BAB 3
Attribute, Behaviour dan Constructor

Dalam materi kali ini, kita akan belajar tentang *attribute*, *behaviour* dan *constructor* pada pemrograman berorientasi objek. *Attribute* merupakan variabel yang terdapat pada sebuah *class*, sedangkan behavior merupakan metode atau fungsi yang dapat dijalankan pada sebuah objek. Kalian juga akan mempelajari tentang *constructor*, yaitu metode khusus yang digunakan untuk membuat objek dari sebuah *class*.

Dengan memahami konsep-konsep dasar ini, kalian akan dapat membangun program yang lebih kompleks dan lebih terstruktur. Selain itu, pengetahuan tentang *attribute*, behavior, dan *constructor* juga sangat berguna untuk mengembangkan aplikasi berbasis objek yang lebih efektif dan efisien.

A. Attribute

Attribute adalah variabel yang dideklarasikan di dalam sebuah *class*. Dalam bahasa pemrograman Java, *attribute* dideklarasikan di dalam *class* dan dapat diakses oleh seluruh *method* di dalam *class* tersebut. *Attribute* dapat memiliki tipe data apa saja, seperti int, double, boolean, atau bahkan tipe data yang dibuat sendiri (user-defined data *type*), seperti *class* lain.

Pada bahasa Java, *attribute* memiliki beberapa ketentuan sebagai berikut:

- *Attribute* dideklarasikan di dalam *Class*, tetapi di luar *method* dan *constructor*.
- *Attribute* dibuat ketika sebuah objek dari *class*nya dibuat dengan *keyword* new dan terhapus ketika objeknya dihapus.
- *Attribute* dapat diberikan access modifier.
- *Attribute* dalam sebuah *class* dapat diaksess dari semua *method* atau *constructor* dalam *class* tersebut. Direkomendasikan untuk memberikan access modifier private untuk setiap *attribute* dalam *class*.
- *Attribute* memiliki nilai default. Untuk angka nilai defaultnya adalah 0, untuk Boolean nilai defaultnya adalah *false*, dan untuk sebuah objek reference nilai defaultnya adalah *null*. Inisialisasi sebuah *attribute* biasanya dilakukan pada *Constructor*.

Berikut ini adalah contoh deklarasi *attribute* dalam sebuah *class* pada bahasa pemrograman Java:

```java
public class Mahasiswa {
  String nama;
  int usia;
  double ipk;
}
```

Class diagramnya:

Mahasiswa
nama : *String* usia : int ipk : double

Pada *Class* Mahasiswa di atas, terdapat tiga *attribute* dengan tipe data yang berbeda-beda. Terdapat *attribute* nama dengan tipe data *String*, *attribute* usia dengan tipe data *integer* dan *attribute* ipk dengan tipe data double.

1. Keyword THIS

Keyword this pada pemanggilan *attribute* dalam *class* pada bahasa pemrograman Java digunakan untuk merujuk pada objek saat ini yang sedang diproses. Hal ini membantu membedakan antara variabel lokal dan *attribute* dari *class* yang sama dengan nama yang sama, sehingga mencegah kesalahan logika dan membuat kode program lebih mudah dipahami dan dijaga kualitasnya.

Contoh kasus penggunaan *keyword this* pada pemanggilan *attribute* dalam *class* pada bahasa pemrograman Java adalah sebagai berikut:

```java
public class Mahasiswa {
    private String nama;

    public Mahasiswa(String nama) {
        this.nama = nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNama() {
        return this.nama;
    }
}
```

Dalam contoh di atas, terdapat *attribute* "nama" pada *class* Mahasiswa. Pada *constructor* dan *method* `setNama()` , parameter yang diterima juga memiliki nama yang sama dengan *attribute* "nama".

Untuk membedakan antara parameter lokal dan *attribute* dari *class* yang sama dengan nama yang sama, maka digunakanlah *keyword this*. Pada *constructor*, `this.nama` merujuk pada *attribute* "nama" dari *class* Mahasiswa, sedangkan pada *method* `setNama()`, `this.nama` juga merujuk pada *attribute* "nama" dari *class* Mahasiswa.

Dengan penggunaan *this*, kita dapat memastikan bahwa nilai yang diberikan ke parameter "nama" akan disimpan pada *attribute* "nama" dari *class* Mahasiswa. Hal ini membantu mencegah kesalahan logika dan membuat kode program lebih mudah dipahami dan terjaga kualitasnya.

B. Behaviour

Behaviour pada pemrograman berorientasi objek mengacu pada tindakan atau perilaku yang dapat dilakukan oleh objek dalam sebuah *class*. Dalam bahasa pemrograman Java, perilaku atau *behaviour* ini diimplementasikan menggunakan fungsi *class* atau *method*. Setiap objek dalam *class* dapat memiliki berbagai macam metode yang dapat dipanggil untuk melakukan tindakan tertentu.

Selain untuk mendeskripsikan sebuah tingkah laku dari objek, *method* dalam sebuah *class* juga dapat digunakan sebagai sarana untuk mengambil (get) nilai suatu *attribute* atau untuk mengubah (set) nilai suatu *attribute* private.

Berikut adalah sebuah *class* Mahasiswa yang memiliki beberapa metode untuk mengimplementasikan perilaku atau *behaviour*-nya:

```
public class Mahasiswa {
    String nama,nim;
    double ipk;

    public void tampilkanInfo() {
        System.out.println("Nama : " + nama);
        System.out.println("NIM : " + nim);
        System.out.println("IPK : " + ipk);
    }

    public String hitungIndeksPrestasi() {
        String indeksPrestasi = "";
        if (ipk >= 3.5) {
            indeksPrestasi = "Cumlaude";
        } else if (ipk >= 3.0) {
            indeksPrestasi = "Sangat Memuaskan";
        } else if (ipk >= 2.75) {
            indeksPrestasi = "Memuaskan";
        } else if (ipk >= 2.0) {
            indeksPrestasi = "Cukup";
        } else {
            indeksPrestasi = "Tidak Lulus";
        }

        return indeksPrestasi;
    }}
```

Class diagramnya:

Mahasiswa
nama : *String* nim : *String* ipk : double

tampilkanInfo() hitungIndeksPrestasi(): *String*

Dalam *class* Mahasiswa di atas, terdapat beberapa metode yang meng implementasikan perilaku atau *behaviour*-nya, yaitu:

1. `tampilkanInfo()` adalah *method* untuk menampilkan informasi mahasiswa seperti nama, NIM, dan IPK.
2. `hitungIndeksPrestasi()` adalah *method* untuk menghitung indeks prestasi dari mahasiswa berdasarkan IPK-nya. *Method* ini akan mengembalikan *String* yang berisi keterangan indeks prestasi, seperti "Cumlaude", "Sangat Memuaskan", dan sebagainya.

Dengan menggunakan *class* Mahasiswa di atas, kita dapat membuat objek-objek Mahasiswa dan memanggil metode-metode yang telah diimplementasikan di dalamnya. Berikut adalah contoh penggunaan *class* Mahasiswa:

```
public class Main {
  Run | Debug
  public static void main(String[] args) {
    // Membuat objek Mahasiswa
    Mahasiswa mahasiswa = new Mahasiswa();

    // Mengisi nilai atribut nya
    mahasiswa.nama= "Muh. Yusuf Syam";
    mahasiswa.nim= "H071191044";
    mahasiswa.ipk= 3;

    // Memanggil method tampilkanInfo()
    System.out.println("Info:");
    mahasiswa.tampilkanInfo();

    // Memanggil method hitungIndeksPrestasi()
    System.out.println("\nIndeks Prestasi: "+mahasiswa.hitungIndeksPrestasi());
  }
}
```

Output:

```
Info:
Nama : Muh. Yusuf Syam
NIM : H071191044
IPK : 3.0

Indeks Prestasi: Sangat Memuaskan
```

C. Constructor

Pada bahasa pemrograman Java, *constructor* adalah sebuah blok statements yang pertama kali dieksekusi dalam sebuah *class*. *Constructor* sangat mirip dengan *method* tetapi tidak memiliki return *type* dan nama *constructor* harus sama dengan sama *class*nya. Setiap *class* memiliki *constructor*.

Jika kita tidak menulis sebuah *constructor* pada sebuah *class* maka Java compiler akan membuatkan *constructor* default untuk *class* tersebut. Setiap kali sebuah objek baru dibuat maka paling sedikit satu *constructor* dipanggil. Sebuah *class* dapat memiliki lebih dari satu *constructor*. *Constructor* sering digunakan untuk inisialisasi instance variable atau *attribute* dalam sebuah *class*. Berikut ini contoh sebuah *constructor* dari *class* mahasiswa:

```java
public Mahasiswa(String nama, String nim, double ipk) {
  this.nama = nama;
  this.nim = nim;
  this.ipk = ipk;
}
```

Dan berikut contoh penggunaannya saat membuat objek

```java
Mahasiswa mahasiswa = new Mahasiswa("Muh. Yusuf Syam", "H071191044", 3);
```

Saat objek mahasiswa dibuat, maka nilai *attribute* nama menjadi “Muh. Yusuf Syam”, *attribute* nim menjadi “H071191044” dan *attribute* ipk menjadi 3. Perhatikan bahwa saat menggunakan *constructor* urutan parameter harus sesuai.

1. Multiple Constructor

Sebuah *class* dapat memiliki lebih dari satu *constructor* tetapi nama *constructor* harus sama dengan nama *class*. Java memperbolehkan sebuah *class* memiliki banyak *constructor* dengan ketentuan:

- Setiap *Constructor* memiliki jumlah parameter yang berbeda
- Jumlah parameter boleh sama tetapi *type* data masing-masing parameternya harus berbeda.
- Apabila terdapat lebih dari satu *constructor* dalam *class* maka *Constructor* yang digunakan adalah *constructor* yang sesuai dengan ketika objek di-instansikan.

Berikut contoh multiple *constructor* pada *class* mahasiswa.

```
public Mahasiswa() {}

public Mahasiswa(String nama) {
   this.nama = nama;
 }

public Mahasiswa(String nama, String nim, double ipk) {
  this.nama = nama;
  this.nim = nim;
  this.ipk = ipk;
}
```

Dapat dilihat pada kode di atas, terdapat tiga *constructor* yang masing-masing memiliki jumlah parameter berbeda. Misalkan saat kita membuat objek dengan *constructor* seperti di bawah:

```
Mahasiswa mahasiswa = new Mahasiswa("Muh. Yusuf Syam");
```

Maka Java secara otomatis mengarahkan ke *constructor* kedua, di mana hanya *attribute* nama yang diinisialisasikan.

Tugas Praktikum

1. Seperti kelas sebelumnya, Buatlah Sebuah *class* dengan syarat:

    - Memiliki minimal 3 *attribute* , 2 *behaviour* , dan 2 constructor
    - Memenuhi salah satu dari syarat di bawah:
        - Salah satu *attribute* harus berupa user defined variable atau bertipe data objek, sebagai contoh pada *class* Mahasiswa pada soal no.5 tugas minggu lalu terdapat *attribute* alamat yang merupakan objek dari class lain yang dibuat. *Attribute* bisa juga merupakan objek dari class yang sama.
        - Salah satu behaviour dari *class* harus berinteraksi dengan attribute user defined variable. Sebagai contoh fungsi `balap()` pada *class* HotWheels pada contoh di bawah yang menerima parameter lawan yang merupakan objek dari HotWheels.
    - Perhatikan bahwa method setter dan getter tidak dianggap sebagai behaviour

    Class harus berbeda untuk tiap orang dan tidak digunakan di soal praktikum. Jadi untuk konfirmasi nama kelas silahkan isi dan cek di spreadsheet berikut : https://docs.google.com/spreadsheets/d/1HZQfSZtY5Pu1FwZSkdb76SQdnNCvTo0h68Bd5GNTO8U/edit?usp=sharing

    Contoh *class* yang memenuji syarat-syarat di atas:

HotWheels
nama : *String* kecepatan : double jumlahMenang : int cash: double
tampilkanSpek() upgrade(); balap()

    Kode dari *class* tersebut dapat dibuka pada :
    https://github.com/YusufSyam/Lab_OOP-Praktikan_Grup_K/tree/main/DLL/HotWheels

2. Lengkapi Program berikut sehingga memiliki output yang sesuai dengan contoh output:

```java
class Player{
  private String name;
  private int hp;
  private int attackPower;
  private int defense;

  public Player(){}

  public void getDamage(Player enemy) {
    hp= hp-Math.abs(enemy.getAttackPower()-defense);
  }

  // Lengkapi

   public void status() {
    System.out.println(name + " status");
    System.out.println("HP = " + hp);
    System.out.println("Defense = " + defense);
    System.out.println("Attack = " + attackPower + "\n");
  }


  int getAttackPower() {
    return attackPower;
  }

}

public class Main {
  public static void main(String [] args) {

    Player player1= new Player("Mino", 30, 15);
    Player player2= new Player("Hilda", 10);

    player2.setAttackPower(35);

    player1.getDamage(player2);

    player1.status();
    player2.status();
  }
}
```

Output yang diinginkan:

```
Mino status
HP = 80
Defense = 15
Attack = 30

Hilda status
HP = 100
Defense = 10
Attack = 35
```