BAB III

Attribute, Behavior dan Constructor

Pada materi kali ini, kita akan belajar tentang *attribute, behavior,* dan *constructor* pada pemrograman berorientasi objek. Apa arti mereka semua? Mari kita telaah satu per satu.

1. *Attribute* atau dalam Bahasa Indonesia disebut atribut, merupakan variabel yang terdapat pada sebuah *class*.
2. *Behavior* atau perilaku, merupakan metode atau fungsi yang dapat dijalankan pada suatu objek.
3. *Constructor* ialah 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 tentunya efisien.

A. Attribute

1. Apa itu *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, bisa tipe data String, int, double, long, boolean, ataupun tipe data yang dibuat sendiri (*user-defined* data *type*) seperti class lain.

Pada bahasa pemrograman 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* tersebut dibuat dengan keyword `new` dan terhapus ketika objeknya dihapus.
- *Attribute* dapat diberikan *access modifier*.

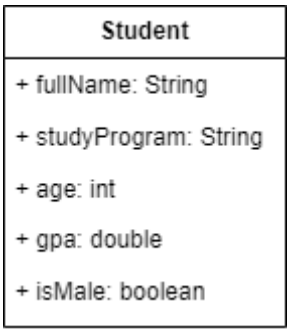
- *Attribute* dalam sebuah *class* dapat diakses dari semua *method* atau *constructor* dalam class tersebut. Direkomendasikan untuk memberikan *access modifier* `private` untuk setiap *attribute* dalam class (akan dipelajari lebih lanjut pada pertemuan selanjutnya).
- *Attribute* memiliki nilai *default*. Nilai *default* dari setiap jenis tipe data ialah sebagai berikut:
    a. Untuk tipe data jenis angka, nilai defaultnya adalah 0.
    b. Tipe data boolean nilai defaultnya adalah false.
    c. Sebuah tipe data objek *reference* (salah satunya ialah String), nilai defaultnya adalah null.

  Inisialisasi sebuah *attribute* biasanya dilakukan pada *constructor*.

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

```
public class Student {
    String fullName;
    String studyProgram;
    int age;
    double gpa;
    boolean isMale;
}
```

Lalu *class* diagramnya ialah sebagai berikut:



Pada *class* Student di atas, terdapat lima *attribute* dengan tipe data yang berbeda-beda. Terdapat *attribute* nama lengkap dan program studi dengan tipe data String, *attribute* usia dengan tipe data integer, *attribute* ipk dengan tipe data double, dan *attribute* apakah laki-laki dengan tipe data boolean.

2. *Keyword* `this`

Dalam bahasa pemrograman Java, terdapat juga *keyword* `this`. *Keyword* `this` pada pemanggilan *attribute* dalam *class* pada bahasa pemrograman Java digunakan untuk merujuk pada objek saat ini yang sedang diproses. Hal tersebut 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:

```
public class Student {
    String fullName;

    public Student(String fullName) {
        this.fullName = fullName;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
}
```

Dalam contoh di atas, terdapat *attribute* “fullName” pada *class* Student. Pada *constructor* dan *method* setFullName(), parameter yang diterima juga memiliki nama yang sama dengan *attribute* "fullName". Untuk membedakan antara parameter lokal dan *attribute* dari *class* yang sama dengan nama yang sama pula, maka digunakanlah keyword `this`. Pada *constructor*, this.fullName merujuk pada attribute "fullName" dari *class* Mahasiswa, sedangkan pada *method* setFullName(), this.fullName pun juga merujuk pada *attribute* "fullName" 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. Behavior

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 di-implementasikan menggunakan fungsi *class* atau *method*. Setiap objek dalam *class* dapat memiliki berbagai macam metode (*method*) 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* (terkait hal *attribute private* akan dibahas lebih lanjut pada pertemuan berikutnya). Berikut adalah sebuah *class* Student yang memiliki beberapa metode untuk mengimplementasikan perilaku atau *behaviour* :

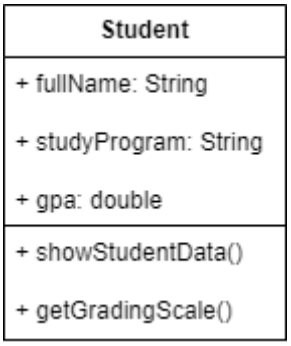
```
public class Student {
  String fullName, studyProgram;
  double gpa;

  public void showStudentData() {
     System.out.println("Full Name: " + fullName);
     System.out.println("Study Program: " + studyProgram);
     System.out.println("GPA: " + gpa);
  }

  public String getGradingScale() {
     String gradingScale = "";
     if (gpa >= 3.5) {
        gradingScale = "Cum Laude";
     } else if (gpa >= 3.0) {
        gradingScale = "Magna Cum Laude";
     } else if (gpa >= 2.5) {
        gradingScale = "Good";
     } else {
        gradingScale = "Not Meeting Criteria";
     }
     return gradingScale;
```

```
  }
}
```

Class diagramnya ialah sebagai berikut:



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

a. showStudentData( ) -> *method* untuk menampilkan informasi mahasiswa seperti fullName, studyProgram, dan gpa.
b. getGradingScale() -> *method* untuk mendapatkan *grading scale* / skala penilaian dari mahasiswa berdasarkan IPK-nya. *Method* ini akan mengembalikan String yang berisi keterangan skala penilaian, seperti "Cum Laude", "Magna Cum Laude", dan sebagainya.

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

```
public class Student {
  String fullName, studyProgram;
  double gpa;

  public static void main(String[] args) {
    // Create a new instance object of Student class
    Student student = new Student();

    // Fill in the attribute values
    student.fullName = "Muh. Adnan Putra A.";
    student.studyProgram = "Information System";
    student.gpa = 4.00;
```

```
    // Call the showStudentData() method
    System.out.println("---> Information <---");
    student.showStudentData();

    // Call the getGradingScale() method
    System.out.println("Grade Scale: " + student.getGradingScale());
  }
  ...
}
```

Output:

```
---> Information <---
Full Name: Muh. Adnan Putra A.
Study Program: Information System
GPA: 4.0
Grade Scale: Cum Laude
```

C. Constructor

1. Apa itu *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 persis dengan sama classnya.

Pada dasarnya, setiap *class* pasti 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. Selain itu, sebuah class *dapat* memiliki lebih dari satu *constructor*. Dalam penggunaannya, *constructor* sering digunakan untuk menginisialisasi *instance variable* atau *attribute* dalam sebuah *class*. Berikut ini contoh sebuah *constructor* dari *class* Student:

```
public Student(String fullName, String studyProgram, double gpa) {
  this.fullName = fullName;
  this.studyProgram = studyProgram;
  this.gpa = gpa;
}
```

Berikut contoh penggunaannya saat membuat objek

```
public static void main(String[] args) {
    Student student = new Student("Muh. Adnan Putra A.", "Information System", 4.0);
}
```

Saat objek mahasiswa dibuat, maka nilai *attribute* fullName menjadi "Muh. Adnan Putra A.", *attribute* studyProgram menjadi "Information System" dan *attribute* gpa menjadi 4.0. Perhatikan bahwa saat menggunakan *constructor*, urutan parameter harus sesuai.

2. *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 tipe 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 Student.

```
public Student() {}

public Student(String fullName) {
    this.fullName = fullName;
}

public Student(String fullName, String studyProgram, double gpa) {
    this.fullName = fullName;
    this.studyProgram = studyProgram;
    this.gpa = gpa;
}
```

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

```
public static void main(String[] args) {
    Student student = new Student("Muh. Adnan Putra A.");
}
```

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