MODUL 4. CONTROL FLOW: IF DAN IF-ELSE

4.1 Tujuan

4.2 Alat dan Bahan

Seperti biasa, siapkan Visual Studio Code kalian!

4.3 Dasar Teori

Pernahkah kalian memikirkan sebuah skenario dengan beberapa kondisi yang berbeda? Dan dengan kondisi yang berbeda itu, maka yang harus dilakukan tentunya juga harus berbeda. Nah, saat kalian berpikir seperti itu, kalian sebenarnya sedang memikirkan algoritma juga, lho! Dalam pemrograman, kita bisa melakukan hal yang berbeda-beda tergantung dengan kondisi yang didapatkan. Beda skenario, beda aksi!

Hal ini disebut juga dengan percabangan. Kalian pasti bisa menebak kenapa dinamai “percabangan”. Tentu saja dikarenakan kondisi-kondisi atau skenario-skenario yang ada kita bisa anggap sebagai “cabang” yang dapat menjalar ke arah yang berbeda-beda! Kalau begitu, mari kita lihat sebuah skenario ini.

“Bila cuaca sedang cerah dan Dira sedang sehat, maka Dira pergi ke taman.”

Nah, dengan skenario di atas, kita bisa menyimpulkan beberapa kondisi dan aksi yang berbeda.

a. Bila kondisi sedang hujan, maka Dira tidak akan ke taman, meskipun Dira sedang sehat.
b. Bila kondisi sedang sakit, maka Dira tidak akan ke taman, meskipun hari sedang cerah.
c. Bila kedua kondisi terpenuhi, maka Dira akan pergi ke taman.
d. Bila kedua kondisi tidak terpenuhi, sudah pasti Dira tidak akan pergi ke taman.

Beginilah kira-kira kondisi yang dapat terjadi, dan juga apa yang akan dilakukan Dira untuk setiap kondisi. Kita telah merangkai algoritmanya. Apakah kalian familiar dengan skenario yang mirip seperti di atas? Soalnya, hal ini ada hubungannya dengan mata kuliah Matematika Informatika. Jadi, perhatikan MATIF juga, ya!

4.3.1 If Statement

Sekarang, kita akan masuk ke bagian menyenangkannya: penulisan kode! Kita akan mengimplementasikan algoritma yang telah kita tuliskan tadi ke dalam bentuk kode. Sebelum itu, mari kita mengenal terlebih dahulu struktur perintah “if-else” pada bahasa Java.

```
if (kondisi) {
   // lakukan sesuatu
} else if (kondisi2) {
   // lakukan sesuatu yang lain
} else {
   // lakukan sesuatu yang lain lagi
}
```

Di sini ada beberapa jenis *keyword* yang bisa kita lihat, yaitu: if, else if, dan else. Kalian mungkin bisa menebak fungsi dari perintah if-else di atas, karena bila kita terjemahkan ke Bahasa Indonesia, arti dari ketiga *keyword* tersebut kurang lebih seperti ini: “Jika”, “Namun jika”, “Selain itu”. Jadi, perintah if-else di atas ini digunakan untuk menjalankan kode tertentu, tergantung kepada kondisi tertentu. Yang artinya, untuk kasus Dira di atas, kita bisa menulis secara kasar seperti ini: “IF cuaca == cerah AND dira == sehat : pergi ke taman ELSE tidak pergi ke taman.” Mudah, bukan?

Ia akan menjalankan perintah dalam bloknya apabila nilai yang diberikan di dalam kurungnya itu bernilai “true”. Bila tidak, ia akan lanjut ke kondisi berikutnya, yaitu else if. Namun, bila else if tidak ada, atau semua kondisi else if juga salah, maka ia akan lari ke pelarian terakhir, yaitu else. Apabila tidak ada else, dan semua kondisi salah, maka ia tidak akan melakukan apa-apa.

Mari kita implementasikan logika yang telah kita rangkai di atas pada program Java!

```
public class ProgramDira {
   public static void main(String[] args) {
       boolean diraSehat = true;
       boolean hariCerah = true;

       if (diraSehat == true && hariCerah == true) {
           System.out.println("Dira pergi ke taman");
       } else {
           System.out.println("Dira tidak pergi ke taman");
       }
   }
}
```

Kode di atas adalah implementasi simulasi dari logika yang telah kita rangkai. Kita menggunakan tipe data boolean, operator perbandingan, dan operator logika untuk memproses kondisi-kondisi yang ada. Namun kode di atas hanyalah simulasi sederhana, sehingga nilai variabel "`diraSehat`" dan "`hariCerah`" akan selalu true. Karena itu, kita akan mengambil *input* atau masukkan dari kita, pengguna ke dalam program dengan menggunakan *Scanner*.

4.3.2 Scanner

Scanner adalah sebuah kelas dalam Java. Ia seperti String, merupakan tipe data objek bila kita membuat sebuah *instance* baru dari *Scanner* ini. Untuk memakai *Scanner*, kita harus menggunakan perintah bernama *import* terlebih dahulu untuk menggunakannya di dalam program kita. Melakukan impor berarti mengambil potongan kode yang sudah tersedia dari luar untuk kita gunakan. Panggil lah *import* di bagian paling atas kode seperti berikut!

```
import java.util.Scanner;

public class ProgramDira {
   public static void main(String[] args) {
       boolean diraSehat = true;
       boolean hariCerah = true;

       if (diraSehat == true && hariCerah == true) {
           System.out.println("Dira pergi ke taman");
       } else {
           System.out.println("Dira tidak pergi ke taman");
```

```
        }
    }
}
```

Selanjutnya, kita akan membuat objek dari kelas *Scanner ini*

```
public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        boolean diraSehat = true;
        boolean hariCerah = true;
```

Dengan begini, kita telah membuat sebuah objek *Scanner* baru, yang dapat kita pakai. Untuk mendapatkan input dari user, kita bisa menggunakan beberapa *method* dari *Scanner*. Di sini, kita akan memasukkan kata “ya” atau “tidak” saja, jadi kita menggunakan method bernama “.next()” yang dapat mengambil sepotong kata dari pengguna, yang nantinya akan menjadi *String*. Kalau ingin memasukkan kalimat yang memiliki spasi, harus menggunakan method bernama “.nextLine()” yang akan mengambil input sebaris, bukan hanya sepotong kata.

```
        Scanner s = new Scanner(System.in);

        boolean diraSehat;
        boolean hariCerah;

        System.out.print("Apakah Dira sehat? (ya/tidak): ");
        String jawabanSehat = s.next();
```

Kemudian, kita akan cek jawaban apa yang diberikan oleh user, dan apakah sesuai dengan “ya” atau “tidak”. Bila jawaban adalah “ya”, maka kita akan melakukan *assignment* terhadap variabel-variabel yang sudah kita deklarasikan. Bila jawaban dari user bukan merupakan di antara keduanya, maka hentikan program dan cetak “Jawaban tidak valid”.

```
        if (jawabanSehat.equals("ya")) {
            diraSehat = true;
        } else if (jawabanSehat.equals("tidak")) {
            diraSehat = false;
        } else {
            System.out.println("Jawaban tidak valid");
            return;
        }
```

Kita menggunakan salah satu *method* yang dimiliki oleh String, yaitu ".equals" di mana ia akan membandingkan apa satu string sama dengan string lainnya, dan mengembalikan boolean true bila sama, dan false bila tidak. Kita membandingkan apakah input user adalah "ya" atau "tidak" dengan method tersebut.

Selanjutnya, lakukan hal yang sama untuk hariCerah.

```
        System.out.print("Apakah hari cerah? (ya/tidak): ");
        String jawabanCerah = s.next();

        if (jawabanCerah.equals("ya")) {
            hariCerah = true;
        } else if (jawabanCerah.equals("tidak")) {
            hariCerah = false;
        } else {
            System.out.println("Jawaban tidak valid");
            return;
        }
```

Sekarang, kita sudah selesai melakukan *assignment* terhadap kedua variabel yang kita deklarasikan. Jadi, mari proses variabel-variabel tersebut dengan kode sebelumnya!

```
if (diraSehat == true && hariCerah == true) {
   System.out.println("Dira pergi ke taman");
} else {
   System.out.println("Dira tidak pergi ke taman");
}
```

Dan step terakhir selesai! Kode lengkapnya adalah sebagai berikut:

```
import java.util.Scanner;

public class ProgramDira {
   public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        boolean diraSehat;
        boolean hariCerah;

        System.out.print("Apakah Dira sehat? (ya/tidak): ");
```

```
        String jawabanSehat = s.next();

        if (jawabanSehat.equals("ya")) {
            diraSehat = true;
        } else if (jawabanSehat.equals("tidak")) {
            diraSehat = false;
        } else {
            System.out.println("Jawaban tidak valid");
            return;
        }

        System.out.print("Apakah hari cerah? (ya/tidak): ");
        String jawabanCerah = s.next();

        if (jawabanCerah.equals("ya")) {
            hariCerah = true;
        } else if (jawabanCerah.equals("tidak")) {
            hariCerah = false;
        } else {
            System.out.println("Jawaban tidak valid");
            return;
        }

        if (diraSehat == true && hariCerah == true) {
            System.out.println("Dira pergi ke taman");
        } else {
            System.out.println("Dira tidak pergi ke taman");
        }
    }
}
```

Kemudian, bila tidak ada error, maka program akan berjalan dengan sempurna!

```
>> "Linux Mint" - rahman @ furude-PC in ~/Projects/Java/Permulaan
 -> javac ProgramDira.java

>> "Linux Mint" - rahman @ furude-PC in ~/Projects/Java/Permulaan
 -> java ProgramDira
Apakah Dira sehat? (ya/tidak): ya
Apakah hari cerah? (ya/tidak): ya
Dira pergi ke taman

>> "Linux Mint" - rahman @ furude-PC in ~/Projects/Java/Permulaan
 -> java ProgramDira
Apakah Dira sehat? (ya/tidak): tidak
Apakah hari cerah? (ya/tidak): ya
Dira tidak pergi ke taman
```

4.3.2.1 Mengatasi input yang tidak muncul

Pada beberapa kasus, input bisa saja tidak muncul dan program akan langsung berhenti, lho! Berbahaya sekali. Salah satu contoh kasusnya adalah kode berikut:

```
int menu = input.nextInt();

if (menu == 1) {
   String nama = input.nextLine();
   System.out.println("Nama: " + nama);
}
```

Di sini, input.nextLine() pada variabel nama tidak akan bisa kita masukkan apa-apa, dan program akan langsung selesai meskipun kita memasukkan “1” pada menu. Hal ini dikarenakan ketika kita memasukkan input pertama pada menu, saat kita menekan enter, enter yang kita tekan akan dimasukkan ke input pada nama! Jadi, nama sudah dianggap punya input, yaitu enter kita sebelumnya. Wah, bagaimana nih cara memperbaikinya?

Cara memperbaikinya cukup mudah! Tinggal tambahkan saja sebuah input nextLine baru setelah input pertama. Seperti ini.

```
int menu = input.nextInt();
input.nextLine();

if (menu == 1) {
   String nama = input.nextLine();
   System.out.println("Nama: " + nama);
}
```

Dan nantinya, saat kita menekan enter, enter kita akan masuk ke input di bawah variabel menu!

4.3.3 Nested If

Sekarang, ada yang dinamakan *nested if*. Kalau kita terjemahkan, *nested* ini secara literal artinya "bersarang" sehingga ada if lain yang bersarang di dalam blok if! Berikut adalah contohnya.

```
int number = 15;

if (number > 0) {
   System.out.println("Bilangan positif.");

   if (number % 2 == 0) {
       System.out.println("Bilangan genap.");
   } else {
       System.out.println("Bilangan ganjil.");
   }
} else if (number < 0) {
   System.out.println("Bilangan negatif.");
} else {
   System.out.println("Bilangan nol.");
}
```

Pada kode di atas kita melakukan cek apakah suatu bilangan positif atau negatif, kemudian bila positif, kita menjalankan perintah if lagi di dalamnya, yaitu untuk melakukan cek apakah bilangan tersebut ganjil atau genap dengan operator modulus yang sudah kita pelajari sebelumnya. Dan ia hanya akan dijalankan kalau bilangan yang dicek itu positif atau lebih besar dari nol. Mudah, kan?

Dan begitulah cara menggunakan perintah if-else untuk membuat percabangan. Mengeksekusi perintah-perintah tertentu tergantung dari kondisi yang diterima. Sampai bertemu di bab selanjutnya!

MODUL 5. CONTROL FLOW: SWITCH-CASE

5.1 Tujuan

5.2 Alat dan Bahan

5.3 Dasar Teori

Kita sudah mempelajari tentang percabangan memakai perintah if-else sebelumnya. Sekarang, kita akan berkenalan dengan perintah percabangan yang lain, yaitu switch-case!

5.3.1 Switch vs If

Lalu, apa yang membedakan switch-case ini dengan if-else? Yang membedakan, adalah pada nilai yang ia terima. Kita tahu kalau if-else ini harus menerima boolean, bukan? Bila true, maka blok kode akan dijalankan. Bila tidak, maka akan di-skip. Namun switch-case ini bisa menerima tipe data apapun, lho. Tapi, tentu cara penggunaannya berbeda. Mari kita lihat strukturnya!

```
switch(kondisi) {
   case nilai_kondisi1:
       // aksi
       break;
   case nilai_kondisi2:
       // aksi
       break;
   default:
       // aksi default
       break;
}
```

Perintah switch ini akan menerima sebuah variabel atau nilai, apapun itu, dan kemudian mencocokkannya kasus-kasus (*cases*) yang ada. Apakah nilai yang diberikan sesuai dengan nilai yang ditulis dalam kasus? Bila ya, maka kasus tersebut akan dijalankan. Kemudian, ada lagi yang perlu kita perhatikan, yaitu perintah *break*. Perintah ini digunakan untuk menghentikan switch-case. Tidak seperti if-else yang hanya mengeksekusi satu blok saja, bila satu kasus sudah terpenuhi, switch-case akan terus berjalan ke kasus-kasus selanjutnya.

Karena itulah, kita membutuhkan break untuk menghentikan jalannya switch-case ketika kondisi yang kita inginkan sudah terpenuhi. Namun, bukan berarti kita akan selalu memakai break untuk switch-case. Ada beberapa kondisi yang mana lebih baik bagi kita untuk tidak menggunakan break pada kasus tertentu.

Yang terakhir, ada *default*. Mungkin dari namanya, kalian sudah bisa menebak apa default ini. Default ini berlaku seperti *else* pada if-else ya? Jadi ia adalah pelarian terakhir bila kasus-kasus sebelumnya tidak ada yang memenuhi.

Kalau begitu, mari kita lihat contoh implementasi sederhananya.

```
import java.util.Scanner;

public class ProgramDira {
   public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Masukkan kelas anda: ");

        String kelas = scanner.next();

        switch(kelas) {
            case "01":
                System.out.println("Selamat datang di kelas 01!");
                break;
            case "02":
                System.out.println("Selamat datang di kelas 02!");
                break;
            case "03":
                System.out.println("Selamat datang di kelas 03!");
                break;
            case "04":
                System.out.println("Selamat datang di kelas 04!");
                break;
            default:
                System.out.println("Kelas tidak ada!");
                break;
        }
   }
}
```

Bila kalian jalankan program di atas, maka kalian akan dimintai input nomor kelas kalian. Dan bila nomor kelas kalian ada pada *case*, maka kalian akan disambut. Namun bila tidak, maka dia akan lari ke *default* dan mencetak “Kelas tidak ada!”. Dari sini, apakah kalian sudah mengetahui apa perbedaan if-else dan switch-case, lalu apa perbedaan kegunaan dari mereka?

Bila belum, mari kita bahas bersama-sama! Pertama-tama, cara switch-case bekerja. Berbeda dengan if-else yang memakai boolean, switch-case ini mencocokkan nilai asli nilai yang di-switch (contohnya kelas di atas) dengan nilai yang dituliskan pada *case*. Sedangkan if-else, ia menggunakan boolean, sehingga kondisi seperti apapun akan selalu dieksekusi apabila kondisi aneh tersebut menghasilkan boolean "true". Dari sini, kita juga bisa simpulkan, lho perbedaan kegunaan dari kedua perintah tersebut.

Yang pertama dan paling jelas, switch-case digunakan untuk nilai yang sudah pasti, atau nilai konstanta. Contoh misalnya tipe pasien di rumah sakit ada tiga, yaitu pasien "BPJS", "Reguler", dan "VIP". Tipe pasien-pasien tersebut tidak mungkin berubah, bukan? Hanya tiga itu saja. Oleh karena itu, di sini sebaiknya kita menggunakan switch-case karena hanya akan ada 3 kasus saja.

```
case "BPJS":
   break;
case "Reguler":
   break;
case "VIP":
   break;
```

Sedangkan if-else dapat kita gunakan untuk kondisi yang tidak kita ketahui secara pasti nilainya. Contohnya adalah rentang indeks nilai

```
if (nilai >= 80) {
   System.out.println("Nilai Anda: A");
} else if (nilai >= 70) {
   System.out.println("Nilai Anda: B");
} else if (nilai >= 50) {
   System.out.println("Nilai Anda: C");
} else if (nilai >= 30) {
   System.out.println("Nilai Anda: D");
} else {
   System.out.println("Nilai Anda: E");
}
```

Tidak mungkin, kan kita memakai switch-case di sini. Masa kita harus menulis 100 kasus? Hahaha, itu bukan kerja cerdas!

5.3.2 Tanpa memakai break?

Terakhir, kita akan membahas kasus di mana *break* bisa tidak dibutuhkan. Kita tidak membutuhkan *break* apabila kita ingin beberapa kasus memiliki proses yang sama persis. Contohnya? Mari kita ubah switch-case yang kita buat sebelumnya, yuk!

```
public class ProgramDira {
   public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Masukkan kelas anda: ");

        String kelas = scanner.next();

        switch(kelas) {
            case "01":
            case "02":
            case "03":
            case "04":
                System.out.println("Kelas ada!");
                break;
            default:
                System.out.println("Kelas tidak ada!");
                break;
        }
   }
}
```

Anggap saja kita sedang membuat program untuk cek kelas mana saja yang ada. Kita hanya membutuhkan dua output di atas, yaitu “Kelas ada!” dan “Kelas tidak ada!”. Ingat, tanpa break, switch-case akan terus berjalan ke kasus-kasus berikutnya. Jadi, misal ada yang input “01”, ia akan terus berjalan hingga “04”, dan kemudian menjalankan perintah *println* yang ada di kasus 04. Tanpa memakai break, kita bisa mempersingkat kode di atas dan kelas apapun yang valid yang pengguna pilih (01-04), maka semua kasus akan berhenti di kasus 04, dan mencetak “Kelas ada!”. Kita tidak perlu menulis

```
System.out.println("Kelas ada!");
break;
```

terus-menerus lagi untuk setiap kasus. Lebih simpel deh!

Baiklah, itu saja untuk bab switch-case. Sampai jumpa di bab selanjutnya!