Modul Praktikum
PBO

3

MODUL 3 ACCESS MODIFIER & ENCAPSULATION

1. ACCESS MODIFIER

Access Modifier ialah sebuah cara dimana kita dapat mengatur aksesibilitas terhadap sebuah field, method, constructor, atau class. Kita dapat mengganti aturan aksesibilitas dengan memasang atau menerapkan access modifier padanya.

Terdapat 4 tipe access modifier:

a. Private
   Aksesibilitas ini mengakibatkan bahwa segala sesuatu yang ada didalam suatu class hanya dapat diakses oleh class tersebut.
b. Default
   Aksesibilitas ini mengakibatkan bahwa diluar suatu package kita tidak dapat mengakses suatu class, jika kita tidak menambahkan akssesibilitas apapun maka akan menjadi default
c. Protected
   Aksesibilitas ini hampir sama dengan private namun dapat diakses oleh kelas yang menerima turunannya (child class).
d. Public
   Aksesibiltas ini membuat kita dapat mengakses dari mana saja.

TABLE ACCESS MOIFIER

	Default	Private	Protected	Public
Same Class	✓	✓	✓	✓
Same Package Subclass	✓	✗	✓	✓
Same Package Non-Subclass	✓	✗	✓	✓
Different Package Subclass	✗	✗	✓	✓
Different Package Non-Subclass	✗	✗	✗	✓

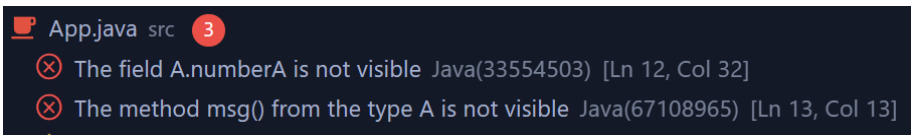
SINTAKS PRIVATE

```java
class A {
    private int numberA;

    private void msg() {
        System.out.println("Hello from class A");
    }
}

public class App {
    public static void main(String[] args) {
        A obj = new A();
        System.out.println(obj.numberA);
        obj.msg();
    }
}
```

Contoh error karena access modifier private:

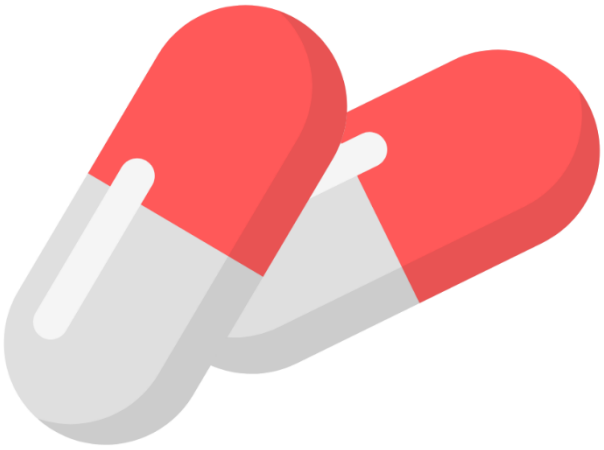


SINTAKS PROTECTED

```java
public class protectClass {
    protected void msg() {
        System.out.println("Hello from protected method");
    }
}
```

```java
public class App extends protectClass{
    public static void main(String[] args) {
        protectClass obj = new protectClass();
        obj.msg();
    }
}
```

2. ENCAPSULATION



Saat ini bayangkanlah gambar diatas dan mari kita berpikir, **mengapa zat obat yang terkandung harus diberikan *kapsul*?** Ada beragam alasan yakni untuk menjaga agar **isi** yang **sensitif** tidak **terekspos**.

Konsep enkapsulasi sama halnya dengan contoh diatas dimana data class yang ***sensitive*** tidak dapat diakses secara langsung atau di-*hidden*-kan dari sang user demi memenuhi hal-hal dibawah ini:

a. Kontrol lebih baik yakni pada atribut *class* dan *method*
b. Atribut *class* dibuat menjadi *read-only* (Jika hanya menggunakan method *get*) dan *write-only* (Jika hanya menggunakan method *set*)
c. Meningkatkan keamanan data

Penggunaan enkapsulasi yakni dengan memberikan access **modifier private** pada atribut dan penggunaan atribut public pada **method setter dan getter**