Nama : Herdiansyah Suhendar
Kelas : 4IA17
NPM : 50421604

Activity M2 Praktikum Rekayasa Perangkat Lunak 2

1. Jelaskan anatomi dasar dari sebuah class dalam Java. Apa saja komponen utama yang ada dalam class dan bagaimana fungsinya? Berikan contoh sederhana dari sebuah class dengan atribut dan method di dalamnya.

Jawab :

Dalam Java, sebuah class adalah template atau blueprint yang mendefinisikan atribut (data atau variabel) dan method (fungsi) dari sebuah objek. Berikut adalah anatomi dasar dari sebuah class dan komponen-komponennya:

Komponen Utama dalam Sebuah Class

1. Deklarasi Class
    - Dimulai dengan kata kunci `class` diikuti dengan nama class. Nama class biasanya diawali dengan huruf kapital dan mengikuti aturan penamaan variabel.
    - Contoh: `public class Person`
2. Atribut (Fields atau Variables)
    - Atribut atau fields menyimpan data atau properti dari sebuah objek. Atribut ini bisa berupa variabel instance atau variabel statis.
    - Variabel instance adalah variabel yang terikat pada setiap objek yang dibuat dari class tersebut.
    - Variabel statis adalah variabel yang terikat pada class itu sendiri, bukan pada objek-objeknya.
3. Method
    - Method adalah blok kode yang menjalankan aksi atau operasi tertentu. Method berfungsi untuk mendefinisikan perilaku objek.
    - Sebuah method biasanya memiliki nama, tipe kembalian (misalnya `void` jika tidak mengembalikan nilai), dan parameter opsional.
4. Constructor
    - Constructor adalah method khusus yang digunakan untuk menginisialisasi objek baru dari class. Constructor memiliki nama yang sama dengan nama class dan tidak memiliki tipe kembalian.
    - Jika tidak ada constructor yang didefinisikan, Java akan menyediakan constructor default tanpa parameter secara otomatis.

5. Access Modifiers
    - Digunakan untuk mengatur visibilitas dari class, atribut, dan method.
    - Contoh access modifiers:

- `public`: Dapat diakses dari mana saja.
- `private`: Hanya dapat diakses dalam class yang sama.
- `protected`: Dapat diakses oleh class di paket yang sama atau oleh subclass.
- Tidak ada modifier (default): Dapat diakses oleh class di paket yang sama.

Contoh Sederhana Class dalam Java

```
// Deklarasi class bernama Person
public class Person {

    // Atribut atau field
    private String name;  // Variabel instance
    private int age;

    // Constructor
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Method untuk mendapatkan nama
    public String getName() {
        return name;
    }

    // Method untuk mendapatkan usia
    public int getAge() {
        return age;
    }

    // Method untuk mencetak informasi dari objek
    public void printInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }

    // Static method yang tidak terikat ke objek manapun
    public static void greeting() {
        System.out.println("Hello! Welcome to the Java class.");
    }
}
```

Penjelasan Contoh:

2. Deklarasi Class: `public class Person` mendefinisikan class dengan nama `Person`.
3. Atribut (Fields): `private String name` dan `private int age` adalah atribut dari class `Person` yang menyimpan nama dan usia.
4. Constructor: `public Person(String name, int age)` adalah constructor yang digunakan untuk menginisialisasi nilai atribut `name` dan `age` saat objek baru dibuat.
5. Method:
    a. `getName()` dan `getAge()` adalah method yang mengembalikan nilai atribut.
    b. `printInfo()` adalah method untuk mencetak informasi tentang objek `Person`.
    c. `greeting()` adalah static method yang dapat dipanggil tanpa perlu membuat objek `Person`.

Cara Menggunakan Class `Person`

```
public class Main {
    public static void main(String[] args) {
        // Membuat objek baru dari class Person
        Person person1 = new Person("Alice", 30);

        // Memanggil method printInfo
        person1.printInfo();

        // Memanggil static method greeting
        Person.greeting();
    }
}
```

Output:
Name: Alice
Age: 30
Hello! Welcome to the Java class.

Kesimpulan:
Class dalam Java adalah struktur dasar yang digunakan untuk membentuk objek. Atribut menyimpan data tentang objek, sementara method memberikan perilaku atau aksi yang bisa dilakukan oleh objek. Dengan menggunakan constructor, kita dapat menginisialisasi objek dengan nilai awal. Static method berguna untuk fungsi yang tidak bergantung pada instansiasi objek.

2. Screenshot code, output, dan beri penjelasan singkat dari program yang telah dibuat pada video!

Jawab :

Source code

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 */

package pert2_50421604;

/**
 *
 * @author Heansy
 */
public class Pert2_50421604 {

    public static void main(String[] args) {
        Mahasiswa mahasiswa = new Mahasiswa("Herdiansyah Suhendar", "50421604", 23);
        mahasiswa.tampilkanData();
        System.out.println();

        MahasiswaSarjana mahasiswaSarjana = new MahasiswaSarjana("Herdiansyah Suhendar", "50421604", 23, "Informatika");
        mahasiswaSarjana.tampilkanData();
        System.out.println();
    }
```

```java
package pert2_50421604;

/**
 *
 * @author Heansy
 */
public class Mahasiswa {
    private String nama;
    private String npm;
    private int umur;

    public Mahasiswa(String nama, String npm, int umur){
        this.nama = nama;
        this.npm = npm;
        this.umur = umur;
    }

    public void tampilkanData() {
        System.out.println("Nama: "+ nama);
        System.out.println("NPM: "+ npm);
        System.out.println("Umur: "+ umur);
    }
```

```java
package pert2_50421604;

/**
 *
 * @author Heansy
 */
public class MahasiswaSarjana extends Mahasiswa {
    private String jurusan;

    public MahasiswaSarjana (String nama, String npm, int umur, String jurusan) {
        super(nama, npm, umur);
        this.jurusan = jurusan;
    }

    @Override
    public void tampilkanData() {
        super.tampilkanData();
        System.out.println("Jurusan: " + jurusan);
    }
}
```

Output

Penjelasan Kode:

Inheritance (Pewarisan):

Inheritance adalah konsep di mana sebuah class mewarisi properti (atribut) dan perilaku (method) dari class lain. Dalam kode di atas:

- Class `MahasiswaSarjana` adalah subclass atau class turunan yang mewarisi properti `nama`, `npm`, dan `umur` serta method `tampilkanData()` dari class `Mahasiswa` (superclass).
- Melalui pewarisan ini, class `MahasiswaSarjana` dapat menggunakan semua properti dan method dari class `Mahasiswa` tanpa harus mendeklarasikannya ulang.

Contoh inheritance:

```
public class MahasiswaSarjana extends Mahasiswa {
    // MahasiswaSarjana mewarisi properti dan method dari
Mahasiswa
}
```

Encapsulation (Enkapsulasi):

Encapsulation adalah konsep OOP yang menyembunyikan detail implementasi dan hanya memberikan akses kepada user melalui method publik. Pada kode di atas:

- Atribut `nama`, `npm`, dan `umur` dalam class `Mahasiswa` ditetapkan sebagai `private`, artinya hanya bisa diakses dari dalam class itu sendiri.
- Akses terhadap atribut tersebut dilakukan melalui method `tampilkanData()`, sehingga pengguna tidak bisa langsung mengakses atau memodifikasi nilai atribut secara langsung.

Contoh enkapsulasi:

```
private String nama;
private String npm;
private int umur;
// Akses terhadap atribut dilakukan melalui method publik
public void tampilkanData() {
    System.out.println("Nama: " + nama);
}
```

Polymorphism (Polimorfisme):

Polymorphism memungkinkan sebuah objek untuk mengambil banyak bentuk (multiple forms), terutama dalam hal method overriding. Dalam kode di atas, method `tampilkanData()` di-*override* di class `MahasiswaSarjana` untuk menampilkan data tambahan, yaitu `jurusan`, meskipun method dengan nama yang sama ada di class `Mahasiswa`.

- Method Overriding: Method `tampilkanData()` di class `MahasiswaSarjana` menggantikan implementasi yang ada di class induknya (`Mahasiswa`) dan menambahkan informasi tambahan.

Contoh polymorphism (method overriding):

```
@Override
public void tampilkanData() {
    // Memanggil method tampilkanData dari class Mahasiswa
    super.tampilkanData();
    System.out.println("Jurusan: " + jurusan);  // Menambahkan
perilaku baru
}
```

Kesimpulan:

- Inheritance: Class `MahasiswaSarjana` mewarisi semua atribut dan method dari class `Mahasiswa`.
- Encapsulation: Atribut pada class `Mahasiswa` seperti `nama`, `npm`, dan `umur` disembunyikan dan hanya bisa diakses melalui method.
- Polymorphism: Method `tampilkanData()` di class `MahasiswaSarjana` di-*override* untuk menampilkan data tambahan yang spesifik untuk mahasiswa sarjana.

Kode ini adalah contoh sederhana bagaimana OOP diterapkan dalam Java untuk menciptakan struktur program yang lebih terorganisir, reusable, dan modular.