

## 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 diintervensi 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;
15    }
16 }

```

```

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    }
23 }

```

```

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!

```

... Nisa-Aulia-Harismadani-PBO-PTI-A
J Testjava M x Student.java M x
Tugas5 > src > J Student.java > Student
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;
15    }
16 }

... Nisa-Aulia-Harismadani-PBO-PTI-A
J Testjava M x J Student.java M
Tugas5 > src > J Testjava > Test
1 public class Test {
2     public static void main(String [] args) {
3         Student s1 = new Student();
4         s1.setName(n:"Enkapsulasi");
5         s1.setMark(m:90);
6         System.out.println("s1Name is " + s1.getName());
7         System.out.println("s1Mark is " + s1.getMark());
8         System.out.println("name dan mark " + s1.getName() + " " + s1.getMark());
9     }
10 }

```

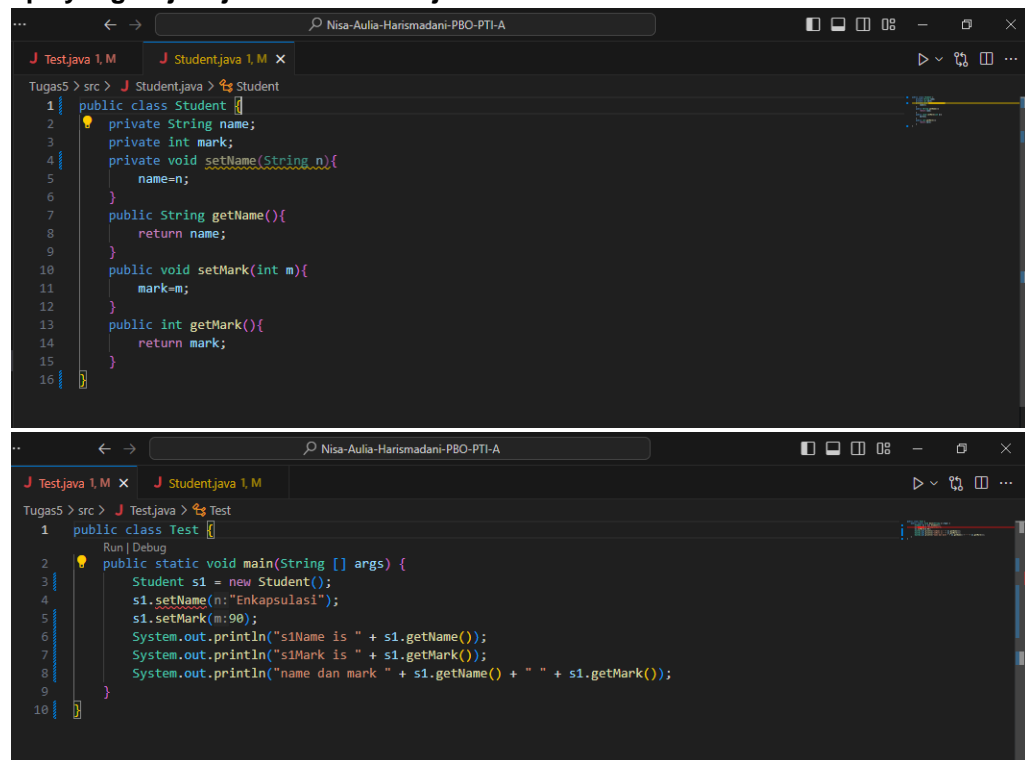
**PENJELASAN :** Saya telah melakukan percobaan diatas, dalam class Student tidak terdapat kesalahan sama sekali, namun pada class Test terdapat kesalahan yaitu pada bagian `System.out.println("s1Name is "+s1.setName());` `System.out.println("s1Mark is "+s1.setMark());`;

karena seharusnya menggunakan metode `getName` dan `getMark`, bukan `setName` dan `setMark`, hal tersebut terjadi karena metode `setName` dan `setMark` hanya digunakan untuk mengatur nilai nama dan nilai mark, sedangkan metode `getName` dan `getMark` digunakan untuk mengambil nilai yang sudah diatur. Selanjutnya pada bagian `System.out.println("name dan mark "+name+" "+mark);` juga mengalami error, hal tersebut diakibatkan karena variable `name` dan `mark` tidak dideklarasikan di metode `main`, maka agar tidak terjadi error seharusnya mengakses nilai `name` dan `mark` dari objek `s1` menggunakan metode `getName` dan `getMark`

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

Jika `s1.setName` diubah menjadi `s1.getName` maka akan menimbulkan kesalahan kompilasi. Hal tersebut dikarenakan `s1.getName` merupakan sebuah metode yang mengembalikan nilai, untuk mengatur nilai atribut `name` dari objek `s1` digunakan metode `setName("Enkapsulasi")` namun, jika diubah menjadi `s1.getName` itu akan memanggil metode `getName()` tanpa melakukan apa apa dengan nilai yang dikembalikannya dan merupakan operasi yang tidak relevan dan menyebabkan kesalahan dalam sintaks.

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



```

1 public class Student {
2     private String name;
3     private int mark;
4     private 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;
15    }
16 }

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

**PENJELASAN :** Jika hak akses atribut `name` pada class `Student` diubah menjadi `private`, maka akan terjadi error pada baris `s1.setName("Enkapsulasi")`; dalam class `Test`, karena atribut `name` hanya dapat diakses dari dalam class `student` itu sendiri dan tidak dapat diakses dari luar class tersebut, termasuk class `Test`

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

```

1 public class Student {
2
3
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;
15    }
16 }
17
18 class Test {
19     public static void main(String [] args) {
20         Student s1 = new Student();
21         s1.setName(n:"Enkapsulasi");
22         s1.setMark(m:90);
23         System.out.println("s1Name is " + s1.getName());
24         System.out.println("s1Mark is " + s1.getMark());
25         System.out.println("name dan mark " + s1.getName() + " " + s1.getMark());
26     }
27 }

```

Run | Debug

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

s1Name is Enkapsulasi
s1Mark is 90
name dan mark Enkapsulasi 90
PS C:\Users\HP\Documents\Tugas_1_PBO\Nisa-Aulia-Harismadani-PBO-PTI-A>

```

**PENJELASAN :** Iya, konsep enkapsulasi tetap berfungsi meskipun kedua kelas berada dalam package yang sama, meskipun kedua kelas berada dalam package yang sama, hak akses yang diberikan kepada atribut dan metode dalam class Student masih berlaku. Atribut name dan mark dalam class Student diakses dengan hak akses private yang berarti mereka hanya dapat diakses dan diubah nilainya dari dalam class Student itu sendiri

## B. Encapsulation 2

Pertanyaan

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

Accessor(getter) itu sendiri merupakan metode yang digunakan untuk mengambil nilai dari suatu atribut dalam sebuah objek. Dalam kode class Vehicle1, method yang menjadi accessor (getter) adalah public double getLoad(), metode ini mengambil nilai atribut load dari objek dan mengembalikannya dan public double getMaxLoad(), method ini mengambil nilai atribut maxLoad dari objek dan mengembalikannya

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.

```

1 public class TestVehicle1{
2     public static void main(String[] args){
3         System.out.println("Creating a vehicle with a 10,000 kg maximumload.");
4         Vehicle1 vehicle = new Vehicle1(10000);
5         System.out.println("Add box #1 (500kg) : " + vehicle.addBox(500));
6         System.out.println("Add box #2 (250kg) : " + vehicle.addBox(250));
7         System.out.println("Add load(100kg) : " + (vehicle.load=500));
8         System.out.println("Add box #3 (5000kg) : " + vehicle.addBox(5000));
9         System.out.println("Add box #4 (4000kg) : " + vehicle.addBox(4000));
10        System.out.println("Add box #5 (300kg) : " + vehicle.addBox(300));
11        System.out.println("Vehicle load is " + vehicle.getLoad() + "kg");
12    }
13 }

```

**PENJELASAN** : jika ditambahkan source code `System.out.println("Add load(100kg) : " + (vehicle.load=500));` dibawah baris 6 pada class TestVehicle1 maka outputnya akan menunjukkan bahwa box box berhasil ditambahkan kedalam kendaraan dengan bobot masing-masing, nilai total beban kendaraan adalah 5550.0kg kemudian kode yang baru ditambahkan memodifikasi langsung nilai atribut load dari objek vehicle menjadi 500

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.

**PENJELASAN** : Apabila mengubah tipe data pada atribut load dan maxload pada class Vehicle1 menjadi public output program tidak akan berubah. Namun, akan menghasilkan risiko yang lebih besar dalam pengelolaan data dalam kelas Vehicle1, meskipun demikian output program tidak dipengaruhi oleh aksesibilitas atribut load dan makLoad

- a.

```

1 public class TestVehicle1 {
2     public static void main(String[] args) {
3         System.out.println("Creating a vehicle with a 10,000 kg maximumload.");
4         Vehicle1 vehicle = new Vehicle1(10000);
5         System.out.println("Add box #1 (500kg) : " + vehicle.addBox(500));
6         System.out.println("Add load(100kg) : " + (vehicle.load=500));
7         System.out.println("Add box #2 (250kg) : " + vehicle.addBox(250));
8         System.out.println("Add box #3 (5000kg) : " + vehicle.addBox(5000));
9         System.out.println("Add box #4 (4000kg) : " + vehicle.addBox(4000));
10        System.out.println("Add box #5 (300kg) : " + vehicle.addBox(300));
11        System.out.println("Vehicle load is " + vehicle.getLoad() + "kg");
12    }
13 }

```

**PENJELASAN :** Output akan menunjukkan bahwa box pertama berhasil ditambahkan kedalam kendaraan dengan bobot 500kg yang menyebabkan total beban kendaraan menjadi 500kg, kemudian kode yang baru ditambahkan akan memodifikasi langsung nilai atribut load dari objek vehicle menjadi 500 yang kemudian dicetak dalam output. Selanjutnya box-box lainnya ditambahkan dan total beban kendaraan dihitung. Namun, box ketiga dengan bobot 5000kg tidak berhasil ditambahkan karena melebihi kapasitas maksimum kendaraan yaitu 10000kg sehingga metode addBox mengembalikan nilai false, kemudian total beban kendaraan dicetak

```

1 public class TestVehicle1 {
2     public static void main(String[] args) {
3         System.out.println("Creating a vehicle with a 10,000 kg maximumload.");
4         Vehicle1 vehicle = new Vehicle1(10000);
5         System.out.println("Add box #1 (500kg) : " + vehicle.addBox(500));
6         System.out.println("Add box #2 (250kg) : " + vehicle.addBox(250));
7         System.out.println("Add box #3 (5000kg) : " + vehicle.addBox(5000));
8         System.out.println("Add box #4 (4000kg) : " + vehicle.addBox(4000));
9         System.out.println("Add load(100kg) : " + (vehicle.load=500));
10        System.out.println("Add box #5 (300kg) : " + vehicle.addBox(300));
11        System.out.println("Vehicle load is " + vehicle.getLoad() + "kg");
12    }
13 }

```

b.

**PENJELASAN :** dengan penambahan source code tersebut maka program akan mencetak output tambahan, pernyataan `System.out.println("Add load(100kg) : " + (vehicle.load=500));` akan menetapkan nilai atribut load dari objek vehicle menjadi 500 dan kemudian mencetak pesan yang mencantumkan nilai tersebut

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

```

1 public class Vehicle1 {
2     protected double load, maxLoad;
3     public Vehicle1 (double max) {
4         this.maxLoad = max;
5     }
6 }

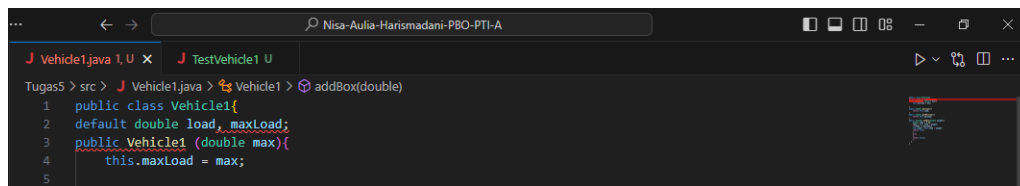
```

**PENJELASAN :** Jika mengubah tipe data pada atribut load dan makload pada class Vehicle1 menjadi protected, maka akses ke atribut tersebut akan dibatasi pada kelas itu sendiri dan kelas turunannya yang berarti atribut tersebut tidak dapat diakses langsung dari luar paket tempat kelas Vehicle1 berada, kecuali jika ad akelas turunan dari Vehicle1

5. Ulangi instruksi pada nomer 4 dengan mengubah tipe data pada atribut load dan



maxload pada class Vehicle1 menjadi default.



```
1 public class Vehicle1{
2     default double load,maxLoad;
3     public Vehicle1 (double max){
4         this.maxLoad = max;
5     }
```

**PENJELASAN :** Jika mengubah tipe data pada atribut load dan maxload pada class Vehicle1 maka akan terjadi error, hal tersebut terjadi karena hak akses default membatasi aksesibilitas atribut atau method hanya untuk kelas-kelas dalam paket yang sama, jika mengakses atribut yang memiliki hak akses default dari luar paket tersebut maka akan terjadi kesalahan

**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.