MODUL II
CONTROL FLOW

A. Tujuan

Pada modul Control Flow ini, mahasiswa diharapkan untuk mampu:

1. Mengetahui perbedaan expression dan statement.
2. Menggunakan eksekusi percabangan melalui if dan switch statement.
3. Melalukan eksekusi perulangan menggunakan for, while, do-while, dan for-each statement.
4. Mengenal array dan mengaksesnya menggunakan perulangan.

B. Teori Dasar

1. Expression

Expression adalah sekumpulan kode yang terdiri dari variable, operator, dan pemanggilan method yang disusun dengan Java syntax untuk dievaluasi menjadi suatu value.

```
1  int price;
2  price = 1000;
```

2. Statement

Statement merupakan susunan kode yang membentuk perintah eksekusi yang lengkap. Statement dibentuk oleh sejumlah expression yang disebut sebagai **expression statement**, yakni: assignment statement, increment statement, method invocation statement, dan object creation statement.

```
1  int age = 17;
2  age++;
3  System.out.println(age);
4  Book harryPotter = new Book();
```

Kode di baris-1 adalah assignment statement karena menggunakan operator assignment, yakni tanda sama dengan. Sedangkan kode di baris ke-2 adalah increment statement karena menggunakan operator increment `++`, dan kode di baris ke-3 adalah method invocation statement karena terdapat pemanggilan method, yakni `println()`. Kode di baris ke-4 adalah object creation statement, yang menunjukkan perintah pembuatan sebuah object `harryPotter`.

3. If-then Statement

Suatu statement bisa diatur agar dieksekusi hanya ketika suatu kondisi dipenuhi Kondisi dipenuhi atau tidak, ditentukan melalui evaluation. **Evaluation** adalah pemeriksaan value bertipe boolean dari suatu expression, misalnya `3 != 4` ialah true. Expression tersebut diletakkan di dalam kurung `()`. Kondisi dipenuhi jika value hasil evaluation adalah true.

Hal ini bisa dilakukan dengan **if-then statement** menggunakan keyword `if`. Kode yang akan dieksekusi diletakkan di dalam **block**, yakni diantara kurung kurawal pembuka `{` dan penutup `}`.

```
1  int grade = 40;
2  if (grade <= 40) {
3    System.out.println("Poor");
4  }
```

Output

```
Poor
```

Kode di baris ke-2 merupakan evaluation terhadap expression `grade <= 40`. Karena bernilai `true`, maka kode di dalam block, yakni baris ke-3, akan dieksekusi.

4. If-then-else Statement

Jika kondisi bernilai `false` sehingga harus ada block lain yang dieksekusi, maka bisa dilakukan dengan **if-then-else statement** menggunakan keyword `if` dan `else`. If-then-else statement akan mengeksekusi block lain jika kondisi tidak dipenuhi.

```
1  int grade = 40;
2  if (grade > 40) {
3    System.out.println("Good");
4  } else {
5    System.out.println("Poor");
6  }
```

Output

```
Poor
```

Kode di baris ke-2 memiliki expression `grade > 40` yang bernilai `false`, sehingga kode di baris ke-3 tidak dieksekusi, maka kode di baris ke-5 yang dieksekusi.

Jika ada sejumlah kondisi untuk sejumlah block yang akan dieksekusi, maka bisa dilakukan dengan keyword `else if`. Block yang dieksekusi adalah block yang memiliki kondisi yang dipenuhi saat evaluation terjadi.

```
1  int grade = 75;
2  if (grade > 70) {
3    System.out.println("Well done");
4  } else if (grade > 50) {
5    System.out.println("Good");
6  } else {
7    System.out.println("Poor");
8  }
```

Output

```
Well done
```

Evaluation terhadap expression di baris ke-2 bernilai `true` karena variable `grade` memiliki value `75` sehingga block di baris ke-3 dieksekusi dan block lainnya tidak.

5. Ternary If

Bentuk sederhana dari if statement adalah ternary if. Bentuk ini tidak memiliki block untuk dieksekusi, namun hanya ada value. **Ternary if** menggunakan tanda tanya `?` dan titik dua `:`.Kondisi yang dilakukan evaluation berada di dalam kurung `()`, lalu diikuti dengan tanda tanya, dilanjutnya dengan value jika evaluation bernilai `true` kemudian titik dua diikuti dengan value jika evaluation bernilai `false`. Value ini akan diberikan ke variable yang diletakkan di awal if statement.

```
1  int time = 20;
2  String result = (time < 18) ? "Good day" : "Good evening";
3  System.out.println(result);
```

Output

```
Good evening
```

Variable result akan memiliki value `Good evening`, karena value dari variable `time` lebih dari `18`.

6. Switch Statement

Jika expression yang dilakukan evaluation memiliki value yang tetap, maka bisa menggunakan switch statement dengan keyword `switch`. Evaluation dilakukan menggunakan keyword `case`. Statement yang akan dieksekusi tidak berbentuk block yang diawali dan diakhiri kurung kurawal `{}`, namun diawali titik dua `:` dan diakhiri perintah `break`.

Apabila semua value tidak ada yang cocok saat evaluation, maka statement yang akan dieksekusi diletakkan setelah keyword `default`.

```
1  char letterGrade = 'C';
2  switch (letterGrade) {
3    case 'A':
4      System.out.println("Wonderful");
5      break;
6    case 'B':
7      System.out.println("Nice");
8      break;
9    case 'C':
10   case 'D':
11     System.out.println("Too bad");
12     break;
13   default:
14     System.out.println("Invalid");
15 }
```

Output

```
Good
```

Kode di baris ke-11 akan dieksekusi karena variable `letterGrade` memiliki value karakter C.

7. For Statement

Sejumlah statement yang ingin dieksekusi berulang kali bisa dilakukan menggunakan loop atau perulangan. Salah satu perulangan yang terdapat di Java bisa menggunakan for statement. **For statement** akan mengeksekusi statement berulang kali berdasarkan suatu rentang angka, misalnya 1 sampai 5.

For statement menggunakan keyword `for` dan diikuti dengan tiga expression di dalam kurung, yaitu:

a) Initialization expression, yang dieksekusi satu kali untuk memulai perulangan.

b) Termination expression, yang akan dilakukan evaluation setiap perulangan. Jika evaluation bernilai false, maka perulangan akan dihentikan.

c) Increment expression, yang dieksekusi di akhir perulangan. Umumnya dilakukan increment atau decrement expression.

Ketiga expression dipisah dengan titik koma. Setelah kurung yang berisi ketiga expression, dilanjutkan dengan block yang berisi statement yang akan dieksekusi berulang kali.

```
1  for (int i = 1; i <= 5; i++) {
2    System.out.println("Jump");
3  }
```

Output

```
Jump
Jump
Jump
Jump
Jump
```

Kode di baris ke-1 terdapat initialization expression `int i = 1` lalu titik koma dan termination expression `i<=5` dan titik koma serta increment expression `i++`.

Perulangan diawali dengan initialization variable `i` value 1 lalu evaluation terhadap `i<=5`. Karena bernilai `true`, maka block kode di baris ke-2 akan dieksekusi, lalu increment expression `i++` dieksekusi sehingga value `i` menjadi 2. Kemudian dilanjutkan dengan perulangan kedua, yakni evaluation `i<=5`, karena masih bernilai `true`, maka kode di dalam block dieksekusi, dan `i++` dieksekusi sehingga `i` menjadi 3. Perulangan dilakukan lagi dan

ketika expression `i<=5` bernilai `false`, perulangan berhenti, yakni ketika variable `i` memiliki value 6.

8. While Statement

Perulangan untuk eksekusi statement berulang kali juga bisa dilakukan dengan while statement. **While statement** akan melakukan eksekusi berulang kali selama suatu kondisi pada termination expression bernilai true.

Berbeda dengan for statement, kurung di while statement hanya berisi sebuah termination expression yang akan dilakukan evaluation sebelum block dieksekusi. Selama expression ini memiliki value true, maka perulangan terus dilakukan.

```
1  int counter = 1;
2  while (counter <= 5) {
3    System.out.println("Run");
4    counter++;
5  }
```

Output

```
Run
Run
Run
Run
Run
```

Kode di baris ke-2 terdapat expression `counter<=5` dengan value `true` karena variable `counter` memiliki value `1`. Sehingga block kode di baris ke-3 dan 4 dieksekusi. Kode di baris ke-4 akan menambah value variable `counter` dengan angka 1, sehingga value-nya menjadi 2. Selanjutnya, expression `counter<=5` dilakukan evaluation dan tetap memiliki value `true`. Sehingga block kode baris ke 3 dan 4 dieksekusi, dan seterusnya hingga expression ini memiliki value `false` saat value variable `counter` menjadi 6.

9. Do-while Statement

Do while statement juga akan melakukan eksekusi berulang kali selama termination expression memiliki value true. Namun berbeda dengan while statement dimana termination expression dilakukan evaluation sebelum eksekusi, pada do-while statement, termination expression akan dilakukan evaluation setelah eksekusi dilakukan terlebih dahulu.

```
1  int anotherCounter = 1;
2  do {
3    System.out.println("Loop " + anotherCounter);
4    anotherCounter++;
5  } while (anotherCounter <= 5);
```

Output

```
Loop 1
Loop 2
Loop 3
Loop 4
Loop 5
```

Kode di baris ke-1 akan memberikan value 1 ke variable `anotherCounter`. Kemudian perulangan pertama dimulai, yakni eksekusi block statement di baris ke-3 yang menampilkan teks Loop dan value dari variable `anotherCounter`. Kemudian di baris ke-4 juga dieksekusi dengan melakukan increment variable `anotherCounter` sehingga value-nya menjadi 2. Selanjutnya dilakukan evaluation terhadap expression `anotherCounter<=5` yang menghasilkan value `true`. Perulangan kedua dan seterusnya dilakukan hingga saat evaluation, expression `anotherCounter<=5` memiliki value `false` sehingga perulangan berhenti.

10. Break dan Continue Statement

Perulangan bisa dihentikan dengan break statement. Ketika perulangan berhenti, maka statement berikutnya yang dieksekusi adalah statement di luar dan setelah perulangan.

```
1  int counter = 1;
2  while (counter <= 5) {
3    System.out.println("While with Break " + counter + " loop");
4    counter++;
5    if (counter == 3) {
6      break;
7    }
8  }
```

Output

```
While with Break 1 loop
While with Break 2 loop
```

Kode di baris ke-5 akan melakukan evaluation jika variable `counter` memiliki value 3 saat perulangan dilakukan, selanjutnya perulangan akan dihentikan.

Selain dihentikan, perulangan juga bisa dilangkahi satu atau beberapa kali. Continue statement tidak akan menghentikan loop seperti break statement, namun akan melangkahi satu loop saja.

```
1  for (int i = 1; i <= 5; i++) {
2    if (i == 2) {
3      continue;
4    }
5    System.out.println("For with Continue " + i + " loop");
6  }
```

Output

```
For with Continue 1 loop
For with Continue 3 loop
For with Continue 4 loop
For with Continue 5 loop
```

Ketika variable `i` memiliki value 2, maka perulangan saat ini akan dilangkahi, sehingga kode di baris ke-5 tidak dieksekusi. Perulangan berikutnya tetap dilakukan hingga perulangan berhenti saat variable `i` memiliki value 5.

11. Array

Array merupakan variable yang diberi sekumpulan value dengan tipe data yang sama. Initialization array dilakukan dengan menuliskan tipe data, kurung siku `[]`, kemudian nama variable, tanda sama dengan `=`, kurung kurawal pembuka `{`, lalu value di dalam array yang dipisah dengan tanda koma, kemudian kurung kurawal penutup `}`.

```
1  String[] names = {"John", "Jane", "George"};
2  int[] ages = {17, 18, 22};
```

Setiap value di dalam array memiliki nomor urut atau index yang diawali dari angka nol untuk value paling kiri. Akses setiap value dilakukan dengan menuliskan nama array kemudian dilanjutkan dengan kurung siku, yang di dalamnya dituliskan index dari value yang akan diakses.

```
1  int[] ages = {17, 18, 22};
2  System.out.println(ages[1]);
```

Output

```
18
```

Kode di baris ke-2 akan mengakses value di array `ages` dengan index `1`, yakni 18.

Banyaknya value di dalam array bisa diketahui menggunakan perintah `length`.

```
1  int[] ages = {17, 18, 22};
2  System.out.println(ages.length);
```

```
Output
3
```

Jika ingin mengakses seluruh value di dalam array, dilakukan menggunakan perulangan, seperti for statement. Jika menggunakan for statement, maka termination expression di dalamnya akan membutuhkan informasi banyaknya value di dalam array, maka perintah `length` akan digunakan. Block statement akan berisi kode untuk mengakses setiap value di dalam array, namun kurung siku diisi dengan variable `i`.

```
1 String[] names = {"John", "Jane", "George"};
2 for (int i = 0; i < names.length; i++) {
3   System.out.println(names[i]);
4 }
```

```
Output
John
Jane
George
```

Bentuk lain untuk mengakses seluruh value di dalam array adalah menggunakan for-each statement. **For-each statement** akan melakukan perulangan dengan expression di dalam kurung, disitu dituliskan nama array, tapi diawali dengan initialization sebuah variable. Variable ini akan digunakan untuk mengakses setiap value di dalam array. Sehingga, block statement tidak membutuhkan akses ke setiap array menggunakan kurung siku, namun digantikan dengan variable ini.

```
1 int[] stocks = {100, 75, 66};
2 for (int stock: stocks) {
3   System.out.println(stock);
4 }
```

```
Output
100
75
66
```