ENKAPSULASI, ACCESSOR METHOD, MUTATOR METHOD

PENDAHULUAN

Sebenarnya sebelumnya kita sudah melihat beberapa implementasi dari enkapsulasi. Sekarang kita akan meninjau konsep dari enkapsulasi tersebut

Maksud dari enkapsulasi adalah pembungkusan.

Sama seperti kapsul obat. Obat **dibungkus** di dalam kapsul supaya tidak terurai dan **rapi.** Obat kapsul juga **menyembunyikan** isinya yang membuat kita tidak tahu apa isidi dalamnya.

Begitu juga dengan enkapsulasi pada pemrograman berorientasi objek.

ENKAPSULASI PADA PEMROGRAMAN

Konsep enkapsulasi pada pemrograman adalah pembungkusan data teknis (**atribut** dan ***method***) dari pengguna

Data teknis ini dibungkus dalam sebuah unit yang telah dijelaskan sebelumnya yaitu **kelas** atau ***class***.

Jadi bisa dianalogikan bahwa data teknis adalah obat, dan kelas adalah kapsulnya sehingga menjadi rapi.

ENKAPSULASI PADA PEMROGRAMAN

Dengan enkapsulasi dalam sebuah kelas inilah yang membuat pemrograman berorientasi objek lebih fleksibel untuk memodelkan kasus nyata.

Penggunaan kembali kelas-kelas menjadi lebih mudah dibanding dengan pendefinisian fungsi-fungsi di pemrograman berparadigma prosedural.

INFORMATION HIDING

- you can use the encapsulation concept to implement an information-hiding mechanism.
- You implement this information-hiding mechanism by making your class attributes inaccessible from the outside using **access modifier** and by providing **getter and/or setter methods** for attributes that shall be readable or updatable by other classes.

ACCESS MODIFIER

- Dalam melakukan pembungkusan kode dan data dalam java terdapat tingkatan akses data yang perlu diketahui :
  - Tingkat akses private
  - Tingkat akses protected
  - Tingkat akses public
  - Tingkat akses default

ENKAPSULASI - PRIVATE

- Akses Private berarti bahwa method yang digunakan hanya dapat diakses oleh kelas yang memilikinya.
- Dengan mendeklarasikan data dan method menggunakan akses *private*, ini berarti data dan method tersebut tidak boleh diakses atau dignakan oleh kelas-kelas lain yang terdapat di dalam program
- Sebuah variabel atau *method yang dideklarasikan private hanya dapat* diakses oleh *method yang merupakan member dari kelas tersebut. Ia* tidak dapat diakses oleh kelas lain yang berada di dalam *package yang* sama ataupun di lain *package.*
- Untuk mendeklarasikan suatu data atau method dengan tingkat akses private, digunakan kata kunci *private*

ENKAPSULASI - PRIVATE

```
public class StudentRecord
{
        //akses dasar terhadap variabel
        private int name;

        //akses dasar terhadap metode
        private String getName(){
                return name;
        }
}
```

ENKAPSULASI - PROTECTED

- Suatu data maupun method yang dideklarasikan dengan tingkat akses protected dapat diakses oleh kelas yang memilikinya dan juga oleh kelas-kelas yang masih memiliki hubungan turunan
- Access control protected berarti member dapat diakses oleh kelas yang berada dalam package yang sama dan subclass yang berada di dalam package yang berbeda.
- Untuk mendeklarasikan tipe data atau method protected digunakan kata kunci protected

ENKAPSULASI - PROTECTED

```
public class StudentRecord
{
        //akses pada variabel
        protected int name;

        //akses pada metode
        protected String getName(){
                return name;
        }
}
```

ENKAPSULASI - PUBLIC

- Tingkat akses publik merupakan kebalikan dari tingkat akses private.
- Data dan method yang bersifat public dapat diakses oleh semua bagian dalam program.
- Dengan kata lain, data-data maupun method-method yang dideklarasikan dengan tingkat akses publik akan dikenali dan diakses oleh semua kelas yang ada di dalam program, baik yang merupakan kelas turunan maupun kelas yang tidak memiliki hubungan sama sekali.

ENKAPSULASI - PUBLIC

```
public class StudentRecord
{
	//akses dasar terhadap variabel
	public int name;

	//akses dasar terhadap metode
	public String getName(){
		return name;
	}
}
```

ENKAPSULASI - DEFAULT

- Tipe ini mensyaratkan bahwa hanya *class dalam package yang sama yang memiliki hak* akses terhadap variabel dan *methods dalam class. Tidak terdapat keyword pada tipe ini*

ENKAPSULASI - DEFAULT

```
public class StudentRecord
{
        //akses dasar terhadap variabel
        int name;

        //akses dasar terhadap metode
        String getName(){
                return name;
        }
}
```

TIPE AKSES

Modifier	Kelas yang sama	*Package* yang Sama	Subclass	Semua Kelas
`private`	Ya			
`default`	Ya	Ya		
`protected`	Ya	Ya	Ya	
`public`	Ya	Ya	Ya	Ya

CONTOH

```
1
2    public class Student {
3      String name;
4      String address;
5
6    }
7
```

Bagaimana cara mengakses atribut ini?

CONTOH

```
1
2   public class StudentAksi {
3   public static void main( String[] args ){
4   //membuat 3 object StudentRecord
5   Student baca = new Student();
6
7   //Memberi nama siswa
8   baca.name="anna";
9
10  //Menampilkan nama siswa "Anna"
11  System.out.println( baca.name );
12
13  }
14
15  }
```

ENKAPSULASI

- Jika name tidak dienkapsulasi:
  - User dapat memasukkan sembarang nilai, sehingga perlu melakukan penyembunyian informasi (information hiding) thd atribut name, sehingga name tidak bisa diakses secara langsung.

- Bagaimana menyembunyikan informasi dari suatu class sehingga atribut-atributnya tersebut tidak dapat diakses dari luar?
  - Dengan memberikan akses control private ketika mendeklarasikan suatu atribut

ENKAPSULASI

```
1
2    public class Student {
3     private String name;
4     private String address;
5
6    }
7
```

ENKAPSULASI

```
1
2   public class StudentAksi {
3   public static void main( String[] args ){
4   //membuat 3 object StudentRecord
5   Student baca = new Student();
6
7   //Memberi nama siswa
    baca.name="anna";
9
10  //Menampilkan nama siswa "Anna"
    System.out.println( baca.name );
12
13  }
14
15  }
```

Hasil Running

```
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - name has private access in Student
        at StudentAksi.main(StudentAksi.java:8)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

ENKAPSULASI

- Lalu, kalau atribut name tersebut disembunyikan, bagaimana cara mengakses atribut name itu untuk memberikan atau mengubah nilai?
  - Diperlukan abstraksi yang didalamnya terdapat implementasi untuk mengakses data name.

ABSTRAKSI

Analogi mengenai abstraksi,
Jika dianalogikan abstraksi dengan mobil, terdapat setir, pedal gas, rem, dll. Kita dapat menggunakannya, tapi kita tidak tahu tepatnya bagaimana komponen tersebut bekerja. Itulah ***abstraksi*** pada dunia nyata.

Begitu juga program, terdapat sebuah kelas yang terdiri dari berbagai fungsi dan prosedur. Kita bisa memakainya, tetapi kita belum tentu bisa mengetahui cara kerja dari fungsi atau prosedur tersebut dengan berinteraksi dengan antarmuka saja (***interface***)

CONTOH ABSTRAKSI

Contoh abstraksi paling sederhana adalah Accessor Methods dan ***Mutator Methods***. Tujuannya masing-masing adalah untuk mengambil dan menetapkan nilai dari data pada sebuah objek.

ACCESSOR METHOD

- 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 METHOD

- ***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.***

ACCESSOR METHOD

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

MUTATOR METHODS

- Bagaimana jika kita menghendaki *object lain untuk mengubah data?*
- *Yang dapat kita* lakukan adalah membuat *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>.***

MUTATOR METHODS

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

ENCAPSULASI

```
1   public class StudentRecord {
2
3   private String name;
4   private String address;
5
6   /**
7   * Menghasilkan nama dari Siswa
8   */
9   public String getName(){
10  return name;
11  }
12  /**
13  * Mengubah nama siswa
14  */
15  public void setName( String temp ){
16  name = temp;
17  }
18  }
19
```

ENCAPSULASI

```
1   public class StudentRecordExample {
2
3   public static void main( String[] args ){
4   //membuat 3 object StudentRecord
5   StudentRecord annaRecord = new StudentRecord();
6
7   //Memberi nama siswa
8   annaRecord.setName("Anna");
9
10  //Menampilkan nama siswa "Anna"
11  System.out.println( annaRecord.getName() );
12
13  }
14  }
15
```

CONTOH IMPLEMENTASI

```
public class Hewan {              // kelas yang terenkapsulasi baik
  private String nama;            // data atribut di set privat
  private int umur;               // data yang bersifat privat tidak bisa diakses sembarang dari luar

  public int getUmur() {          // pembentukan getter dan setter merupakan bentuk abstraksi
    return umur;                  // terdapat getter untuk tiap data atribut
  }                               // getter ini untuk mengambil data dari kelas yang ada

  public String getNama() {
    return nama;
  }

  public void setUmur(int umurBaru) {   // terdapat juga setter untuk tiap data atribut
    umur = umurBaru;
  }

  public void setNama(String namaBaru) {
    nama = namaBaru;
  }
}
```