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

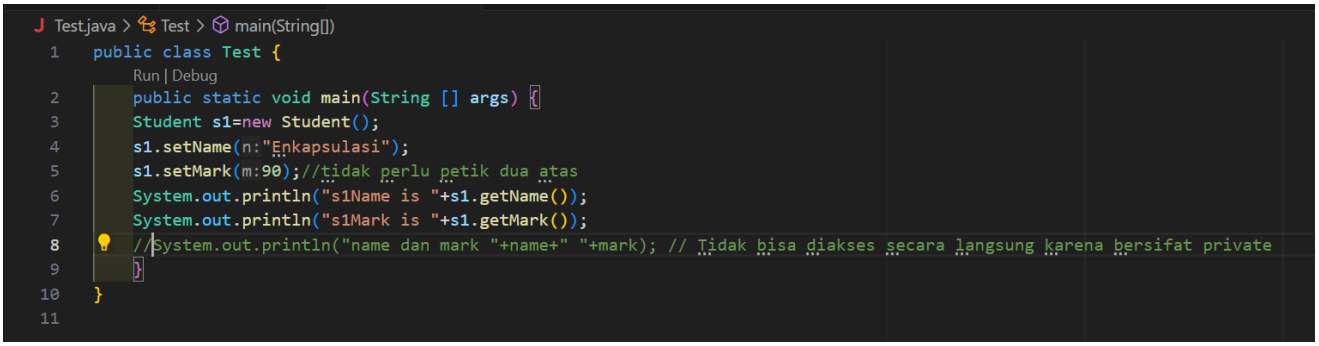
A. Encapsulation 1

Pertanyaan

1. Lakukan percobaan diatas dan benahi jika menemukan kesalahan!

   Disini saya merubah kodingan di bagian Test :

   1. Pada kelas Test, setName() dan setMark() untuk mencetak nilai nama dan nilai tanda, seharusnya menggunakan getName() dan getMark() untuk mendapatkan nilai tersebut.
   2. Untuk memberikan nilai string ke parameter mark yang bertipe int pada metode setMark(). Ini akan menyebabkan kesalahan karena tipe data tidak cocok / tidak sesuai.
   3. Pada baris-8 untuk mencetak variabel name dan mark di luar kelas Student tanpa mengaksesnya melalui metode get.



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

   Jika pada baris 6 kode s1.setName diubah menjadi s1.getName(), maka akan terjadi kesalahan dalam pemanggilan metode.

Metode getName() adalah metode akses untuk mengambil nilai dari variabel name dalam objek s1, sedangkan metode setName() adalah metode mutator yang digunakan untuk mengatur nilai dari variabel name. Ketika mengubah s1.setName, menjadi s1.getName(), sebenarnya tidak melakukan apa pun untuk mengatur nilai dari variabel name, tetapi mencoba untuk mengambil nilai dari variabel name yang telah diatur sebelumnya.
Karena getName() mengembalikan nilai dari variabel name, tetapi tidak melakukan apa pun untuk mengubahnya, hasilnya tidak akan berguna dan biasanya akan diabaikan atau menyebabkan kesalahan dalam kompilasi jika nilainya tidak digunakan dalam operasi lain.

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

   Jika hak akses pada baris keempat (pada kelas Student) diubah menjadi private, maka atribut name dan mark akan berubah menjadi private juga. Ini berarti bahwa hanya metode dalam kelas Student itu sendiri yang bisa mengakses atribut name dan mark. Ketika class Test dijalankan, tidak akan bisa mengakses langsung atribut name dan mark dari objek Student dalam class Test.

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

   Ya, konsep enkapsulasi tetap berlaku meskipun keduanya berada dalam package yang sama. Meskipun kelas Test berada dalam package yang sama dengan kelas Student, kelas Test masih harus menggunakan metode setter dan getter yang disediakan untuk mengakses atribut name dan mark.

B. Encapsulation 2

Pertanyaan

1. Method apakah yang menjadi accessor (getter) ?
   - getLoad(): Metode ini mengembalikan nilai dari atribut load yang merupakan beban saat ini dari kendaraan.

   - getMaxLoad(): Metode ini mengembalikan nilai dari atribut maxLoad yang merupakan beban maksimum yang dapat ditampung oleh kendaraan.

……..

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.

```
Creating a vehicle with a 10,000 kg maximumload.
Add box #1 (500kg) : true
Add load(100kg) : 500.0
Add box #2 (250kg) : true
Add box #3 (5000kg) : true
Add box #4 (4000kg) : true
Add box #5 (300kg) : false
Vehicle load is 9750.0kg
```

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.

```
public class Vehicle1 {
    public double load, maxLoad;

    public Vehicle1(double max) {
        this.maxLoad = max;
    }

    public double getLoad() {
        return this.load;
    }

    public double getMaxLoad() {
        return this.maxLoad;
    }

    public boolean addBox(double weight) {
        double temp = 0.0D;
        temp = this.load + weight;
        if (temp <= maxLoad) {
            this.load = this.load + weight;
            return true;
        } else {
            return false;
        }
    }
}
```

A.

```
Creating a vehicle with a 10,000 kg maximumload.
Add box #1 (500kg) : true
Add load(100kg) : 500.0
Add box #2 (250kg) : true
Add box #3 (5000kg) : true
Add box #4 (4000kg) : true
Add box #5 (300kg) : false
Vehicle load is 9750.0kg
```

B.

```
Creating a vehicle with a 10,000 kg maximumload.
Add box #1 (500kg) : true
Add load(100kg) : 500.0
Add box #2 (250kg) : true
Add box #3 (5000kg) : true
Add box #4 (4000kg) : true
Add box #5 (300kg) : false
Vehicle load is 9750.0kg
Add load(100kg) : 500.0
```

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

```
28
29  public class Vehicle1 {
30      public protected load, maxLoad;
31
```

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

```
public class Vehicle1 {
    public default load, maxLoad;
```

Tugas Praktikum

Anda dan tim anda mendapat sebuah proyek untuk merancang sistem transaksi pada sebuah swalayan Tiny. Anda ditugasi oleh tim untuk membuat programnya berdasarkan hasil analisis tim anda :

1. Informasi akun seorang pelanggan (saldo, nomor pelanggan, nama) tidak bias diubah oleh pelanggan secara langsung.
2. Nomor pelanggan terdiri dari 10 digit, dimana 2 digit awal adalah jenis rekening
    - 38 : Pelanggan jenis silver; setiap pembelian diatas 1 jt maka mendapat cashback sebesar 5%
    - 56 : Pelanggan jenis gold; setiap pembelian diatas 1 jt maka mendapat cashback sebesar 7%, selain itu cashback 2% (cashback kembali ke saldo)
    - 74 : Pelanggan jenis platinum; setiap pembelian diatas 1 jt maka mendapat cashback sebesar 10%, selain itu cashback 5% (cashback kembali ke saldo)
3. Pelanggan harus memiliki saldo minimal Rp10.000, jika saldo pasca transaksi kurang dari batas minimal tadi, maka transaksi pembelian dianggap gagal
4. Buatlah sistem transaksi swalayan ini terbatas pada pembelian dan top up saja dan menggunakan PIN dan nomor pelanggan sebagai syarat transaksi pembelian atau top up.
5. Apabila pelanggan melakukan 3x kesalahan dalam autentifikasi, maka akun pelanggan akan defreeze / diblokir sehingga tidak bisa digunakan lagi.