Implementasi Encapsulation dengan private, public, dan protected

Definisi Encapsulation:

- Encapsulation adalah konsep dalam OOP yang mengharuskan data (variabel) dan kode yang mengoperasikan data tersebut (method) untuk dibungkus menjadi satu unit yang disebut objek. Tujuannya adalah untuk membatasi akses langsung ke data dan menjaga integritasnya.

Keuntungan Encapsulation:

1. Keamanan Data: Melindungi data dari akses yang tidak sah dan modifikasi yang tidak diinginkan.
2. Modularitas: Mempermudah perubahan dan pemeliharaan kode.
3. Kontrol: Memberikan kontrol lebih terhadap data melalui method getter dan setter.

Access Modifiers:

- Private: Hanya dapat diakses dalam class yang sama.
- Protected: Dapat diakses dalam package yang sama dan subclass.
- Public: Dapat diakses dari mana saja.
- Default (package-private): Dapat diakses dalam package yang sama tanpa menggunakan keyword tertentu.

Contoh Kode Java yang Menunjukkan Encapsulation:

```java
public class Person {
    private String name;
    private int age;

    // Public getter for name
    public String getName() {
        return name;
    }

    // Public setter for name
    public void setName(String name) {
        this.name = name;
    }

    // Public getter for age
    public int getAge() {
        return age;
    }
```

```
    // Public setter for age
    public void setAge(int age) {
        if(age > 0) {
            this.age = age;
        } else {
            System.out.println("Age must be positive");
        }
    }
}
```

Implementasi dengan protected:

```
public class Employee extends Person {
    protected int employeeId;

    public int getEmployeeId() {
        return employeeId;
    }

    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }
}
```

5.2 Pembuatan Getter dan Setter

Definisi Getter dan Setter:

- Getter: Method yang digunakan untuk mengakses nilai dari variabel private.
- Setter: Method yang digunakan untuk mengubah nilai dari variabel private.
- Getter dan setter memberikan kontrol penuh terhadap bagaimana variabel diakses dan diubah.

Keuntungan Penggunaan Getter dan Setter:

1. Keamanan: Mengontrol akses ke variabel.
2. Validasi: Memungkinkan penambahan logika validasi sebelum mengubah nilai variabel.
3. Read-Only: Membuat variabel menjadi read-only dengan hanya menyediakan getter.

Contoh Kode Getter dan Setter:

```java
public class Person {
    private String name;
    private int age;

    // Getter for name
    public String getName() {
        return name;
    }

    // Setter for name
    public void setName(String name) {
        this.name = name;
    }

    // Getter for age
    public int getAge() {
        return age;
    }

    // Setter for age with validation
    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        } else {
            System.out.println("Age must be positive");
        }
    }
}
```

Penggunaan Getter dan Setter:

```java
public class Main {
    public static void main(String[] args) {
        Person person = new Person();
        person.setName("Alice");
        person.setAge(25);

        System.out.println("Name: " + person.getName());
        System.out.println("Age: " + person.getAge());
    }
}
```

Kesimpulan Enkapsulasi

Pada bab ini, kita telah mempelajari konsep encapsulation dalam OOP dengan Java. Encapsulation membantu melindungi data dan memastikan integritasnya dengan membatasi akses langsung melalui penggunaan access modifiers (private, protected, public). Kita juga telah melihat bagaimana membuat getter dan setter untuk mengakses dan mengubah nilai variabel private. Pemahaman konsep encapsulation sangat penting untuk membuat aplikasi yang aman dan mudah di-maintain.