7. Struktur Kendali Program

By : Mahendar Dwi Payana, S.ST., M.T

Prasyarat :

1. Pengenalan Algoritma & Flowchart
2. Pengenalan Java
3. Variabel dan Implementasi
4. Tipe Data dan Implementasi
5. Method

Referensi :

1. Buku "Pemrograman Borientasi Objek. Teori dan Implementasi Java"
2. Pengarang : Raden Budiarto
3. Tahun : 2018
4. Jenis : E-Book - ebook play.books.com

7.1 Pengenalan Struktur Kendali

Struktur kendali adalah sebuah perintah yang digunakan untuk mengatur alur program. Struktur kendali digunakan untuk mengatur alur program agar dapat berjalan sesuai dengan kebutuhan. Struktur kendali pada **java** terdiri dari 3 jenis yaitu:

1. Struktur Kendali Seleksi atau Percabangan.
2. Struktur kendali Perulangan.
3. Struktur kendali Peloncatan.

7.2 Struktur Kendali Seleksi

Struktur kendali seleksi atau percabangan adalah sebuah perintah yang digunakan untuk memilih alur program yang akan dijalankan. Struktur kendali seleksi terdiri dari 2 jenis yaitu:

1. Struktur kendali seleksi if.
2. Struktur kendali seleksi switch.

7.2.1 Struktur Kendali Seleksi *IF*

Pernyataan `IF` akan menentukan sebuah pertanyaan (atau blok kode) yang akan di eksekusi dan hanya jika persyaratan kondisi yang ditentukan benar (`true`). Jika persyaratan yang ditentukan salah, maka tidak ada yang akan terjadi. Kondisi disini merupakan sebuah ekspresi yang memiliki nilai balik `true` atau `false`.

```
if(kondisi)
   statement;
```

atau jika menggunakan multiple statement

```
if(kondisi){
   statement1;
   statement2;
   statement3;
}
```

Pernyatan `if-else` dapat digunakan apabila kita ingin mengeksekusi beberapa pertanyaan dengan kondisi `true` dan pernyataan lain dengan kondisi `false`. Bentuk *statement* `if-else`:

```
if(kondisi)
   statement1;
else
   statement2;
```

atau jika menggunakan multiple statement

```
if(kondisi){
   statement1;
   statement2;
   statement3;
}
else{
   statement4;
   statement5;
   statement6;
}
```

Jika kondisi lebih dari 2 dapat menggunakan struktur `if-else` berjenjang. Bentuk *statement* `if-else` berjenjang:

```
if(kondisi1)
   statement1;
else if(kondisi2)
   statement2;
else if(kondisi3)
   statement3;
```

```
else
   statement4;
```

atau jika menggunakan multiple statement

```
if(kondisi1){
   statement1;
   statement2;
   statement3;
}
else if(kondisi2){
   statement4;
   statement5;
   statement6;
}
else if(kondisi3){
   statement7;
   statement8;
   statement9;
}
else{
   statement10;
   statement11;
   statement12;
}
```

> Yang perlu diperhatikan adalah pertanyaan `if-else` dijalankan dari atas ke bawah. Begitu salah satu dari pernytaan benar, maka pernyataan dibawahnya tidak akan dieksekusi.

Contoh program dengan menggunakan struktur kendali seleksi `if`:

```
public class PernyaaanIFELSE{
   public static void main(String[] args){
      int skorUjian = 86;
      char grade;

      if(skorUjian >= 90){
         grade = 'A';
      }
      else if(skorUjian >= 80){
         grade = 'B';
      }
      else if(skorUjian >= 70){
         grade = 'C';
      }
```

```
        else if(skorUjian >= 60){
           grade = 'D';
        }
        else{
           grade = 'E';
        }

        System.out.println("Grade = " + grade);
    }
}
```

Output:

```
Grade = B
```

7.2.2 Struktur Kendali Seleksi *Switch*

Pernyataan `switch` adalah sebuah pernyataan yang digunakan untuk memilih salah satu dari banyak blok kode yang akan dieksekusi. Pernyataan `switch` akan mengevaluasi ekspresi yang diberikan dan mencocokkan nilai ekspresi tersebut dengan nilai ekspresi yang ada pada setiap pernyataan `case`. Jika ada nilai ekspresi yang cocok, maka pernyataan yang ada pada `case` tersebut akan dieksekusi. Jika tidak ada nilai ekspresi yang cocok, maka pernyataan yang ada pada `default` akan dieksekusi.

Berbeda dengan pernyataan `IF` yang mengeksekusi kode program secara berurutan dari atas ke bawah, Pernyataan *switch* diimplementasikan menggunakan table percabangan (*jump table*) yang memungkinkan untuk trasnfer kontrol program ke salah satu dari banyak blok kode yang ada. Bentuk *statement* `switch`:

```
switch(ekspresi){
   case nilai1:
      statement1;
      break;
   case nilai2:
      statement2;
      break;
   case nilai3:
      statement3;
      break;
   default:
      statement4;
}
```

Ekspresi pada `switch` harus bertipe data `byte`, `short`, `int`, `char`, `String`, atau `enum`. Setiap `case` harus bersifat unik atau tidak boleh duplikat.

> Jika `switch` tidak memiliki `case` yang cocok maka akan dieksekusi `default`. Jika tidak ada `default` maka tidak ada kode yang akan dieksekusi.

Contoh program dengan menggunakan struktur kendali seleksi `switch`:

```
public class PernyataanSWITCH{
   public static void main (String[] args){
      int bulan = 4;
      String namaBulan;

      switch(bulan){
         case 1:
            namaBulan = "Januari";
            break;
         case 2:
            namaBulan = "Februari";
            break;
         case 3:
            namaBulan = "Maret";
            break;
         case 4:
            namaBulan = "April";
            break;
         case 5:
            namaBulan = "Mei";
            break;
         case 6:
            namaBulan = "Juni";
            break;
         case 7:
            namaBulan = "Juli";
            break;
         case 8:
            namaBulan = "Agustus";
            break;
         case 9:
            namaBulan = "September";
            break;
         case 10:
            namaBulan = "Oktober";
            break;
         case 11:
            namaBulan = "November";
            break;
         case 12:
            namaBulan = "Desember";
```

```
            break;
         default:
            namaBulan = "Silahakan pilih 1-12";
      }

      System.out.println("Nama bulan = " + namaBulan);
   }
}
```

7.2.3 Pernyataan Kondisi Bersarang

Sebuah pertanyaan kondisi bersarang (*nested condition*) biasa digunakan dalam pemrograman. Pernyataan kondisi bersarang adalah sebuah pernyataan kondisi yang berada di dalam pernyataan kondisi lainnya. Bentuk *statement* pernyataan kondisi bersarang:

```
public class KondisiBersarang {
   public static void main(String[] args){
      int bulan = 2;
      int tahun = 2023;

      switch (bulan){
         case 1:
         case 3:
         case 5:
         case 7:
         case 8:
         case 10:
         case 12:
            System.out.println("Jumlah hari = 31");
            break;
         case 4:
         case 6:
         case 9:
         case 11:
            System.out.println("Jumlah hari = 30");
            break;
         case 2:
            if(((tahun % 4 == 0) && !(tahun % 100 == 0)) || (tahun
% 400 == 0))
               System.out.println("Jumlah hari = 29");
            else
               System.out.println("Jumlah hari = 28");
            break;
         default:
            System.out.println("Bulan / Tahun tidak valid");
      }
   }
}
```

Output:

```
Jumlah hari = 28
```

7.3 Struktur Kendali Perulangan

Struktur kendali perulangan adalah sebuah perintah yang digunakan untuk mengulang eksekusi sebuah blok kode program. Struktur kendali perulangan terdiri dari 3 jenis yaitu:

1. Struktur kendali perulangan `for`.
2. Struktur kendali perulangan `while`.
3. Struktur kendali perulangan `do-while`.

Pernyataan perulangan mengeksekusi set instruksi berulang kali sampai kondisi terpenuhi. Pernyataan perulangan akan mengevaluasi kondisi yang diberikan. Jika kondisi bernilai `true`, maka pernyataan perulangan akan dieksekusi. Jika kondisi bernilai `false`, maka pernyataan perulangan akan berhenti dieksekusi.

7.3.1 Struktur Kendali Perulangan *FOR*

Pernyataan perulangan `for` digunakan untuk mengulang eksekusi sebuah blok kode program sebanyak jumlah tertentu. Bentuk *statement* pernyataan perulangan `for`:

```
for(inisialisasi; kondisi; perubahan){
   statement;
}
```

> Pernyataan `for` akan mengeksekusi `statement` berulang kali sampai kondisi bernilai `false`. Artinya selama kondisi bernilai `true`, maka `statement` akan terus dieksekusi.

Pernyataan `for` terdiri dari 3 bagian yaitu:

1. Inisialisasi : bagian ini digunakan untuk menginisialisasi variabel yang digunakan dalam perulangan.
2. Kondisi : bagian ini digunakan untuk mengevaluasi kondisi yang diberikan. Jika kondisi bernilai `true`, maka pernyataan perulangan akan dieksekusi. Jika kondisi bernilai `false`, maka pernyataan perulangan akan berhenti dieksekusi.
3. Perubahan : bagian ini digunakan untuk mengubah nilai variabel yang digunakan dalam perulangan.

Contoh program dengan menggunakan struktur kendali perulangan `for`:

```
```

```
public class BilPrima{
   public static void main(String args[]){
      int angka;
      boolean isPrisma = true;
      angka = 11;

      for(int i = 2; i <= angka / 2; i++){
         if(angka % i == 0){
            isPrisma = false;
            break;
         }
      }

      if(isPrisma){

         System.out.println(angka + " adalah bilangan prima");
      }else{

         System.out.println(angka + " bukan bilangan prima");
      }
   }
}
```

Output:

```
11 adalah bilangan prima
```

7.3.2 Struktur Kendali Perulangan *WHILE*

Pernyataan perulangan `while` digunakan untuk mengulang eksekusi sebuah blok kode program selama kondisi bernilai `true`. Bentuk *statement* pernyataan perulangan `while`:

```
while(kondisi){
   statement;
}
```

> Pernyataan `while` akan mengeksekusi `statement` berulang kali sampai kondisi bernilai `false`. Artinya selama kondisi bernilai `true`, maka `statement` akan terus dieksekusi.

Contoh program dengan menggunakan struktur kendali perulangan `while`:

```
public static void main(String[] args){
   int i = 10;
```

```
    while(i > 0){
       System.out.println("Hitung mundul: " + i);
       i--;
    }
}
```

Output:

```
Hitung mundur: 10
```

Latihan dengan user input:

```
import java.util.Scanner;

public class HitungMundur{
   public static void main(String[] args){
      Scanner input = new Scanner(System.in);
      int i;

      System.out.print("Masukkan angka: ");
      i = input.nextInt();

      while(i > 0){
         System.out.println("Hitung mundur: " + i);
         i--;
      }
   }
}
```

7.3.3 Struktur Kendali Perulangan *DO-WHILE*

Pernyataan perulangan `do-while` merupakan bentuk `loop` yang paling dasar dalam `java`. Jika ekspresi kondisional mengendalikan `loop` *while* bernilai `false`, maka *statement* pada tubuh `loop` tidak akan dieksekusi sama sekali. Hal ini berbeda dengan perulangan `do-while` karena bentuk perualngan `do-while` *block statement* pada tubuh `loop` setidaknya akan dieksekusi satu kali walau ekspresi kondisional bernilai `false`. Bentuk *statement* pernyataan perulangan `do-while`:

```
do{
   statement;
}while(kondisi);
```

Contoh program dengan menggunakan struktur kendali perulangan `do-while`:

```
public class HitungMundur{
   public static void main(String[] args){
      int i = 10;
      do{
         System.out.println("Hitung mundur: " + i);
         i--;
      }while(i > 0);
   }
}
```

Program pilih menu:

```
import java.util.Scanner;

public class PilihMenu{
   public static void main(String[] args){
      Scanner input = new Scanner(System.in);
      int pilihan;

      do{
         System.out.println("Menu");
         System.out.println("1. Menu 1");
         System.out.println("2. Menu 2");
         System.out.println("3. Menu 3");
         System.out.println("4. Keluar");
         System.out.print("Pilih menu: ");
         pilihan = input.nextInt();

         switch(pilihan){
            case 1:
               System.out.println("Anda memilih menu 1");
               break;
            case 2:
               System.out.println("Anda memilih menu 2");
               break;
            case 3:
               System.out.println("Anda memilih menu 3");
               break;
            case 4:
               System.out.println("Anda memilih keluar");
               break;
            default:
               System.out.println("Pilihan tidak tersedia");
         }
      }while(pilihan != 4);
```

```
    }
}
```

7.4 Struktur Kendali Peloncatan

Java mendukung 3 pernyataan perloncatan (*jump statement*) yaitu :

1. Pernyataan `break`.
2. Pernyataan `continue`.
3. Pernyataan `return`.

Pernyataan - pernyataan ini akan mengalihkan aliran kontrol ke bagian tertentu dari program. Pernyataan ini memungkinkan kita untuk menjalan program secara tidak berurutan.

7.4.1 Pernyataan *BREAK*

Pernyataan `break` digunakan untuk menghentikan eksekusi sebuah blok kode program. Pernyataan `break` biasanya digunakan dalam pernyataan `switch` dan pernyataan perulangan. Bentuk *statement* pernyataan `break`:

```
break;
```

Contoh program dengan menggunakan pernyataan `break`:

```
public class PernyataanBREAK{
   public static void main(String[] args){
      int i = 1;
      while(i <= 10){
         if(i == 5){
            break;
         }
         System.out.println(i);
         i++;
      }
   }
}
```

7.4.2 Pernyataan *CONTINUE*

Pernyataan `continue` digunakan untuk melanjutkan eksekusi sebuah blok kode program. Pernyataan `continue` biasanya digunakan dalam pernyataan perulangan. Bentuk *statement* pernyataan `continue`:

```
continue;
```

Contoh program dengan menggunakan pernyataan `continue`:

```
public class PernyataanCONTINUE{
   public static void main(String[] args){
      int i = 1;
      while(i <= 10){
         if(i == 5){
            i++;
            continue;
         }
         System.out.println(i);
         i++;
      }
   }
}
```

7.4.3 Pernyataan *RETURN*

Pernyataan `return` digunakan untuk mengembalikan nilai dari sebuah metode. Pernyataan `return` biasanya digunakan dalam metode. Bentuk *statement* pernyataan `return`:

```
return;
```

Contoh program dengan menggunakan pernyataan `return`:

```
public class PernyataanRETURN{
   public static void main(String[] args){
      int i = 1;
      while(i <= 10){
         if(i == 5){
            return;
         }
         System.out.println(i);
         i++;
      }
   }
}
```

Latihan Pagi

1. Latihan 1

```
int nilaiAkhir = 70;
        char grade;

        //kondisi mengubah nilai grade
        if(nilaiAkhir >= 80){
            grade = 'A';
        }else if(nilaiAkhir < 80 && nilaiAkhir >= 70){
            grade = 'B';
        }else if(nilaiAkhir < 70 && nilaiAkhir >=60){
            grade = 'C';
        }else{
            grade = 'F';
        }

        // ini kondisi untuk mengeluarkan output
        if(grade == 'F'){
            System.out.println("Anda gagal!");
        }else{
            System.out.println("Nilai anda = " + grade);
        }
```

<< Back

Next >>