

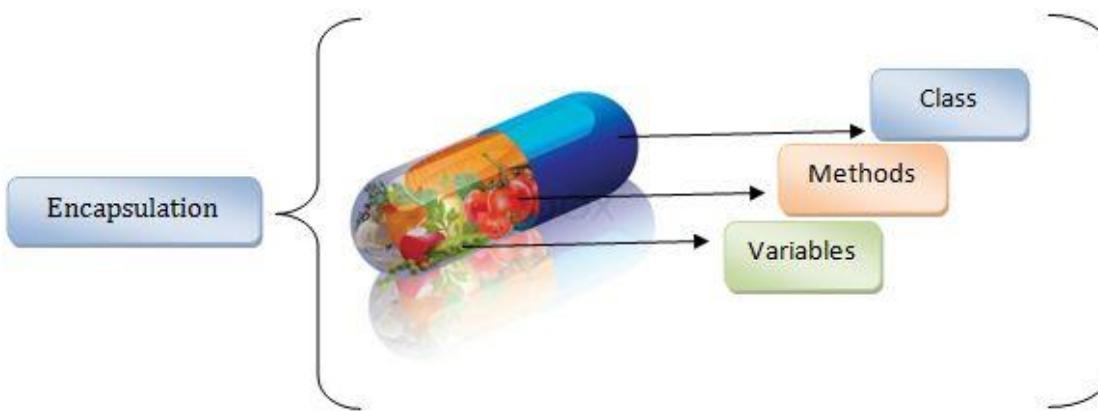
ENKAPSULASI

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 detil implementasi 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** → memungkinkan adanya **validasi**
- Mempermudah implementasi **perubahan requirements**
- Mempermudah **pengujian** unit sistem

Access Level Modifier

- Access Level Modifier adalah fitur fitur dalam OOP yang digunakan untuk mengatur tingkat akses terhadap atribut/method/class
- 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

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

Akses Modifier

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

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**
 - Method yang berfungsi mengembalikan nilai dari atribut private
 - Ada return value
- **Setter**
 - 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 Attribute

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

```
public class Rekening {  
    private double saldo;  
  
    public double getSaldo() {  
        return saldo;  
    }  
  
    public void setor(double jumlah) {  
        if (jumlah > 0) {  
            saldo += jumlah;  
        }  
    }  
  
    public void tarik(double jumlah) {  
        if (jumlah > 0 && jumlah <= saldo) {  
            saldo -= jumlah;  
        }  
    }  
}
```

Write-Only Attribute

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

```
public class User {  
    private String hashedPassword;  
  
    public void setPassword(String password) {  
        this.hashedPassword = PasswordUtil.hashPassword(password);  
    }  
}
```

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

