

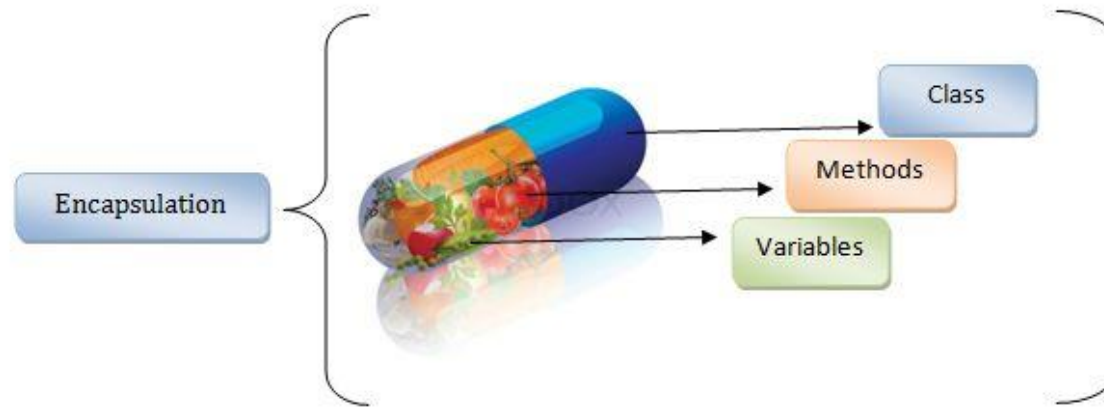
ENKAPSULASI

Minggu ke-3 PBO

Overview

- Enkapsulasi
 - ▣ Definisi
 - ▣ Tujuan
 - ▣ Mekanisme
 - ▣ Access Level Modifier
 - ▣ Setter & Getter
- Notasi pada UML Class Diagram

Enkapsulasi



Enkapsulasi

- Definisi:
 - ▣ Penyatuan/penggabungan atribut dan method dari suatu objek menjadi suatu kesatuan
 - ▣ Pembatasan akses langsung terhadap komponen dari suatu objek

Tujuan Enkapsulasi

- Penyembunyian struktur internal dari suatu objek → information hiding/data hiding
- Melindungi atribut dari perubahan di luar class secara random. Atribut dapat dibuat menjadi read-only atau write-only
- Mempermudah implementasi perubahan requirements
- Mempermudah pengujian unit sistem

Access Level Modifier

- 4 access level modifier untuk **atribut dan method**:
 - public, protected, no modifier, private
- 2 access level modifier untuk **class**:
 - public dan no modifier

Access Level Modifier

- Terdapat 4 access level modifier yaitu:
 - public – dapat diakses dari mana saja
 - protected – dapat diakses di luar package asal merupakan subclass (inherit terhadap class nya)
 - no modifier (*package-private*) – hanya dapat diakses di dalam package yang sama
 - private – hanya dapat diakses di dalam kelas yang sama

Akses Modifier

Modifier	Class	Package	Subclass	Outside Package
public	v	v	v	v
protected	v	v	v	
no modifier	v	v		
private	v			

Mekanisme Enkapsulasi

- Set access level modifier atribut menjadi private sehingga tidak dapat diakses secara langsung di luar class
- Sediakan getter dan setter (sesuai kebutuhan) sebagai cara untuk mengakses atau memodifikasi private attribute

Setter dan Getter

□ Getter

- Public method yang berfungsi mengembalikan nilai dari atribut private
- Ada return value

□ Setter

- Public method yang berfungsi untuk memanipulasi nilai dari atribut private
- Tidak ada return value

Tanpa Enkapsulasi

```
public class Mobil{  
    public String merek;  
    public int kecepatan;  
}
```

```
public class MobilDemo {  
    Run | Debug  
    public static void main(String[] args) {  
        Mobil mobil1 = new Mobil();  
        mobil1.merek = "Suzuki";  
        mobil1.kecepatan = 100;  
  
        System.out.println(mobil1.merek);  
        System.out.println(mobil1.kecepatan);  
    }  
}
```

Dengan Enkapsulasi

```
public class Mobil {  
    private String merek;  
    private int kecepatan;  
  
    public void setMerek(String merek) {  
        this.merek = merek;  
    }  
  
    public String getMerek() {  
        return merek;  
    }  
  
    public void setKecepatan(int kecepatan) {  
        this.kecepatan = kecepatan;  
    }  
  
    public int getKecepatan() {  
        return kecepatan;  
    }  
}
```

```
public class MobilDemo {  
    Run | Debug  
    public static void main(String[] args) {  
        Mobil mobil1 = new Mobil();  
        mobil1.setMerek("Suzuki");  
        mobil1.setKecepatan(100);  
  
        System.out.println(mobil1.getMerek());  
        System.out.println(mobil1.getKecepatan());  
    }  
}
```

this keyword

- Keyword *this* merujuk pada current object
- Keyword ini biasanya digunakan untuk mengeliminasi kebingungan antara atribut dan parameter dengan nama yang sama

Read-Only & Write-Only Attribute

- Read-only attribute → hanya memiliki getter, tetapi tidak memiliki setter
- Write-only attribute → hanya memiliki setter, tetapi tidak memiliki getter

Notasi Pada UML Class Diagram

- Notasi access level modifier pada UML class diagram adalah sebagai berikut:
 - Tanda plus (+) untuk public
 - Tanda pagar (#) untuk protected
 - Tanda minus (-) untuk private
 - Untuk no-modifier tidak diberi notasi

Anggota
- nama: String - alamat: String
+ getNama(): String + setNama(newNama: String): void + getAlamat(): String + setAlamat(newAlamat: String): void