MODUL III
METHOD

A. Tujuan

Pada modul Method ini, mahasiswa diharapkan untuk mampu:

1. Menggunakan method di Java.
2. Mengirimkan value ke method menggunakan parameter dan mengembalikan value dari method.
3. Menggunakan method overloading.
4. Memahami scope untuk setiap variable di Java.
5. Menggunakan comment untuk mendokumentasikan kode Java.

B. Teori Dasar

1. Method

Method adalah block yang berisi sejumlah statement yang akan dieksekusi ketika dipanggil. Pada bahasa pemrograman lain, method disebut sebagai function. Method digunakan agar kita bisa menuliskan suatu kode satu kali, dan menggunakannya berkali-kali.

Syntax pembuatan atau deklarasi method yang dasar, ditulis dengan keyword `static void`, kemudian nama method, lalu tanda kurung (), dan dilanjutkan dengan block yang berisi sejumlah statement yang akan dieksekusi. Method dipanggil dengan menuliskan nama method, diikuti dengan tanda kurung ().

```
1  static void greeting() {
2    System.out.println("Good Morning!");
3  }
4
5  public static void main(String[] args) {
6    greeting();
7  }
```

Output

```
Good Morning
```

Kode di baris ke-1 hingga 3 adalah deklarasi method bernama `greeting()`. Kode di baris ke-6 adalah pemanggilan method tersebut.

2. Parameter

Saat memanggil method, kita bisa mengirimkan value untuk diproses di dalam block deklarasi method. Value ini dikirim melalui parameter, yakni variable di dalam tanda kurung. Parameter ditentukan saat deklarasi method dengan syntax: tipe data dan nama parameter. Jika parameter lebih dari satu, maka masing-masing dipisahkan dengan tanda koma.

```
1  static void greetingWithName(String name) {
2    System.out.println("Hi, " + name);
3  }
4
5  public static void main(String[] args) {
6    greetingWithName("John");
7  }
```

Output

```
Hi, John
```

Kode di baris ke-1 terdapat deklarasi parameter `name` bertipe String. Parameter ini digunakan di dalam block method. Value untuk parameter dikirim oleh pemanggilan method `greetingName()` di baris ke-6.

3. Return

Suatu method bisa menghasilkan atau mengembalikan suatu value. Method yang mengembalikan value, saat dideklarasikan harus menggunakan keyword `static` lalu diikuti dengan tipe data dari value yang akan dikembalikan. Kemudian statement terakhir di dalam block, value dikembalikan menggunakan keyword `return`.

```
1  static int countAge(int birthYear) {
2    int age = 2022 - birthYear;
3    return age;
4  }
5
6  System.out.println(countAge(1997))
```

Output

```
25
```

Kode di baris ke-1 menunjukkan deklarasi method `countAge()` yang akan mengembalikan value bertipe int. Value yang dikembalikan berada di dalam variable `age` di kode baris ke-3. Method ini dipanggil di baris ke-6.

4. Method Overloading

Jika terdapat beberapa method dengan nama yang sama, maka akan memunculkan error kecuali apabila parameternya berbeda. **Sejumlah method dengan nama yang sama namun parameter berbeda, disebut dengan method overloading**. Block statement di method overloading bisa memiliki kode yang berbeda.

```
1   static void courseResult(int grade) {
2     System.out.println("Your grade is " + grade);
3   }
4
5   static void courseResult(char grade) {
6     System.out.println("You get " + grade);
7   }
8
9   public static void main(String[] args) {
10    courseResult(90);
11    courseResult('B');
12  }
```

Output

```
Your grade is 90
You get B
```

Kode di baris ke-1 hingga 3 adalah deklarasi method `courseResult()`, serta di baris ke-5 hingga 7 juga deklarasi method yang sama. Ini merupakan method overloading. Kode di baris ke-10 merupakan pemanggilan method `courseResult()` dengan parameter bertipe int, sedangkan di baris ke-11 merupakan pemanggilan method `courseResult()` dengan parameter bertipe char.

5. Scope

Variable hanya bisa diakses di wilayahnya sendiri. Hal ini disebut dengan scope. Ada dua tipe scope, yakni method scope dan block scope.

Method scope adalah wilayah dimana variable yang dideklarasikan di dalam method, bisa diakses dimana saja setelah dideklarasikan.

```
1  public static void main(String[] args) {
2    // System.out.println(x); // Error
3    int x = 1;
4    System.out.println(x);
5  }
```

Output

```
1
```

Kode di baris ke-2 gagal mengakses variable `x`, karena variable ini belum dideklarasikan. Kode di baris ke-4 berhasil mengakses, karena variable `x` telah dideklarasikan sebelumnya.

Block scope merupakan wilayah dimana variable diakses di block yang sama, di dalam tanda kurung kurawal `{}`. Block scope juga berlaku ketika di dalam if, while, dan for statement.

```
1  {
2    int y = 2;
3    System.out.println(y);
4  }
5  // System.out.println(y);
```

```
Output
2
```

Kode di baris ke-5 ingin mengakses variable `y` yang dideklarasikan di baris ke-2. Namun gagal, karena berada di scope yang berbeda.

6. Comment

Comment atau komentar adalah kode program yang diabaikan oleh compiler sehingga tidak akan dieksekusi. Komentar digunakan untuk memberi informasi mengenai kode lainnya. Java memiliki dua tipe komentar, yakni single line comment dan multi line comment. Single line comment menggunakan two forward slashes `//` dan hanya berlaku untuk satu baris saja. Sedangkan multi line comment menggunakan awalan `/*` dan akhiran `*/` dan berlaku untuk beberapa baris.

```
1  /*
2  * Get an area of a rectangle
3  *
4  * @param side length
5  * @return area
6  */
7  static int rectangleArea(int sideLength) {
8    return sideLength ^ 2 ; // side length times side length
9  }
```

Kode di baris ke-1 hingga ke-6 merupakan multi line comment, dan kode di baris ke-8 terdapat single line comment.