MODUL IV
CLASS DAN OBJECT

A. Tujuan

Pada modul Class dan Object ini, mahasiswa diharapkan untuk mampu:

1. Membuat class dan menggunakannya untuk membuat object.
2. Memanipulasi field dan method di dalam class.
3. Menggunakan constructor untuk memberi value kepada field.

B. Teori Dasar

1. Object Oriented Programming

Object Oriented Programming atau disingkat menjadi OOP, merupakan suatu paradigma pemrograman yang berfokus pada konsep class dan object. Paradigma ini populer digunakan dalam pembangunan program yang besar, kompleks, dan rutin diperbaharui. Kode yang ditulis menggunakan OOP akan menjadi terstruktur, mudah digunakan berulang kali, mudah diperbesar, dan diperbanyak, dan sangat efisien.

2. Class

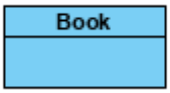
Class merupakan blueprint atau template untuk membuat object. Sebaga analogi, suatu penerbit menggunakan sebuah template setiap kali menyusun buku. Nantinya template ini akan digunakan saat penerbit akan mencetak beberapa buku. Karena menggunakan template yang sama, maka semua buku akan memiliki struktur yang serupa, contohnya terdiri dari cover, daftar isi, dan sebagainya. Template tadi adalah class, dan buku-buku yang dicetak penerbit, adalah object.

Class dibuat menggunakan keyword `class`, diikuti dengan nama class, kemudian kurung kurawal. Sebuah class ditulis di dalam sebuah file .java, dengan nama yang sama dengan nama file.

```
1  class Book {
2
3  }
```

Kode di baris ke-1 adalah deklarasi class bernama Book.

Sebuah class sederhana direpresentasikan menggunakan salah satu Unified Modeling Language, yakni class diagram seperti di gambar berikut.



Gambar 4.1 Class Book

3. Object

Object dibuat menggunakan Class. Syntax pembuatan object diawali dengan nama class yang digunakan, kemudian nama object, tanda sama dengan, keyword `new`, lalu nama Class diikuti tanda kurung `()`. Pembuatan object dilakukan di dalam main method.

```
1  public class BookApp {
2    public static void main(String[] args) {
3      Book myBook = new Book();
4    }
5  }
```

Kode di baris ke-3 adalah pembuatan object `myBook` dari class `Book`.

Object akan disimpan di suatu lokasi di dalam Heap memory, dan setiap object berada di lokasi yang berbeda-beda. Sedangkan variable disimpan di suatu lokasi di Stack memory.

4. Field

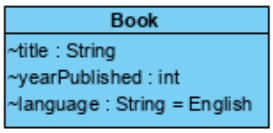
Object bisa menyimpan data menggunakan field. Data yang disimpan misalnya judul buku, tahun terbit, dan lain sebagainya. Field juga terkadang disebut sebagai property atau attribute. Field dideklarasikan di class terlebih dahulu sebelum bisa menyimpan data di object. Syntax deklarasi field serupa dengan variable.

Class

```
1  class Book {
2    String title;
3    int yearPublished;
4    final String language = "English";
5  }
```

Kode di baris ke-2 hingga ke-4 merupakan deklarasi field `title`, `yearPublished`, dan `language` di dalam class `Book`.

Class Book yang memiliki field di dalamnya diperlihatkan di class diagram berikut ini.



Gambar 4.2 Class Book dengan Field

Data disimpan ke dalam field dengan cara memberikan value kepada field melalui assignment, seperti halnya variable. Data dapat diakses dengan menuliskan nama object, kemudian tanda titik, dilanjutkan dengan nama field.

App

```
1  Book hisBook = new Book();
2  hisBook.title = "The Hunger Games";
3  hisBook.yearPublished = 2008;
4
5  System.out.println(hisBook.title);
6  System.out.println(hisBook.language);
```

Kode di baris ke-2 dan ke-3 adalah pemberian value kepada field `title` dan `yearPublished`. Sedangkan kode di baris ke-5 dan ke-6 dilakukan akses terhadap value kedua field. Object `hisBook` sekarang memiliki data judul buku dan tahun terbit melalui kedua field tadi.

Apabila field tidak dilakukan inisialisasi, maka Java akan memberikan default value seperti di Tabel berikut. Null value adalah lokasi yang kosong di memori. Jika program mengakses field yang berisi null value, maka program akan mengalami crash.

Tipe Data	**Default Value**
String	Null
int	0
double	0.0
boolean	false
Object	Null

5. Method

Object bisa berisi method untuk melakukan suatu operasi. Misalnya, sebuah object buku memiliki data tahun terbit. Suatu method bisa digunakan untuk menghitung dan menentukan apakah buku ini sudah tua atau belum. Deklarasi method dilakukan di dalam class dengan syntax yang serupa dengan method di modul sebelumnya. Field yang ada di dalam class dapat diakses di dalam method.

Book.java

```
1   class Book {
      ...
3     int yearPublished;
      ...
6
7     boolean isVintage(int currentYear) {
8       int age = currentYear - yearPublished;
9       return age >= 50;
10    }
    }
```

Kode di baris ke-6 hingga ke-10 merupakan deklarasi method `isVintage()` yang mengembalikan value bertipe boolean. Method ini akan menentukan apakah buku sudah tua atau belum. Method ini mengakses field `yearPublished`.

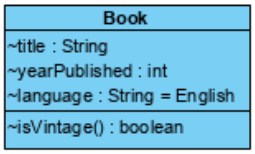
Method di dalam class dipanggil dengan cara yang berbeda dari method biasa, yakni dengan menuliskan nama object, tanda titik, diikuti dengan nama method.

BookApp.java

```
    ...
12  System.out.println(hisBook.isVintage(2020));
```

Kode di baris ke-12 merupakan pemanggilan method `isVintage()` dengan mengirimkan parameter 2020.

Class `Body` yang memiliki sejumlah field dan sebuah method digambarkan di class diagram seperti di bawah ini.



Gambar 4.3 Class Body dengan method

Suatu class diperbolehkan untuk memiliki beberapa method yang kembar asalkan masing-masing dilengkapi parameter yang berbeda, situasi ini disebut dengan method overloading.

Book.java

```
1   import java.time.YearMonth;
2
3   class Book {
    ...
8     boolean isVintage(int currentYear) {
9       int age = currentYear - yearPublished;
10      return age >= 50;
11    }
12
13    boolean isVintage() {
14      int currentYear = YearMonth.now().getYear();
```

```
15      int age = currentYear - yearPublished;
16      return age >= 50;
17    }
18  }
```

Kode di baris ke-8 hingga ke-10 adalah deklarasi method `isVintage()` dengan parameter `currentYear`, sedangkan kode di baris ke-13 hingga ke-17 adalah deklarasi method dengan nama yang sama namun tanpa parameter. Kode di baris ke-14 akan menghitung tahun saat ini.

BookApp,java

```
    ...
13  System.out.println(hisBook.isVintage());
```

Kode di baris ke-13 akan memanggil method `isVintage()` yang tak membutuhkan parameter.

6. Constructor

Diantara method di dalam class, ada satu method yang akan dieksekusi pertama kali dan hanya sekali, secara otomatis, ketika object dibuat. Method ini disebut sebagai constructor. Syntax pembuatan constructor menggunakan nama class, diikuti dengan tanda kurung, di dalamnya bisa berisi parameter. Constructor umumnya digunakan untuk melakukan inisialisasi field.

Java memiliki tiga tipe constructor, yakni no argument constructor, parameterized constructor, dan default constructor.

No argument constructor adalah constructor yang tidak membutuhkan parameter.

Magazine.java

```
1  public class Magazine {
2    String title;
3    int issue;
4
5    Magazine() {
6      System.out.println("Magazine Class Created");
7    }
8  }
9
```

Kode di baris ke-5 hingga ke-7 adalah no argument constructor di class `Magazine`.

Parameterized constructor adalah constructor yang membutuhkan parameter. Parameter ini dapat digunakan untuk mengirimkan value kepada field. Jika nama parameter di dalam constructor sama dengan nama field, maka gunakan keyword `this` kepada field untuk membedakan keduanya.

Magazine.java

```
1  public class Magazine {
2    String title;
3    int issue;
4
5    Magazine(String title) {
6      this.title = title;
7    }
8  }
9
```

Kode di baris ke-5 hingga ke-7 adalah parameterized constructor di class `Magazine`. Constructor ini membutuhkan parameter `title`.

Keyword `this` digunakan agar tidak terjadi **variable shadowing**, yakni ketika terdapat dua variable dengan nama yang sama namun terletak di scope yang berbeda. Jika hal ini terjadi, maka variable yang berada di scope terluar, tidak bisa diakses.

Magazine.java

```
1  public class Magazine {
2    String title;
3    int issue;
4
5    Magazine(String title, int edition) {
6      this.title = title;
7      issue = edition;
8    }
9  }
```

Kode di baris ke-5 merupakan deklarasi constructor untuk class `Magazine`. Constructor ini menerima parameter `title` dan `edition`. Value kedua parameter akan diberikan kepada field `title` dan `issue`.

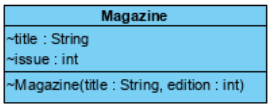
Default constructor adalah class yang tidak memiliki constructor. Jika suatu class tidak dilengkapi constructor maka, Java akan membuatkan sebuah no argument constructor secara otomatis. Constructor yang dibuat oleh Java ini, merupakan default constructor.

Constructor dipanggil ketika object dibuat. Jika constructor memiliki parameter, maka ketika object dibuat, parameter ini wajib diberikan ke dalam tanda kurung.

MagazineApp.java

```
1  public class MagazineApp {
2      public static void main(String[] args) {
3          Magazine myMagazine = new Magazine("Trubus", 1);
4          System.out.println(myMagazine.title);
5      }
6  }
```

Kode di baris ke-3 terdapat pemberian parameter kepada constructor untuk object `myMagazine`.



Gambar 4.4 Class Magazine dengan Constructor

Seperti halnya method bisa terdapat beberapa method dengan nama yang sama, constructor juga bisa melakukannya. Suatu class bisa memiliki beberapa constructor namun parameternya harus berbeda, ini disebut dengan **constructor overloading**.

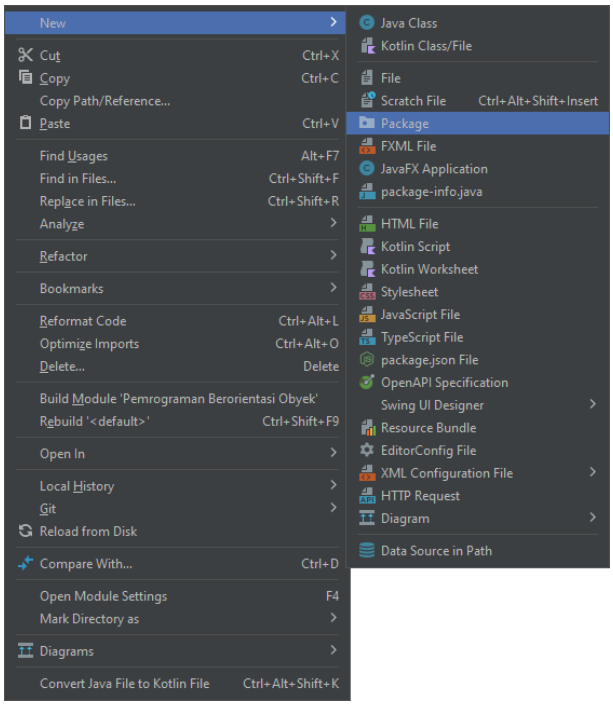
Magazine.java

```
1  public class Magazine {
     ...
5    Magazine() {
6      title = "Untitled";
7      issue = 0;
8    }
9
10   Magazine(String title, int edition) {
11     this.title = title;
12     issue = edition;
13   }
14 }
```

Kode di baris ke-5 hingga 8 merupakan deklarasi constructor, seperti halnya kode di baris ke-10 hingga ke-13 yang juga merupakan deklarasi constructor.

7. Package

File .java di dalam project bisa disimpan ke dalam sebuah folder atau umumnya disebut package. Package digunakan agar file lebih tersusun rapi sehingga mudah untuk dikelola. Pembuatan package melalui IntelliJ IDEA dilakukan dengan klik kanan pada folder **src**, pilih **New**, lalu klik **Package** di Project Tool Window.



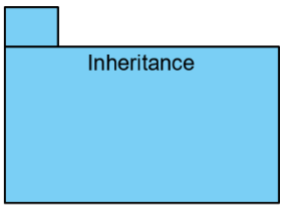
Kemudian ketik nama package dan tekan tombol **Enter** untuk menampilkan package. Sekarang file .java bisa dibuat di dalam package ini.

Shape.java

```
1  package classobject;
2
3  class Book {
4
5  }
```

Kode di baris ke-1 menunjukkan bahwa class `Book` berada di dalam package `inheritance`.

Package bisa digambarkan ke dalam Class Diagram seperti di bawah ini.



Gambar 4.5 Package di Class Diagram