MODUL 4 – Enkapsulasi dan Akses Modifier

4.1. CAPAIAN PEMBELAJARAN

1 Mahasiswa memahami konsep dasar Enkapsulasi dan Akses Modifier dalam pemrograman berorientasi objek.

2 Mahasiswa mampu mengimplementasikan Enkapsulasi dan Akses Modifier di dalam program.

4.2. ALAT DAN BAHAN

1 Seperangkat komputer lengkap/Laptop dengan koneksi internet

2 Web Browser (Chrome/Firefox/Opera/Edge/Safari/dll)

3 Aplikasi Kantor (Microsoft Office/Libre Office/WPS Office/etc)

4 JDK (https://www.oracle.com/java/technologies/downloads/)

5 Netbeans (https://netbeans.apache.org/front/main/download/)

4.3. DASAR TEORI

4.3.1. Enkapsulasi

Enkapsulasi adalah salah satu pilar utama dalam pemrograman berorientasi objek. Enkapsulasi berarti membungkus data (variabel) dan metode (fungsi) dalam satu unit yang disebut objek, di mana detail implementasi disembunyikan dari pengguna luar. Konsep ini juga disebut information hiding karena objek melindungi datanya dari modifikasi langsung dari luar, sehingga hanya dapat diakses melalui metode yang disediakan.

Manfaat Enkapsulasi:

- Keamanan Data: Mencegah perubahan data yang tidak sah.
- Kemudahan Pemeliharaan: Memisahkan antarmuka (interface) dari implementasi sehingga perubahan pada kode tidak mempengaruhi pengguna kelas.
- Modularitas: Membuat sistem lebih modular, sehingga lebih mudah dikelola dan dipahami.

Contoh penerapan enkapsulasi :

```
public class Kendaraan {
    private String merek;
    private String model;
    private int tahun;
    // Constructor
    public Kendaraan(String merek, String model, int tahun) {
```

```
        this.merek = merek;
        this.model = model;
        this.tahun = tahun;
    }
    // Getter dan Setter untuk merek
    public String getMerek() {
        return merek;
    }
    public void setMerek(String merek) {
        this.merek = merek;
    }
    // Getter dan Setter untuk model
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    // Getter dan Setter untuk tahun
    public int getTahun() {
        return tahun;
    }

    public void setTahun(int tahun) {
        this.tahun = tahun;
    }
}
```

4.3.2. Akses Modifier

Akses modifier adalah mekanisme dalam pemrograman berorientasi objek (PBO) yang digunakan untuk mengontrol visibilitas dan aksesibilitas anggota kelas (seperti variabel dan metode). Dalam Java, terdapat empat jenis akses modifier: private, protected, public, dan default (tanpa keyword). Masing-masing memiliki tingkat kontrol yang berbeda terkait siapa yang dapat mengakses anggota tersebut.

1) Private Access Modifier

Keyword: private

Visibilitas: Hanya bisa diakses dari dalam kelas yang mendefinisikannya. Anggota yang dideklarasikan sebagai private tidak dapat diakses langsung dari kelas lain, bahkan jika kelas tersebut berada dalam package yang sama.

Contoh :

```
public class Kendaraan {
    private String merk;
```

```
    public String getMerk() {
        return merk;
    }
    public void setMerk(String merk) {
        this.merk = merk;
    }
}
```

Penjelasan :

ada contoh di atas, variabel merk bersifat private, artinya tidak bisa diakses langsung dari luar kelas Kendaraan. Untuk mengakses atau mengubah nilainya, kita harus menggunakan metode getMerk() dan setMerk().

Manfaat :

- Menyembunyikan detail implementasi.
- Melindungi data dari modifikasi yang tidak disengaja atau tidak valid.

2) Protected Access Modifier

Keyword: protected

Visibilitas: Anggota yang dideklarasikan sebagai protected dapat diakses oleh:

- Kelas yang berada di package yang sama.
- Subclass (kelas yang mewarisi) bahkan jika berada di package berbeda.

Contoh :

```
public class Komputer {
    protected String prosesor;
    public void tampilkanProsesor() {
        System.out.println("Prosesor: " + prosesor);
    }
}
public class Laptop extends Komputer {
    public void tampilkanSpesifikasi() {
        System.out.println("Laptop menggunakan
prosesor: " + prosesor);
    }
}
```

Penjelasan :

Pada contoh di atas, variabel prosesor bersifat protected. Kelas Laptop yang

merupakan subclass dari Komputer dapat mengakses variabel prosesor secara langsung, meskipun kelas Laptop berada di package yang berbeda.

Manfaat :

protected memungkinkan akses lebih luas dalam hierarki kelas (inheritance), tetapi tetap menjaga perlindungan dari akses langsung oleh kelas non-subclass.

3) Public Access Modifier

Keyword: public

Visibilitas: Anggota yang dideklarasikan sebagai public dapat diakses dari mana saja, baik dari kelas yang berada di package yang sama maupun package yang berbeda. Akses ini sangat terbuka.

Contoh :

```
public class Kendaraan {
    public String jenisMesin;
    public void tampilkanJenisMesin() {
        System.out.println("Jenis Mesin: " +
jenisMesin);
    }
}
public class Main {
    public static void main(String[] args) {
        Kendaraan mobil = new Kendaraan();
        mobil.jenisMesin = "Bensin";
        mobil.tampilkanJenisMesin(); // Output: Jenis
Mesin: Bensin
    }
}
```

Penjelasan:

Variabel jenisMesin bersifat public, sehingga bisa diakses langsung dari luar kelas Kendaraan tanpa harus menggunakan metode getter/setter.

Manfaat:

Memudahkan akses untuk anggota yang bersifat umum atau tidak memerlukan kontrol khusus.

4) Default (Package-Private) Access Modifier

Keyword: Tidak ada (tidak menuliskan keyword apa pun).

Visibilitas: Anggota yang dideklarasikan tanpa modifier (default) hanya bisa diakses oleh kelas yang berada dalam package yang sama. Tidak bisa diakses oleh kelas yang berada di luar package, bahkan oleh subclass.

```
class Produk {
    String namaProduk;  // Default, tidak ada keyword
public/protected/private
    void tampilkanProduk() {
        System.out.println("Nama Produk: " +
namaProduk);
    }
}
```

Penjelasan:

Variabel namaProduk di atas memiliki visibilitas default. Artinya, hanya kelas-kelas yang berada dalam package yang sama dengan Produk yang dapat mengaksesnya. Kelas dari package lain tidak dapat mengakses anggota ini.

Manfaat:

Mengamankan anggota kelas untuk digunakan hanya dalam package yang sama, cocok untuk menjaga modularitas.

Perbandingan akses modifier dapat dilihat pada .

Tabel 1. Perbandingan Akses Modifier

Modifier	Visibilitas dalam Kelas	Visibilitas dalam Package	Visibilitas di Subclass	Visibilitas dari Luar Package
private	Ya	Tidak	Tidak	Tidak
default	Ya	Ya	Tidak	Tidak
protected	Ya	Ya	Ya	Ya, hanya di subclass
public	Ya	Ya	Ya	Ya

Kasus Penggunaan Akses Modifier :

Private:

Kapan digunakan? Gunakan private ketika Anda ingin sepenuhnya melindungi anggota kelas dari akses luar. Ini biasanya digunakan untuk properti yang tidak ingin diakses atau dimodifikasi langsung oleh pengguna kelas lain, seperti atribut penting dalam objek seperti password, saldo bank, atau detail penting lainnya.

Protected:

Kapan digunakan? Gunakan protected ketika Anda ingin memberikan akses kepada subclass atau kelas-kelas lain dalam package yang sama, tetapi tetap melindungi anggota dari akses luar package. Biasanya digunakan dalam sistem warisan (inheritance) di mana kelas turunan membutuhkan akses ke atribut tertentu dari kelas induk.

Public:

Kapan digunakan? Gunakan public ketika Anda ingin membuat anggota kelas dapat diakses dari mana saja. Cocok untuk metode atau atribut yang bersifat umum, seperti metode toString(), atau konstanta yang perlu diakses secara global.

Default:

Kapan digunakan? Gunakan akses default ketika Anda hanya ingin anggota kelas diakses oleh kelas-kelas dalam package yang sama, tetapi tidak ingin memberikan akses luar package. Ini cocok digunakan pada proyek besar untuk menjaga modularitas dan batasan antar package.

4.4. PRAKTIKUM

2.4.1. Perisapan

1) Buka Netbeans yang sudah terinstall pada komputer
2) Buat proyek baru dengan nama PraktikumPBO_4.
3) Buat package baru dengan nama praktikum4.

2.4.2. Penerapan Enkapsulasi dan Akses Modifier

1) Buat kelas Kendaraan yang menggunakan enkapsulasi untuk melindungi datanya:

```java
package Praktikum4;

/**
 *
 * @author HP 14
 */
public class Kendaraan {
    private String merek;
    private String model;
    private int tahun;

    // Constructor
    public Kendaraan(String merek, String model, int tahun) {
        this.merek = merek;
        this.model = model;
        this.tahun = tahun;
    }
    // Getter dan Setter untuk merek
    public String getMerek() {
        return merek;
    }
    public void setMerek(String merek) {
```

```java
    public void setMerek(String merek) {
        this.merek = merek;
    }
    // Getter dan Setter untuk model
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    // Getter dan Setter untuk tahun
    public int getTahun() {
        return tahun;
    }
    public void setTahun(int tahun) {
        this.tahun = tahun;
    }
}
```

2) Buat objek dari kelas Kendaraan dan tampilkan data menggunakan getter.

3) Ubah data menggunakan setter, dan lihat bagaimana data diatur melalui metode yang disediakan.

```java
package Praktikum4;

/**
 *
 * @author HP 14
 */
public class Praktikum_PBO4 {

    public static void main(String[] args) {
        Kendaraan mobil = new Kendaraan("Toyota", "Avanza", 2020);

        // Menampilkan data awal
        System.out.println("Merek: " + mobil.getMerek());
        System.out.println("Model: " + mobil.getModel());
        System.out.println("Tahun: " + mobil.getTahun());

        // Mengubah data
        mobil.setModel("Innova");
        mobil.setTahun(2021);

        // Menampilkan data setelah perubahan
        System.out.println("Model baru: " + mobil.getModel());
        System.out.println("Tahun baru: " + mobil.getTahun());
    }
}
```

4) Ubah isi dari class kendaraan untuk menerapkan akses modifier seperti berikut :

```java
 * @author ImamAdiNata#root
 */
public class Kendaraan {
    // Atribut dengan akses modifier berbeda
    private String nama;              // Hanya bisa diakses dalam kelas ini
    protected int kecepatanMaks;      // Bisa diakses di package yang sama dan subclass
    public String jenisMesin;         // Bisa diakses dari mana saja

    // Constructor
    public Kendaraan(String nama, int kecepatanMaks, String jenisMesin) {
        this.nama = nama;
        this.kecepatanMaks = kecepatanMaks;
        this.jenisMesin = jenisMesin;
    }
    // Getter dan Setter untuk variabel private nama
    public String getNama() {
        return nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
    // Method public untuk menampilkan informasi kendaraan
    public void tampilkanInfoKendaraan() {
        System.out.println("Nama Kendaraan: " + nama);
        System.out.println("Kecepatan Maksimum: " + kecepatanMaks + " km/h");
        System.out.println("Jenis Mesin: " + jenisMesin);
    }
}
```

5) Buat class baru dengan nama mobil untuk percobaan akses modifier protected dalam subclass :

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change th
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package Praktikum4;

/**
 *
 * @author ImamAdiNata#root
 */
public class Mobil extends Kendaraan {
    private int jumlahPintu;  // Atribut tambahan khusus untuk mobil

    // Constructor
    public Mobil(String nama, int kecepatanMaks, String jenisMesin, int jumlahPintu) {
        super(nama, kecepatanMaks, jenisMesin);  // Memanggil constructor dari kelas induk
        this.jumlahPintu = jumlahPintu;
    }

    // Method untuk menampilkan informasi mobil
    public void tampilkanInfoMobil() {
        // Dapat mengakses kecepatanMaks karena protected
        System.out.println("Kecepatan Maksimum Mobil: " + kecepatanMaks + " km/h");
        System.out.println("Jumlah Pintu: " + jumlahPintu);
    }
}
```

6) Coba buat objek ke dalam class main seperti berikut :

4.5. TUGAS MODUL 3

4.5.1. Soal

- Buat Kelas Manusia:
  - Kelas ini memiliki atribut nama, usia, dan pekerjaan.
  - Gunakan akses modifier yang tepat untuk setiap atribut:
  - nama harus bersifat private.
  - usia bersifat protected.
  - pekerjaan bersifat public.
  - Buat constructor untuk menginisialisasi atribut nama, usia, dan pekerjaan.
  - Buat metode getter dan setter untuk atribut nama.
- Buat Kelas Pekerja yang Mewarisi Manusia:
  - Kelas ini harus memiliki atribut tambahan gaji (gunakan akses modifier private).
  - Buat constructor untuk menginisialisasi atribut nama, usia, pekerjaan, dan gaji.
  - Buat metode getter dan setter untuk atribut gaji.
  - Override metode toString() untuk menampilkan semua informasi tentang pekerja, termasuk nama, usia, pekerjaan, dan gaji.
- Buat Kelas Main:

- Buat objek dari kelas Pekerja dan inisialisasi dengan nama, usia, pekerjaan, dan gaji.
- Tampilkan informasi pekerja dengan menggunakan metode toString().
- Ubah nama pekerja menggunakan metode setter dan tampilkan ulang informasi pekerja.
- Coba akses langsung atribut nama, usia, dan gaji dari objek pekerja. Apa yang terjadi? Jelaskan mengapa hal tersebut terjadi.

- Push ke github masing-masing (Tautan Github cantumkan pada laporan medium)

4.5.2. Petunjuk Pengerjaan

a) Laporan:

- Buatlah laporan akhir sesuai dengan praktikum yang dilakukan
- Tuliskan laporan ke dalam akun masing-masing web medium
- Publikasikan laporan tersebut
- Kirimkan tautan laporan ke elita.
- **Batas Pengumpulan**: Sebelum Pertemuan Praktik Ke 5.