Pertemuan IV

STRUKTUR KONTROL

4.1. Percabangan

Percabangan adalah pernyataan dari Java yang mengijinkan user untuk memilih dan mengeksekusi blok kode spesifik sesuai kondisi yang telah ditentukan dan mengabaikan blok kode yang lain.

Pernyataan-pernyataan yang dapat digunakan untuk struktur kontrol percabangan antara lain :

- *if*
- *if - else*
- *switch*

4.1.1. Percabangan dengan *if*

Pernyataan *if* digunakan untuk menentukan sebuah pernyataan (atau blok kode) akan dieksekusi jika dan hanya jika persyaratan bernilai benar (true).

Bentuk dari pernyataan *if* adalah :

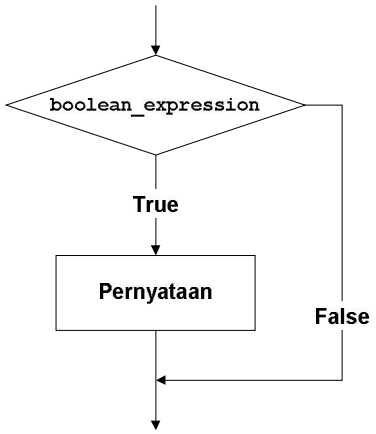
```
if(boolean_expression) statement;
```

atau :

```
if (boolean_expression){
          statement1;
          statement2;
          . . .
}
```

dimana, *boolean_expression* adalah sebuah penyataan logika yang bernilai benar (*true*) atau salah (*false*), atau variabel bertipe *boolean*.

Pernyataan *if* dapat digambarkan dalam *flowchart* sebagai berikut :



Gambar 4.1. *Flowchart* pernyataan *if*

Penggunaan pernyataan *if* dalam program dapat dilihat pada contoh program dibawah ini:

Pernyataanif1.java

```
public class Pernyataanif1{
    public static void main(String args[]){
        Byte grade = 95;
        if ( grade > 80 ) System.out.println("Selamat Anda Lulus!");
    }
}
```

atau :

Pernyataanif2.java

```
public class Pernyataanif2{
    public static void main(String args[]){
        Byte grade = 95;
        if ( grade > 80 ){
            System.out.println("Selamat Anda Lulus");
            System.out.println("Dengan nilai "+grade+"!");
        }
    }
}
```

4.1.2. Percabangan dengan *if - else*

Pernyataan *if – else* digunakan untuk menentukan dua kondisi alur program, jika persyaratan bernilai benar (true) maka sebuah pernyataan (atau blok kode) akan dieksekusi. Sedangkan jika persyaratan bernilai salah (false) maka sebuah pernyataan (atau blok kode) lain yang akan dieksekusi.

Bentuk dari pernyataan *if - else* adalah :

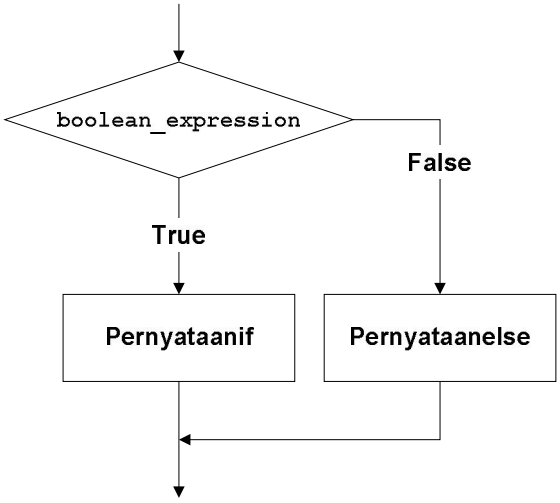
```
if (boolean_expression)
    statementif;
else
    statementelse;
```

atau :

```
if (boolean_expression){
    statementif1;
    statementif2;
    . . .
}
else){
    statementelse1;
    statementelse2;
    . . .
}
```

dimana, *boolean_expression* adalah sebuah penyataan logika yang bernilai benar (*true*) atau salah (*false*), atau variabel bertipe *boolean*.

Pernyataan *if - else* dapat digambarkan dalam *flowchart* sebagai berikut :



Gambar 4.2. *Flowchart* pernyataan *if-else*

Penggunaan pernyataan *if - else* dalam program dapat dilihat pada contoh program dibawah ini:

Pernyataanifelse1.java

```
 import java.io.*;

 public class Pernyataanifelse1{
    public static void main(String args[]){
       BufferedReader dataIn = new BufferedReader(new
InputStreamReader(System.in));
       String gradestring = "";
       Byte grade = 0;

       System.out.print("Ketik nilai Anda : ");
       try{
          gradestring = dataIn.readLine();
       }
       catch( IOException e ){
          System.out.println("Ada kesalahan !");
       }

       grade = new Byte (gradestring);

       if ( grade >= 80 ) System.out.println("Selamat Anda Lulus!");
       else System.out.println("Maaf Anda Belum Lulus!");
    }
 }
```

Atau

Pernyataanifelse2.java

```
 import java.io.*;

 public class Pernyataanifelse2{
    public static void main(String args[]){
       BufferedReader dataIn = new BufferedReader(new
InputStreamReader(System.in));
```

```
      String gradestring = "";
      Byte grade = 0;

      System.out.print("Ketik nilai Anda : ");
      try{
         gradestring = dataIn.readLine();
      }
      catch( IOException e ){
         System.out.println("Ada kesalahan !");
      }

      grade = new Byte (gradestring);

      if ( grade >= 80 ) {
         System.out.println("Selamat Anda Lulus!");
         System.out.println("Karena nilai Anda "+grade+"!");
      }
      else {
         System.out.println("Maaf Anda Belum Lulus!");
         System.out.println("Karena nilai Anda "+grade+"!");
      }
   }
}
```

Karena input yang kita ketik berupa data bertipe string, maka harus dikonversi ke bentuk byte menggunakan pernyataan :

```
grade = new Byte (gradestring);
```

Pernyataan pada bagian kondisi *else* dari blok *if-else* dapat menjadi struktur *if-else* yang lain. Kondisi struktur seperti ini mengijinkan kita untuk membuat seleksi dengan persyaratan yang lebih kompleks.

Bentuk pernyataan *if-else if* :

```
if ( boolean_expression1 )
    statement1;
else if ( boolean_expression2 )
    statement2;
else
    statement3;
```

Kita dapat memiliki banyak blok *else-if* sesudah pernyataan *if*. Blok *else* bersifat opsional dan dapat dihilangkan. Pada contoh yang ditampilkan di atas, jika *boolean_expression1* bernilai benar (*true*), maka program akan mengeksekusi *statement1* dan melewati pernyataan yang lain. Jika *boolean_expression1* bernilai salah (*false*) dan *boolean_expression2* bernilai benar (*true*), maka program akan mengeksekusi *statement2* dan mengabaikan *statement* yang lain.

Penggunaan pernyataan *if - else* dalam program dapat dilihat pada contoh program dibawah ini :

Pernyataanifelseif.java

```
import java.io.*;

public class Pernyataanifelseif{
   public static void main(String args[]){
```

```
      BufferedReader dataIn = new BufferedReader(new
InputStreamReader(System.in));
      String gradestring = "";
      byte grade = 0;

      System.out.print("Ketik nilai Anda : ");
      try{
         gradestring = dataIn.readLine();
      }
      catch( IOException e ){
         System.out.println("Ada kesalahan !");
      }

      //Konversi nilai string ke Byte
      grade = new Byte (gradestring);

      if ( grade == 100 ) {
         System.out.println("Selamat Anda Lulus dengan Kategori
Sempurna!");
         System.out.println("Karena nilai Anda "+grade+"!");
      }
      else if ( grade >= 95 ) {
         System.out.println("Selamat Anda Lulus dengan Kategori Sangat
Memuaskan!");
         System.out.println("Karena nilai Anda "+grade+"!");
      }
      else if ( grade >= 90 ) {
         System.out.println("Selamat Anda Lulus dengan Kategori
Memuaskan!");
         System.out.println("Karena nilai Anda "+grade+"!");
      }
      else if ( grade >= 80 ) {
         System.out.println("Selamat Anda Lulus dengan Kategori Baik!");
         System.out.println("Karena nilai Anda "+grade+"!");
      }
      else {
         System.out.println("Maaf Anda Belum Lulus!");
         System.out.println("Karena nilai Anda "+grade+"!");
      }
   }
 }
```

4.1.3. Percabangan dengan *switch*

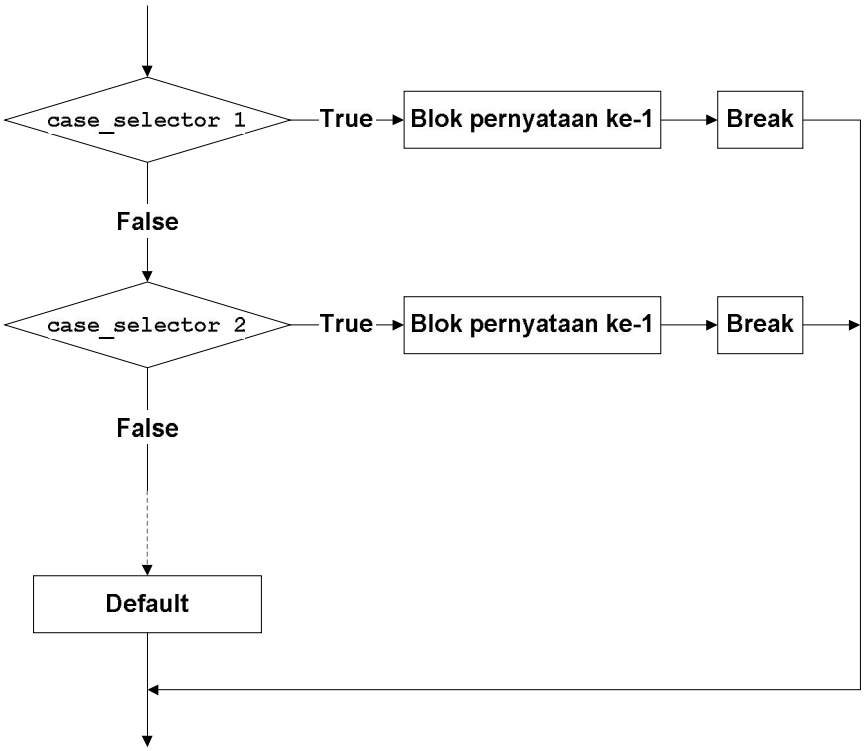
Cara lain untuk membuat cabang adalah dengan menggunakan kata kunci *switch*. *Switch* digunakan untuk menangani percabangan yang memiliki banyak kondisi.

Bentuk statement *switch* adalah :

```
switch (switch_expression){
   case case_selector1:
      statement1; //
      statement2; //blok pernyataan ke-1
      . . . //
      break;
   case case_selector2:
      statement1; //
      statement2; //blok pernyataan ke-2
      . . . //
      break;
   . . .
```

```
   default:
     statement1; //
     statement2; //block n
     . . . //
     break;
}
```

Pernyataan *switch* dapat digambarkan dalam *flowchart* sebagai berikut :



Gambar 4.3. *Flowchart* pernyataan *switch*

switch_expression adalah ekspresi yang menghasilkan nilai *integral* (untuk java SE 7 sudah dapat menggunakan nilai *String*). *case_selector1*, *case_selector2* dan seterusnya adalah konstanta unik dari nilai *switch_expression*.

Ketika pernyataan *switch* ditemukan pada potongan kode program, java pertama kali akan memeriksa *switch_expression*, dan menuju ke *case* yang akan menyamakan nilai yang dimiliki oleh *switch_expression* dengan nilai *case_selector*. Jika ditemukan program akan mengeksekusi pernyataan dari kode setelah *case* yang ditemukan sampai menemui pernyataan *break*, selanjutnya akan mengabaikan pernyataan yang lainnya hingga akhir dari struktur pernyataan *switch*.

Jika tidak ditemui *case* yang cocok, maka program akan mengeksekusi blok *default* (kalau ada), karena bagian blok *default* bersifat opsional. Sebuah pernyataan *switch* bisa jadi tidak memiliki blok *default*.

Tidak seperti pada pernyataan ***if***, beberapa pernyataan pada struktur pernyataan *switch* akan dieksekusi tanpa memerlukan tanda kurung kurawal (**{}**).

Ketika sebuah *case* pada pernyataan *switch* menemui kecocokan, semua pernyataan setelah case tersebut akan dieksekusi. Untuk menghindari program mengeksekusi pernyataan pada *case* berikutnya, kita menggunakan pernyataan *break* sebagai pernyataan akhir pada setiap blok *case*.

Program if - else bertingkat sebelumnya dapat kita buat menggunakan pernyataan switch sebagai berikut :

Pernyataanswitch.java

```
 import java.io.*;

 public class Pernyataanswitch{
    public static void main(String args[]){
       BufferedReader dataIn = new BufferedReader(new
InputStreamReader(System.in));
      String angkastring = "";
      byte angka = 0;

      System.out.print("Ketik angka 0..9 : ");
      try{
        angkastring = dataIn.readLine();
      }
      catch( IOException e ){
        System.out.println("Ada kesalahan !");
      }

      //Konversi nilai string ke Byte
      angka = new Byte (angkastring);

      switch (angka){
        case 0: System.out.println("Angka yang diketik adalah nol");
                break;
        case 1: System.out.println("Angka yang diketik adalah satu");
                break;
        case 2: System.out.println("Angka yang diketik adalah dua");
                break;
        case 3: System.out.println("Angka yang diketik adalah tiga");
                break;
        case 4: System.out.println("Angka yang diketik adalah empat");
                break;
        case 5: System.out.println("Angka yang diketik adalah lima");
                break;
        case 6: System.out.println("Angka yang diketik adalah enam");
                break;
        case 7: System.out.println("Angka yang diketik adalah tujuh");
                break;
        case 8: System.out.println("Angka yang diketik adalah delapan");
                break;
        case 9: System.out.println("Angka yang diketik adalah
sembilan");
                break;
        default: System.out.println("Angka yang diketik tidak sesuai");

      }
    }
 }
```

4.2. Perulangan

Struktur kontrol perulangan adalah berupa pernyataan dari Java yang menyebabkan eksekusi terhadap blok kode program dilakukan berulang-ulang sesuai dengan kondisi tertentu. Ada tiga macam struktur kontrol pengulangan, yaitu *for*, *while*, dan *do - while*.

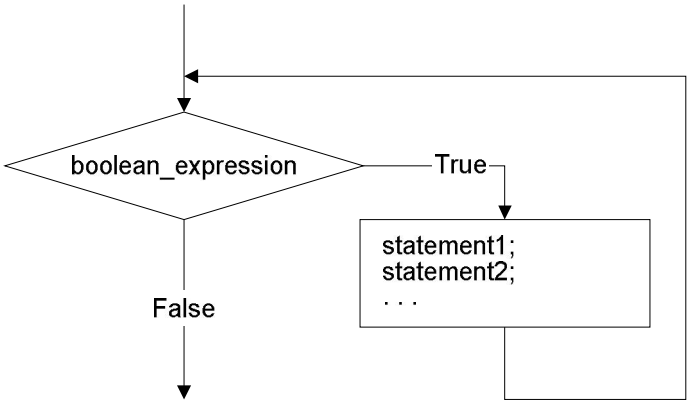
4.2.1. Menggunakan *while*

Pernyataan pengulangan *while* adalah pernyataan atau blok pernyataan yang diulang-ulang sampai mencapai kondisi yang tidak cocok.

Bentuk pernyataan while :

```
while (boolean_expression){
    statement1;
    statement2;
    . . .
}
```

Alurnya dapat dilihat pada flowchart di bawah ini :



Gambar 4.4. *Flowchart* pernyataan *while*

Pernyataan di dalam pengulangan *while* akan dieksekusi berulang-ulang selama kondisi *boolean_expression* bernilai benar (*true*).

Contoh, pada kode dibawah ini :

```
int i = 4;
while  ( i > 0 ){
    System.out.print(i);
    i--;
}
```

Contoh diatas akan mencetak angka 4321 pada layar. Perlu dicatat jika bagian i--; dihilangkan, akan menghasilkan pengulangan yang terus menerus (*infinite loop*). Sehingga, ketika menggunakan *while loop* atau bentuk pengulangan yang lain, pastikan untuk memberikan pernyataan yang membuat pengulangan berhenti pada suatu kondisi.

4.2.2. Menggunakan *do - while*

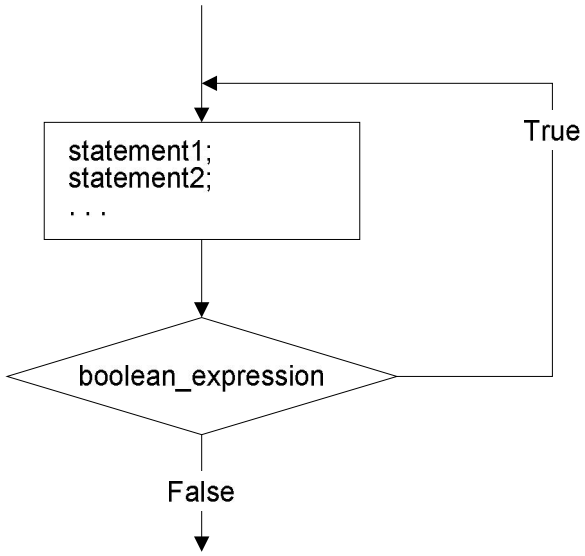
Pengulangan *do-while* mirip dengan pengulangan *while*. Pernyataan di dalam pengulangan *do-while* akan dieksekusi beberapa kali selama kondisi bernilai benar(*true*).

Perbedaan antara *while* dan *do-while* adalah dimana pernyataan di dalam pengulangan *do-while* akan dieksekusi sedikitnya satu kali, sedangkan pada perngulangan *while* ada kemungkinan tidak dieksekusi.

Bentuk pernyataan *do-while* :

```
do{
     statement1;
     statement2;
     . . .
}while( boolean_expression );
```

Alurnya dapat dilihat pada flowchart di bawah ini :



Gambar 4.5. *Flowchart* pernyataan *do - while*

Pernyataan di dalam *do-while loop* akan dieksekusi pertama kali, dan akan dievaluasi kondisi dari *boolean_expression*. Jika nilai pada *boolean_expression* tersebut bernilai *true*, pernyataan di dalam *do-while loop* akan dieksekusi lagi.

Berikut ini contoh pengulangan *do-while* :

```
int x = 0;
do
{
    System.out.print(x);
    x++;
}while (x<10);
```

Contoh diatas akan memberikan output 0123456789 pada layar.

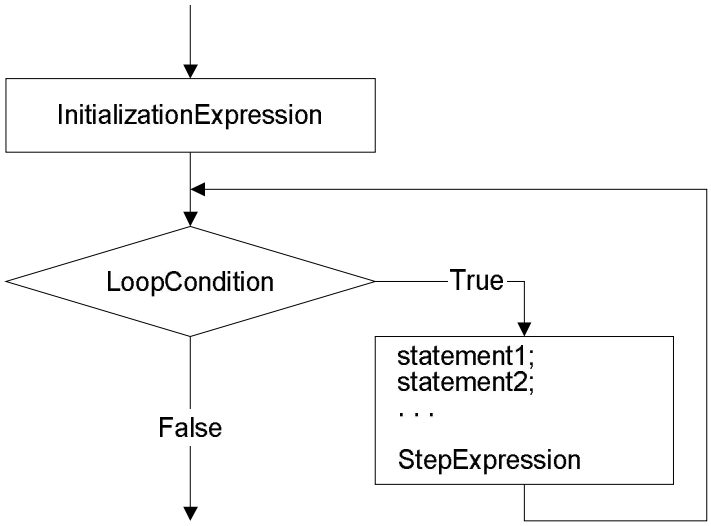
4.2.3. Menggunakan *for*

Pernyataan pengulangan *for* memiliki kondisi hampir mirip seperti struktur pengulangan sebelumnya yaitu melakukan pengulangan untuk mengeksekusi kode yang sama selama kondisi/jumlah tertentu.

Bentuk dari perulangan for :

```
for (InitializationExpression; LoopCondition; StepExpression){
    statement1;
    statement2;
    . . .
}
```

Alurnya dapat dilihat pada flowchart di bawah ini :



Gambar 4.6. *Flowchart* pernyataan *for*

dimana, *InitializationExpression* adalah inisialisasi dari variabel loop. *LoopCondition digunakan untuk* membandingkan variabel pengulangan pada nilai batas tertentu. *StepExpression* untuk melakukan update pada variabel loop.

Berikut ini adalah contoh dari for loop :

```
int i;
for( i = 0; i < 10; i++ ){
    System.out.print(i);
}
```

Pada contoh ini, pernyataan i=0 merupakan inisialisasi dari variabel. Selanjutnya, kondisi i<10 diperiksa. Jika kondisi bernilai true, pernyataan di dalam pengulangan *for* dieksekusi. Kemudian, ekspresi i++ dieksekusi, lalu akan kembali pada bagian pemeriksaan terhadap kondisi i<10 lagi. Kondisi ini akan dilakukan berulang-ulang sampai mencapai nilai yang salah (false).

Referensi:

1. Hariyanto, Bambang, (2007), *Esensi-esensiBahasaPemrograman Java*, Edisi 2, Informatika Bandung, November 2007.
2. Tim Pengembang JENI, JENI 1-6, Depdiknas, 2007
3. Utomo, EkoPriyo, (2009), *PanduanMudahMengenalBahasa Java*, YramaWidya, Juni 2009.