulisan *get<namaInstanceVariable>*. Method ini juga mempunyai sebuah return value. Sebagai contoh, kita ingin menggunakan accessor method untuk dapat membaca nama, alamat, nilai bahasa Inggris, Matematika, dan ilmu pasti dari siswa. Mari kita perhatikan salah satu contoh implementasi accessor method.

```
public class StudentRecord {
  private String name;
  :
  :
  public String getName(){
     return name;
  }
}
```

C. Mutator

Method yang dapat memberi atau mengubah nilai variable dalam class, baik itu berupa instance maupun static. Method semacam ini disebut dengan mutator methods. Sebuah mutator method umumnya tertulis *set<namaInstanceVariabel>*. Mari kita perhatikan salah satu dari implementasi mutator method.

```
public class StudentRecord{
  private String name;
  :
  :
  public void setName( String temp ){
      name = temp;
  }
}
```

Pelaksanaan Percobaan

A. Encapsulation 1

Ketikkan program di bawah ini

```
1   public class Student {
2       private String name;
3       private int mark;
4       public void setName(String n){
5          name=n;
6       }
7       public String getName(){
8          return name;
9       }
10      public void setMark(int m){
11         mark=m;
12      }
13      public int getMark(){
14         return mark;
        }
    }
```

```
1   public class Test {
2       public static void main(String [] args) {
3           Student s1=new Student();
4           s1.setName("Enkapsulasi");
5           s1.setMark("90");
6           System.out.println("s1Name is "+s1.setName());
7           System.out.println("s1Mark is "+s1.setMark());
8           System.out.println("name dan mark "+name+" "+mark);
9       }
10  }
```

B. Encapsulation 2

Buatlah class Vehicle1

```
1   public class Vehicle1
2   {
3   private double load, maxLoad;
4
5   public Vehicle1 (double max){
6         this.maxLoad = max;
7
8   }
9   public double getLoad(){
10        return this.load;
11  }
12  public double getMaxLoad(){
13        return this.maxLoad;
14  }
15  public boolean addBox(double weight){
16        double temp = 0.0D;
17        temp = this.load + weight;
18        if(temp <= maxLoad){
19        this.load = this.load + weight;
20        return true;
21        }
```

```
22        else
23        {
24        return false;
25        }
26  }
27  }
```

```
1   public class TestVehicle1{
2         public static void main(String[] args){
3               System.out.println("Creating a vehicle with a 10,000
4   kg maximumload.");
5               Vehicle1 vehicle = new Vehicle1(10000);
6               System.out.println("Add box #1 (500kg) : " +
7   vehicle.addBox(500));
8               System.out.println("Add box #2 (250kg) : " +
9   vehicle.addBox(250));
10              System.out.println("Add box #3 (5000kg) : " +
11  vehicle.addBox(5000));
12              System.out.println("Add box #4 (4000kg) : " +
13  vehicle.addBox(4000));
14              System.out.println("Add box #5 (300kg) : " +
15  vehicle.addBox(300));
16              System.out.println("Vehicle load is "
17  +vehicle.getLoad() + "kg");
18        }
19  }
```

Data dan Analisis hasil percobaan

A. Encapsulation 1

Pertanyaan

1. Lakukan percobaan diatas dan benahi jika menemukan kesalahan!
   ……
   ……

2. Jika pada baris 6 *s1.setName* diubah menjadi *s1.getName* apa yang terjadi? jelaskan!
   ……
   ……

3. Setelah diperbaiki, ubahlah hak akses pada baris 4 (pada class Student) menjadi *private* apa yang terjadi jika class Test dijalankan? Jelaskan!
   ……
   ……

4. Jika kedua kelas diatas terdapat dalam package yang sama apakah konsep enkapsulasi tetap berfungsi? jelaskan!
   ……
   ……

B. Encapsulation 2

Pertanyaan

1. Method apakah yang menjadi accessor (getter) ?
   ……

……..

2. Tambahkan source code berikut dibawah baris ke 6 pada class TestVehicle1.
   System.out.println("Add load(100kg) : " + (vehicle.load=500));
   Jalankan program, apakah output dari program tersebut?
   Kembalikan program seperti semula.

   ……..
   ……..

3. Ubahlah tipe data pada atribut load dan maxload pada class Vehicle1 menjadi **public**.
   Jalankan program, apakah output dari program tersebut?
   a. Tambahkan source kode berikut dibawah baris ke 6 pada class TestVehicle1.
      System.out.println("Add load(100kg) : " + (vehicle.load=500));
      Jalankan program, apakah output dari program tersebut?
      Kembalikan program seperti semula.
   b. Tambahkan source kode berikut dibawah baris ke 12 pada class TestVehicle1.
      System.out.println("Add load(100kg) : " + (vehicle.load=500));
      Jalankan program, apakah output dari program tersebut?
      Kembalikan program seperti semula.

   ……..
   ……..

4. Ulangi instruksi pada nomer 4 dengan mengubah tipe data pada atribut load dan maxload pada class Vehicle1 menjadi **protected.**

   ……..
   ……..

5. Ulangi instruksi pada nomer 4 dengan mengubah tipe data pada atribut load dan maxload pada class Vehicle1 menjadi **default.**

   ……..
   ……..

Tugas Praktikum

Anda saat ini bekerja sebagai IT engineer di bank XY, dan anda diminta untuk membuat sistem transaksi yang ada di mesin ATM yang baru dibeli. Aplikasi yang anda buat harus memenuhi requirement yang telah dirancang oleh analis di bank XY. Kira-kira begini requirementnya:

- Informasi rekening (saldo, nomor rekening, nama pemegang rekening) tidak bisa diubah ubah oleh nasabah secara langsung.
- Nomer rekening terdiri dari 15 digit, yang mana 3 digit awal menentukan jenis rekening yang dimiliki oleh nasabah:
  - 001 : rekening gold, memiliki limit penarikan tunai 10 juta, tanpa biaya transfer
  - 192 : rekening silver, memiliki limit penarikan tunai 7 juta, namun ketika transfer akan dikenakan biaya admin 6500
  - 283 : rekening bronze, memiliki limit penarikan tunai 5 juta, namun ketika transfer akan dikenakan biaya admin 7500
- Nasabah harus memiliki saldo minimal 10000, jika saldo pasca transaksi kurang dari batas minimal tadi, maka transaksi dianggap gagal
- Buatlah sistem ATM tadi terbatas pada penarikan tunai dan transfer saja, dan sebelum memulai transaksi harus melakukan otentikasi dahulu dengan memasukkan PIN dan nomor rekening.