BAB 6

Struktur Kontrol

6.1 Tujuan

Pada bab sebelumnya, kita sudah mendapatkan contoh dari program terstruktur, dimana setiap pernyataan dieksekusi setelah pernyataan sebelumnya sesuai dengan urutannya. Pada bagian ini, kita akan mempelajari tentang struktur kontrol dimana kita dapat mengubah cara eksekusi pada pernyataan yang dibuat di program kita.

Pada akhir pembahasan, diharapkan pembaca dapat :

- Menggunakan struktur kontrol pemilihan (if, else, switch) yang digunakan untuk memilih blok kode yang akan dieksekusi
- Menggunakan struktur kontrol pengulangan (while, do-while, for) untuk mengeksekusi blok tertentu pada program beberapa kali.
- Menggunakan pernyataan-pernyataan percabangan (break, continue, return) yang digunakan untuk mengatur arah dari aliran program.

6.2 Struktur Kontrol Pemilihan

Struktur kontrol pemilihan adalah pernyataan dari Java yang mengijinkan user untuk memilih dan mengeksekusi blok kode spesifik dan mengabaikan blok kode yang lain.

6.2.1 Statement if

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

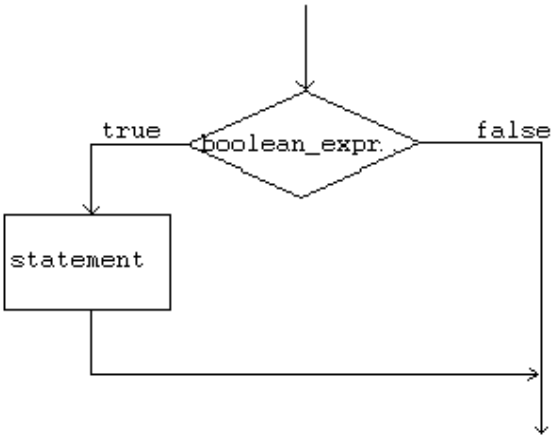
Bentuk dari pernyataan if,

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

atau

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

dimana, *boolean_expression* adalah sebuah penyataan logika *(true/false)* atau variabel bertipe *boolean*.



Gambar 1: Flowchart Statement If

Berikut ini adalah potongan kode dari pernyataan if:

```
int grade = 68;

if( grade > 60 ) System.out.println("Congratulations!");
```

atau

```
int grade = 68;

if( grade > 60 ){
       System.out.println("Congratulations!");
       System.out.println("You passed!");
}
```

Petunjuk Penulisan Program :

1. ***Boolean_expression*** *pada pernyataan if harus merupakan nilai boolean).Hal ini berarti persyaratan harus bernilai* ***true*** *atau* ***false****.*
2. *Masukkan statement di dalam blok if. Contohnya,*

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

6.2.2 Statement if-else

Pernyataan *if-else* digunakan apabila kita ingin mengeksekusi beberapa pernyataan dengan kondisi *true* dan pernyataan yang lain dengan kondisi *false*.

Bentuk statement if-else,

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

dapat juga ditulis seperti,

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

Berikut ini contoh code statement if-else,

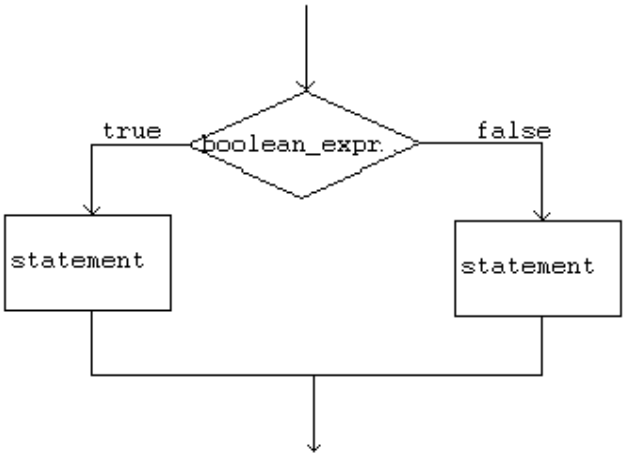
```
int grade = 68;

if( grade > 60 )   System.out.println("Congratulations!");
else            System.out.println("Sorry you failed");
```

atau

```
int grade = 68;

if( grade > 60 ){
       System.out.println("Congratulations!");
       System.out.println("You passed!");
}
else{
       System.out.println("Sorry you failed");
}
```



Gambar 2: Flowchart Statement If-Else

Petunjuk Penulisan Program :

1. *Untuk menghindari kebingungan, selalu letakkan sebuah pernyataan atau beberapa pernyataan di dalam blok if-else didalam tanda kurawal {},*
2. *Anda dapat memiliki blok if-else yang bersarang. Ini berarti anda dapat memiliki blok if-else yang lain di dalam blok if-else. Contohnya,*

```
if( boolean_expression ){
        if( boolean_expression ){
                . . .
        }
}
else{
 . . .
}
```

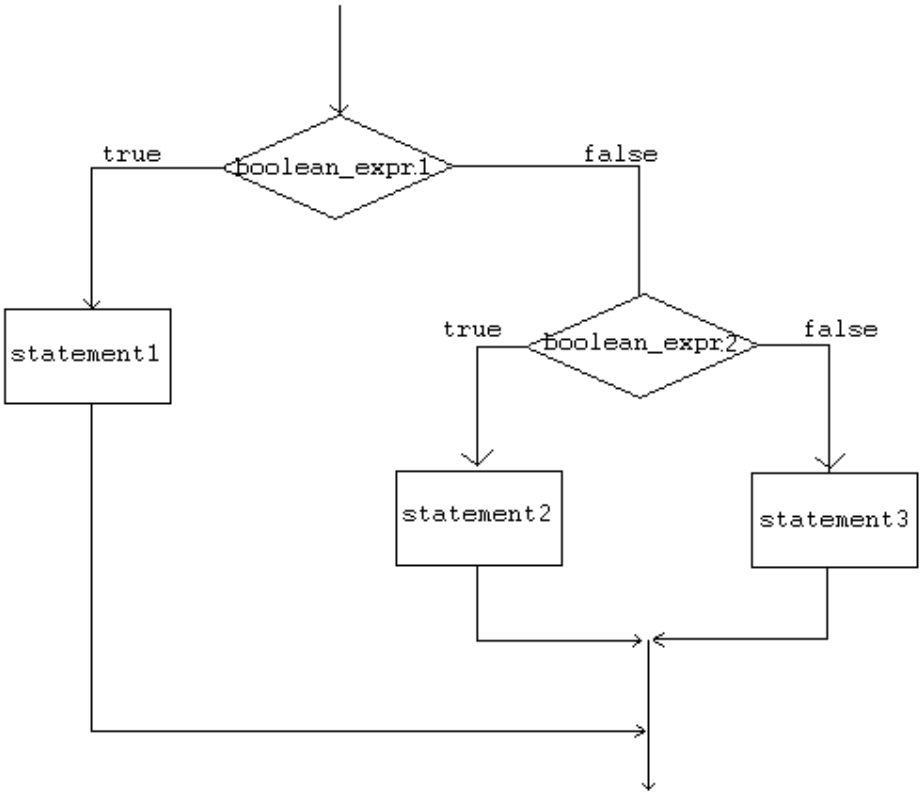
6.2.3 Statement if-else-if

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 persyaratan yang lebih kompleks.

Bentuk statement if-else if,

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

Sebagai catatan : anda 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 *true*, maka program akan mengeksekusi *statement1* dan melewati pernyataan yang lain. Jika *boolean_expression2* bernilai *true*, maka program akan mengeksekusi *statement2* dan melewati *statement3*.



Gambar 3: Flowchart Statement If-Else-If

Berikut ini contoh code statement *if-else-if*

```
int grade = 68;

if( grade > 90 ){
	System.out.println("Very good!");
}
else if( grade > 60 ){
	System.out.println("Very good!");
}
else{
	System.out.println("Sorry you failed");
}
```

6.2.4 Kesalahan umum ketika menggunakan statement if-else:

1. Kondisi pada statement *if* tidak mengevaluasi nilai logika *boolean*. Contohnya :

```
//SALAH
int number = 0;
if( number ){
	//some statements here
}
```

Variabel *number* tidak memiliki nilai Boolean.

2. Menggunakan operator = sebagai operator perbandingan yang seharusnya adalah operator == . Contohnya,

```
//SALAH
int number = 0;
if( number = 0 ){
	//Beberapa pernyataan
}
```

Seharusnya kode tersebut ditulis,

```
//BENAR
int number = 0;
if( number == 0 ){
	//beberapa pernyataan
}
```

3. Penulisan **elseif** yang seharusnya ditulis sebagai **else if**.

6.2.5 Contoh statement if-else-else if

```
public class Grade
{
    public static void main( String[] args )
    {
        double grade = 92.0;

        if( grade >= 90 ){
               System.out.println( "Excellent!" );
        }
        else if( (grade < 90) && (grade >= 80)){
               System.out.println("Good job!" );
        }
        else if( (grade < 80) && (grade >= 60)){
               System.out.println("Study harder!" );
        }
        else{
               System.out.println("Sorry, you failed.");
        }
    }
}
```

6.2.6 Statement switch

Cara lain untuk membuat cabang adalah dengan menggunakan kata kunci ***switch***. *Switch* mengkonstruksikan cabang untuk beberapa kondisi dari nilai.

Bentuk statement switch,

```
switch( switch_expression ){
       case case_selector1:
                statement1;        //
                statement2;        //block 1
                . . .              //
                break;

       case case_selector2:
                statement1;        //
                statement2;        //block 2
                . . .              //
                break;
       . . .
       default:
                statement1;        //
                statement2;        //block n
                . . .              //
                break;
}
```

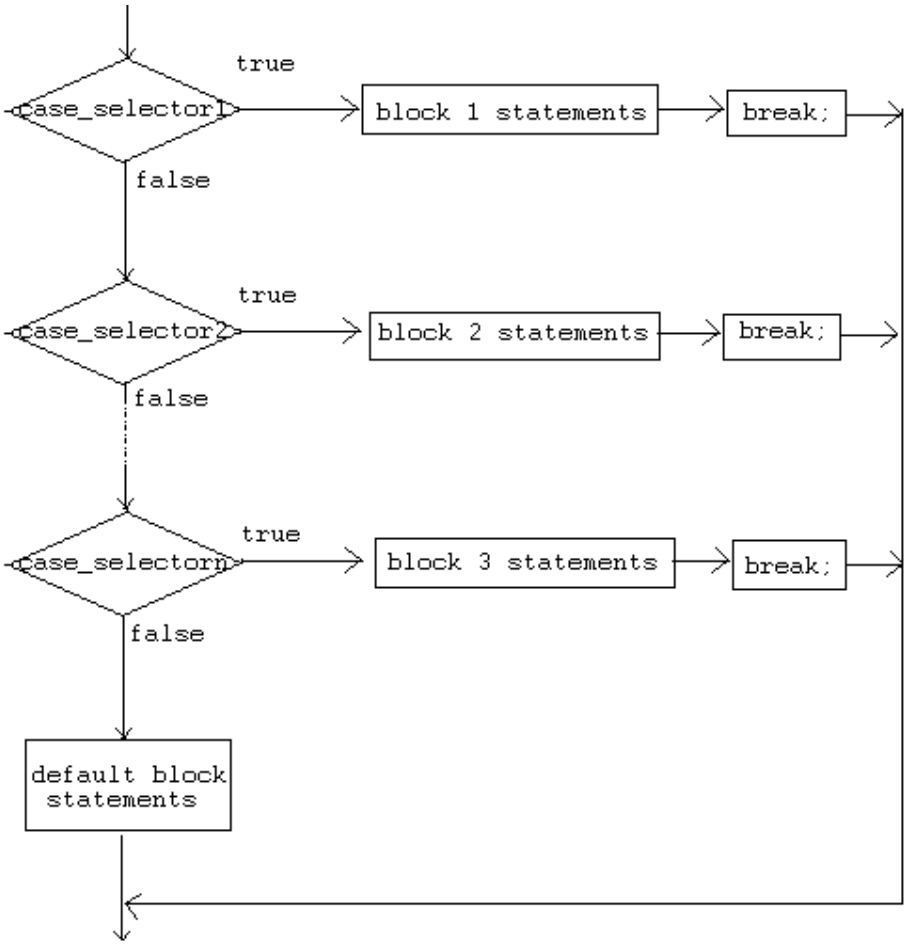
switch_expression adalah ekspresi **integer** atau **karakter** dan *case_selector1*, *case_selector2* dan seterusnya adalah konstanta unik dari nilai *integer* atau karakter.

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*. Selanjutnya program akan mengeksekusi pernyataan pada dari kode setelah *case* yang ditemukan sampai menemui pernyataan *break*, selanjutnya akan mengabaikan pernyataan yang lainnya hingga akhir dari struktur dari pernyataan *switch*.

Jika tidak ditemui *case* yang cocok, maka program akan mengeksekusi blok *default*. Sebagai catatan, bahwa bagian blok *default* adalah opsional. Sebuah pernyataan *switch* bisa jadi tidak memiliki blok kode *default*.

CATATAN:

- 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 pada case tersebut akan dieksekusi. Tidak hanya demikian, pernyataan lain yang berada pada case yang sesuai juga akan dieksekusi.
- Untuk menghindari program mengeksekusi pernyataan pada *case* berikutnya, kita menggunakan pernyataan ***break*** sebagai pernyataan akhir pada setiap blok *case*.



Gambar 4: Flowchart Statement Switch

Petunjuk Penulisan Program :

1. *Menentukan penggunaan pernyataan if atau pernyataan switch adalah sebuah keputusan programmer. Programmer dapat menentukan pernyataan yang mana yang akan dipakai berdasarkan kemudahan membaca program dan faktor-faktor yang lain.*
2. *Pernyataan if dapat digunakan untuk membuat keputusan berdasarkan rentang nilai tertentu atau kondisi tertentu, sedangkan pernyataan switch membuat keputusan hanya berdasarkan nilai unik dari tipe integer atau karakter.*

6.2.7 Contoh statement switch

```
public class Grade
{
    public static void main( String[] args )
    {
        int grade = 92;

        switch(grade){
       case 100:
               System.out.println( "Excellent!" );
               break;
       case 90:
               System.out.println("Good job!" );
               break;
       case 80:
               System.out.println("Study harder!" );
               break;
       default:
               System.out.println("Sorry, you failed.");
        }
    }
}
```

6.3 Struktur Kontrol Perulangan

Struktur kontrol pengulangan adalah berupa pernyataan dari Java yang mengijinkan kita untuk mengeksekusi blok code berulang-ulang sesuai dengan jumlah tertentu yang diinginkan. Ada tiga macam jenis dari struktur kontrol pengulangan yaitu *while, do-while, dan for-loops*.

6.3.1 while loop

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

Bentuk pernyataan while,

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

Pernyataan di dalam *while loop* 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 Anda memberikan pernyataan yang membuat pengulangan berhenti pada suatu kondisi.

Berikut ini adalah beberapa contoh *while loop*,

Contoh 1:

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

Contoh 2:

```
//infinite loop
while(true)
       System.out.println(“hello”);
```

Contoh 3:

```
//no loops
// statement is not even executed
while (false)
       System.out.println(“hello”);
```

6.3.2 do-while loop

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

Perbedaan antara *while* dan *do-while loop* adalah dimana pernyataan di dalam *do-while loop* akan dieksekusi sedikitnya **satu kali.**

Bentuk pernyataan do-while,

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

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 beberapa contoh do-while loop:

Contoh 1:

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

Contoh ini akan memberikan output 0123456789 pada layar.

Contoh 2:

```
//infinite loop
do{
        System.out.println(“hello”);
} while (true);
```

Contoh di atas akan melakukan pengulangan terus menerus yang menulis kata “hello” pada layar.

Contoh 3:

```
//one loop
// statement is executed once
do
  System.out.println(“hello”);
while (false);
```

Contoh di atas akan memberikan output hello pada layar.

Panduan pemrograman:

1. *Kesalahan pemrograman yang biasa terjadi ketika menggunakan do-while loop adalah lupa untuk menulis titik koma (;) setelah ekspresi while.*

```
do{
        ...
}while(boolean_expression)//-> salah>tidak ada titik koma(;)
```

2. *Seperti pada while loop, pastikan do-while loop anda berhenti pada suatu kondisi.*

6.3.3 for loop

Pernyataan *for loop* memiliki kondisi hampir mirip seperti struktur pengulangan sebelumnya yaitu melakukan pengulangan untuk mengeksekusi kode yang sama sebanyak jumlah yang telah ditentukan.

Bentuk dari for loop,

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

dimana,

InitializationExpression – inisialisasi dari variabel loop.
LoopCondition **-** membandingkan variabel loop pada nilai batas tertentu
. ***StepExpression*** - 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 for loop 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).

Contoh tadi, adalah contoh yang sama dari while loop,

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

6.4 Pernyataan Percabangan

Pernyataan percabangan mengijinkan kita untuk mengatur aliran eksekusi program. Java memberikan tiga bentuk pernyataan percabangan: break, continue dan return.

6.4.1 Pernyataan break

Pernyataan break memiliki dua bentuk: tidak berlabel (*unlabeled*) dan berlabel (*labeled*).

6.4.1.1 Pernyataan break tidak berlabel (unlabeled)

Pernyataan *break* tidak berlabel (*unlabeled*) digunakan untuk menghentikan jalannya pernyataan *switch*. Selain itu pernyataan *break unlabeled* juga bisa digunakan untuk menghentikan pernyataan-pernyataan *for, while* atau *do-while loop*.

Contohnya,

```
String names[] = {"Beah", "Bianca", "Lance", "Belle",
                  "Nico", "Yza", "Gem", "Ethan"};

String          searchName = "Yza";
boolean         foundName = false;

for( int i=0; i< names.length; i++ ){
       if( names[i].equals( searchName )){
                  foundName = true;
                  break;
       }
}

if( foundName ){
       System.out.println( searchName + " found!" );
}
else{
       System.out.println( searchName + " not found." );
}
```

Pada contoh diatas, jika string “Yza” ditemukan, pengulangan pada *for loop* akan dihentikan dan akan dilanjutkan ke pernyataan berikutnya yang terletak setelah pernyataan for.

6.4.1.2 Pernyataan break berlabel

Bentuk label dari pernyataan break akan menghentikan pernyataan diluarnya, dimana sebelumnya harus diberikan label yang sudah di spesifikasikan pada program pada pernyataan *break*. Program berikut ini akan mencari nilai dalam array dua dimensi. Terdapat dua pengulangan bersarang (*nested loop*). Ketika sebuah nilai ditemukan, brea akan menghentikan pernyataan yang diberi label *searchLabel* yang terletak di luar pernyataan for loop.

```
int[][] numbers = {{1, 2, 3},
                   {4, 5, 6},
                   {7, 8, 9}};

int searchNum = 5;
boolean foundNum = false;

searchLabel:
for( int i=0; i<numbers.length; i++ ){
       for( int j=0; j<numbers[i].length; j++ ){
                  if( searchNum == numbers[i][j] ){
                           foundNum = true;
                           break searchLabel;
                  }
       }
}

if( foundNum ){
       System.out.println( searchNum + " found!" );
}
else{
       System.out.println( searchNum + " not found!" );
}
```

Pernyataan break menghentikan pernyataan yang diberi label; dan tidak menjalankan aliran kontrol apapun pada label. Aliran kontrol pada label akan diberikan secara otomatis pada pernyataan yang terletak dibawah label.

6.4.2 Pernyataan *Continue*

Pernyataan *continue* memiliki dua bentuk: berlabel dan tidak berlabel. Anda dapat menggunakan pernyataan *continue* untuk melanjutkan pengulangan yang sedang dijalankan oleh pernyataan *for*, *while*, atau *do- while loop*.

6.4.2.1 Pernyataan continue tidak berlabel (*unlabeled*)

Bentuk pernyataan continue tidak berlabel (*unlabeled*) akan melewati bagian pernyataan setelah pernyataan ini dituliskan dan memeriksa ekspresi logika (*boolean*) yang mengkontrol pengulangan. Jika ekspresi logika (*boolean*) masih bernilai *true*, maka pengulangan tetap dilanjutkan. Pada dasarnya pernyataan ini akan melanjutkan bagian pengulangan pada pernyataan loop.

Berikut ini adalah contoh dari penghitungan angka dari “Beah” dalam suatu array.

```
String names[] = {"Beah", "Bianca", "Lance", "Beah"};
int    count = 0;

for( int i=0; i<names.length; i++ ){

       if( !names[i].equals("Beah") ){
                continue; //skip next statement
       }

       count++;
}

System.out.println("There are " + count + " Beahs in the
list");
```

6.4.2.2 Labeled continue statement

Bentuk pernyataan continue berlabel (*labeled*) akan melanjutkan pengulangan yang sedang terjadi dan dilanjuti ke pengulangan berikutnya dari pernyataan pengulangan yang diberi label (tanda).

```
outerLoop:
for( int i=0; i<5; i++ ){

   for( int j=0; j<5; j++ ){
      System.out.println("Inside for(j) loop"); //message1
      if( j == 2 )       continue outerLoop;
   }

   System.out.println("Inside for(i) loop"); //message2
}
```

Pada contoh ini, bagian message2 tidak pernah akan dicetak, karena pernyataan continue akan melewati pengulangan.

6.4.3 Pernyataan Return

Pernyataan *return* digunakan untuk keluar dari sebuah method. Pernyataan return memiliki dua bentuk: memberikan sebuah nilai, dan tidak memberikan nilai.

Untuk memberikan sebuah nilai, cukup berikan nilai (atau ekspresi yang menghasilkan sebuah nilai) sesudah kata return. Contohnya,

```
return ++count;
```

atau

```
return "Hello";
```

Tipe data dari nilai yang diberikan harus sama dengan tipe dari method yang dibuat. Ketika sebuah method void dideklariskan, gunakan bentuk return yang tidak memberikan nilai. Contohnya,

```
return;
```

Kita akan membahas lebih lanjut tentang pernyataan return ketika mempelajari tentang method.

6.5 Latihan

6.5.1 Nilai

Ambil tiga nilai ujian dari user dan hitung nilai rata-rata dari nilai tersebut. Berikan output rata-rata dari tiga ujian. Berikan juga smiley face pada output jika nilai rata-rata lebih besar atau sama dengan 60, selain itu beri output :-(.

1. Gunakan BufferedReader untuk mendapat input dari user, dan System.out untuk output hasilnya.
2. Gunakan JOptionPane untuk mendapat input dari user dan output hasilnya.

6.5.2 Membaca Bilangan

Ambil sebuah angka sebagai input dari user, dan outputnya berupa kata yang sesuai dengan angka. Angka yang dimasukkan antara 1-10. Jika user memasukkan nilai yang tidak sesuai berikan output “Invalid number”.

1. Gunakan statement if-else untuk menyelesaikan
2. Gunakan statement switch untuk menyelesaikan

6.5.3 Cetak Seratus Kali

Buat sebuah program yang mencetak nama Anda selama seratus kali. Buat tiga versi program ini menggunakan while loop, do while dan for-loop.

6.5.4 Perpangkatan

Hitung pangkat sebuah nilai berdasarkan angka dan nilai pangkatnya. Buat tiga versi dari program ini menggunakan while loop, do-while dan for-loop.