

MATA KULIAH : PEMROGRAMAN BERORIENTASI OBJEK
SESI PERTEMUAN : V (LIMA)
MATERI : ENCAPSULATION
DOSEN : ALUN SUJJADA, S.KOM., M.T

A. PENGERTIAN ENCAPSULATION

Enkapsulasi (encapsulation) adalah pembungkusan data atau menyembunyian data-data privat dari suatu obyek sehingga tidak dapat diakses dari obyek lain.

Mengapa menggunakan teknik enkapsulasi?

1. Secara teknik dalam enkapsulasi data, *method*, atau *variable* dalam suatu *class* disembunyikan dari *class* lain dan hanya bisa diakses dikelas itu sendiri.
2. Teknik menyembunyian *variable*, data atau *method* yang hanya bisa diakses oleh anggota dalam *class* itu sendiri menggunakan keyword “**private**”, dan ketika kelas itu diakses oleh kelas lain, maka bagian-bagian tersebut tidak bisa diakses secara langsung. Hal ini dilakukan agar data yang dibuat aman dari proses injeksi, dan perubahan nilai yang tidak seharusnya.
3. Pada implementasinya bisa menggunakan metode abstraksi.
4. Implementasi teknik ini lebih mudah menggunakan *Setter* dan *Getter*.

berikut ini adalah beberapa manfaat enkapsulasi:

- **Data Hiding:** Cara membatasi akses ke suatu bagian tertentu (baik itu *methods* atau *variable* didalam suatu *class*) dengan cara menyembunyikan detail implementasi.
- **Increased Flexibility:** Proses untuk membuat *variable* atau *method* hanya bisa membaca saja atau menulis saja tergantung pada kebutuhan yang akan digunakan. Lebih lanjut mengenai hal ini akan dibahas pada *method Setter* dan *Getter*.
- **Control Over Data:** Dikarenakan property tidak bisa diakses secara langsung, maka *programmer* bisa mengontrol data apa saja yang bisa masuk, sehingga lebih mudah untuk dikontrol.
- **Reusability:** Memudahkan untuk mengorganisasikan fungsi yang bisa digunakan kembali

- **Testing code is easy:** Kode yang di enkapsulasi cenderung mudah untuk dilakukan unit testing.

B. AKSES MODIFIER

Modifier adalah sebuah keyword yang memberikan keterangan tambahan dan ditempatkan di awal deklarasi kelas, konstruktor, atribut, maupun *method*. *Modifier* dibagi menjadi dua:

- *Modifier* akses (*access Modifier*)
- *Modifier* non-akses (*non-access Modifier*)

Modifier Akses

Modifier akses (*access Modifier*) adalah keyword yang menyatakan hak akses. Fungsi dari *Modifier* akses adalah membatasi kepada siapa saja suatu obyek dapat diakses.

- *Modifier* akses yang ditempatkan sebelum deklarasi kelas:
- default (tanpa *Modifier*) – kelas hanya dapat diakses didalam package.
- Public - kelas dapat diakses baik dari dalam package maupun luar package.

Modifier akses yang ditempatkan pada constructor, attribute dan *method*

Modifier	Dapat Diakses Dari			
	Dalam Kelas	Dalam Package	Dalam Subclass	Package Lain
private	true	false	false	false
Default (tanpa modifier)	true	true	false	false
protected	true	true	true	false
public	true	true	true	true

Modifier Non-Akses

Modifier non akses adalah *Modifier* yang tidak menyatakan hak akses, namun memiliki fungsi tertentu terhadap apa yang dideklarasikan.

- *Modifier* non-akses yang ditempatkan pada kelas :
- *final* - *Modifier* yang menyatakan bahwa kelas tidak dapat diturunkan.
- *abstract* - *Modifier* yang menyatakan bahwa kelas tidak dapat dibuat obyek.

Modifier akses yang ditempatkan pada constructor, atribut, maupun *method*

- *final* - atribut maupun *method* tidak dapat di-override dan bernilai final (tidak dapat dirubah).
- *static* - atribut maupun *method* adalah variabel kelas, bukan variabel obyek sehingga dapat diakses tanpa membuat obyeknya.
- *abstract* - *method* yang tidak memiliki tubuh dan hanya dapat digunakan pada kelas abstrak.
- *transient* - atribut maupun *method* yang dilewati ketika dilakukan proses *serialize*.
- *synchronized* - *method* yang hanya dapat diakses oleh satu thread dalam waktu yang sama.
- *volatile* - nilai dari atribut tidak dilakukan *caching* dan hanya dibaca pada memori utama.

C. Method Setter & Getter

Method setter (setter method) adalah *method* publik yang berfungsi untuk memberikan nilai dari atribut yang bersifat privat. Pada enkapsulasi setiap atribut dapat dibuat privat agar data didalamnya terlindungi. Sebagai gantinya, kita dapat membuat sebuah *method* publik sebagai jembatan agar obyek lain dapat memberikan nilai dari atribut.

Berkebalikan dengan *method setter*, *method getter (getter method)* adalah *method* yang berfungsi untuk mendapatkan nilai dari atribut *private*.

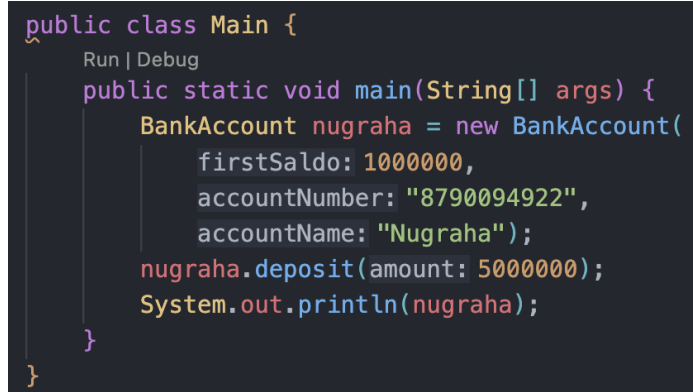
```
public class BankAccount {  
    private double saldo;  
    private String accountNumber;  
    private String accountName;  
  
    public BankAccount(double firstSaldo, String accountNumber, String accountName) {  
        this.saldo = firstSaldo;  
        this.accountNumber = accountNumber;  
        this.accountName = accountName;  
    }  
  
    public void deposit(double amount) {  
        this.saldo += amount;  
    }  
  
    public void withdraw(double amount) {  
        this.saldo -= amount;  
    }  
  
    public double getSaldo() {  
        return this.saldo;  
    }  
  
    public String getAccountNumber() {  
        return this.accountNumber;  
    }  
  
    public String toString() {  
        return  
            "Account number: " + this.accountNumber +  
            "\nAccount name: " + this.accountName +  
            "\nSaldo: " + this.saldo;  
    }  
}
```

Gambar 5.1 Contoh Enkapsulasi Pada Java

Attribute **saldo**, **accountNumber** dan **accountName** diatas adalah *private* ditandai dengan kata kunci *private* sehingga tidak dapat diakses dari luar kelas **BankAccount**. Pada Umumnya, jika dibuat sebuah rekening maka Nama pemilik dan Nomer Akun Bank hanya akan dibuat pada saat pertama kali di daftarkan dan tidak bisa diubah. Sehingga pada kasus diatas, Nomer Akun dan Nama tidak bisa diubah dikemudian hari, sehingga tidak perlu menyiapkan *method* untuk merubah akun tersebut.

Sedangkan untuk attribute saldo sifatnya berubah-ubah namun tentu saja tidak bisa berubah sembarangan. Saldo hanya bisa bertambah ketika ada transaksi debit, seperti deposit atau menerima transfer dan saldo akan berkurang ketika ada transaksi kredit seperti mengirim uang atau melakukan penarikan.

Pada Java, *Method* String `toString()` digunakan untuk memberikan informasi ketika nama objek dicetak seperti pada contoh gambar 5.2



```
public class Main {  
    Run | Debug  
    public static void main(String[] args) {  
        BankAccount nugraha = new BankAccount(  
            firstSaldo: 1000000,  
            accountNumber: "8790094922",  
            accountName: "Nugraha");  
        nugraha.deposit(amount: 5000000);  
        System.out.println(nugraha);  
    }  
}
```

Gambar 5.2 Contoh pemanggilan kelas BankAccount