BAB 6

Struktur Kontrol

6.1 Tujuan

Pada bab sebelumnya, kita sudah mendapatkan contah dari program *sequential*, dimana statement dieksekusi setelah statement sebelumnya dengan urutan tertentu. Pada bagian ini, kita mempelajari tentang struktur kontrol yang bertujuan agar kita dapat menentukan urutan statement yang akan dieksekusi.

Pada akhir bab, siswa diharapkan mampu:

- Menggunakan struktur kontrol keputusan (if, else, switch) yang digunakan untuk memilih blok kode yang akan dieksekusi
- Menggunakan struktur kontrol pengulangan (while, do-while, for) yang digunakan untuk melakukan pengulangan pada blok kode yang akan dieksekusi
- Menggunakan statement percabangan (break, continue, return) yang digunakan untuk mengatur *redirection* dari program

6.2 Struktur Kontrol Keputusan

Struktur kontrol keputusan adalah statement dari Java yang mengijinkan user untuk memilih dan mengeksekusi blok kode dan mengabaikan blok kode yang lain.

6.2.1 Statement if

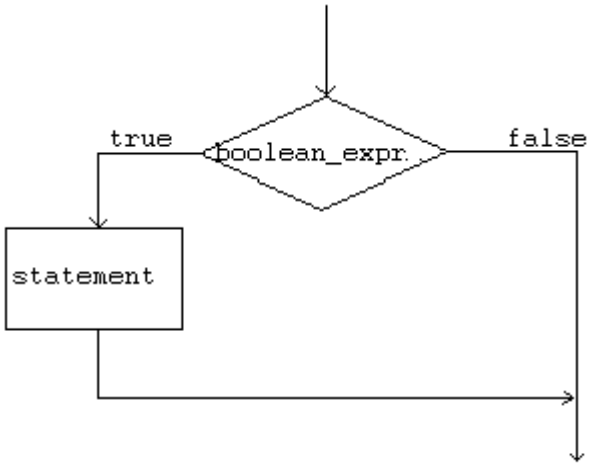
Statement-if menentukan sebuah statement (atau blok kode) yang akan dieksekusi jika dan hanya jika persyaratan boolean (*boolean statement*) bernilai *true.*

Bentuk dari statement if,

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

atau

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



Gambar 1: Flowchart Statement If

dimana, boolean_expression adalah sebuah persyaratan *boolean* (*boolean statement*) atau *boolean* variabel.

Berikut ini adalah contoh code statement 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 statement 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

Statement if-else digunakan apabila kita ingin mengeksekusi sebuah statement dengan kondisi *true* dan statement 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,

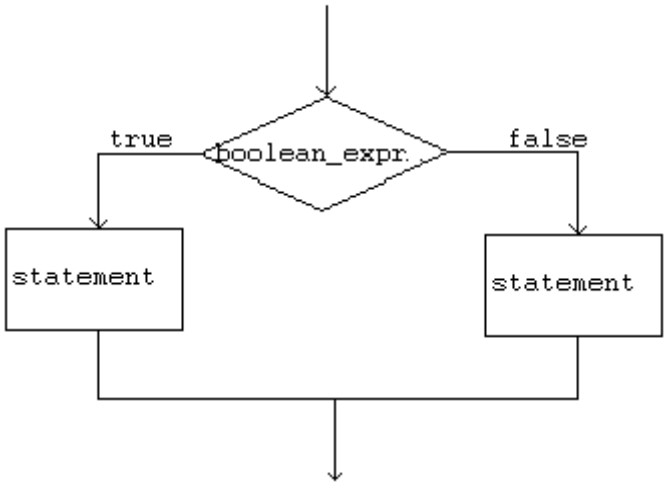
```
int grade = 68;

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 statement di dalam blok if-else di dalam tanda { },*
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{
 . . .
}
```

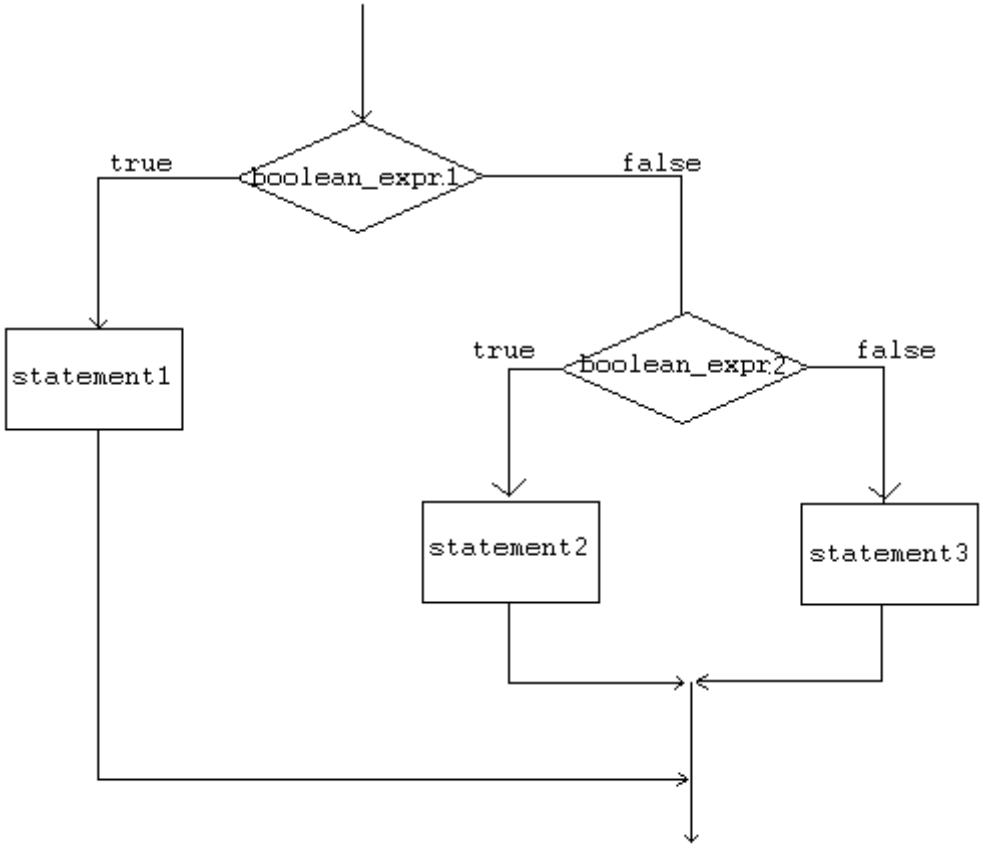
6.2.3 Statement if-else-if

Statement pada bagian else dari blok if-else dapat menjadi struktur if-else yang lain. 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;
```

Bisa anda catat anda dapat memiliki banyak blok else-if sesudah statement if. Blok else bersifat optional dan dapat dihilangkan. Pada contoh di bawah atas, jika boolean_expression1 bernilai true, maka program akan mengeksekusi statement1 dan melewati statement 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 bukan merupakan nilai boolean. Contohnya,

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

Variabel number tidak memiliki nilai Boolean.

2. Using = instead of == for comparison. For example,
3. Menggunakan = daripada == untuk operator perbandingan. Contohnya,

```
//SALAH
int number = 0;
if( number = 0 ){
	//Statement Selanjutnya
}
```

Seharusnya code tersebut ditulis,

```
//BENAR
int number = 0;
if( number == 0 ){
	//Statement Selanjutnya
}
```

3. Menulis **elseif** daripada **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 percabangan adalah dengan menggunakan kata kunci **switch**. Dengan menggunakan switch kita bisa melakukan percabangan dengan persyaratan yang beragam.

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;
}
```

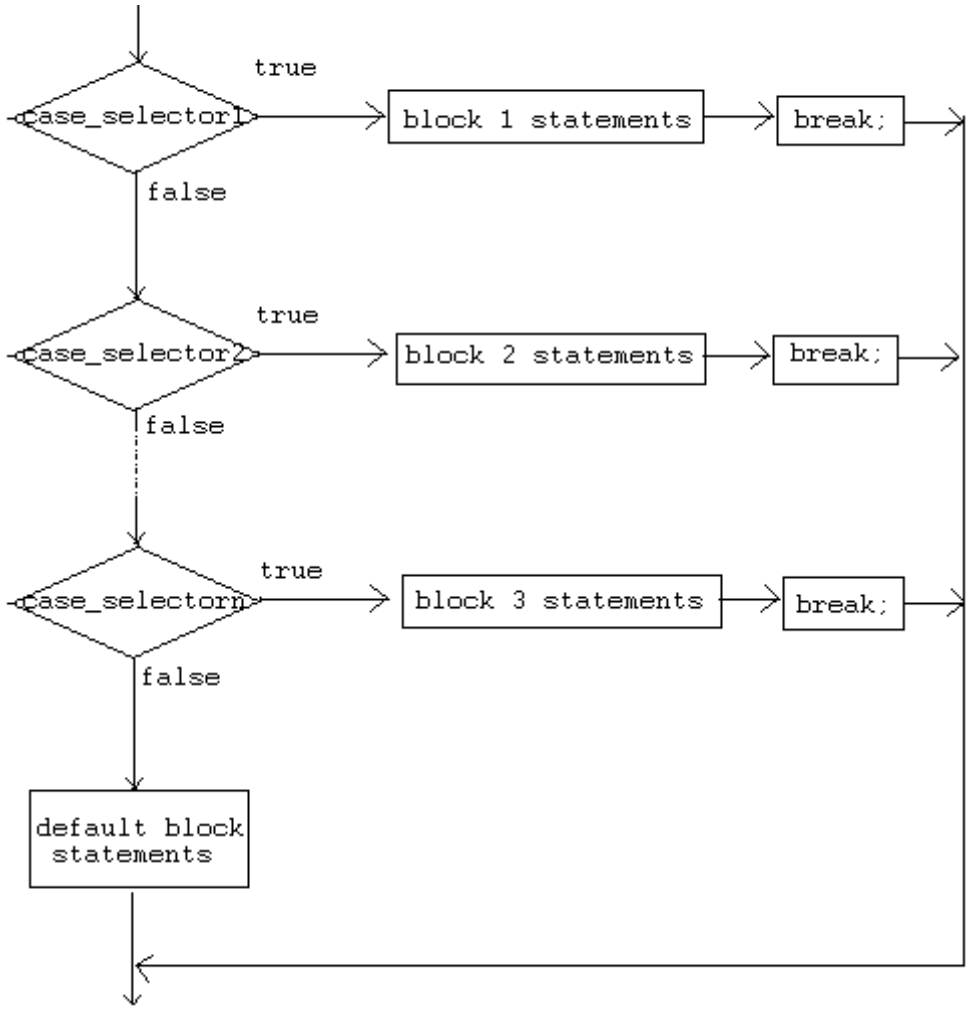
dimana, switch_expression adalah persyaratan **integer** atau **character** dan case_selector1, case_selector2 dan seterusnya adalah konstanta nilai integer yang unique (unik).

Ketika statement switch ditemukan, pertama kali Java memeriksa switch_expression, dan meloncat ke case dan mencocokkan nilai yang sama dengan persyaratannya. Program mengeksekusi statement dari awal sampai menemui statement break, dan melewati statement yang lain sampai akhir struktur switch.

Jika tidak ditemui case yang cocok, maka program akan mengeksekusi blok default. Bisa anda catat bahwa blok default adalah optional. Sebuah statement switch bisa tidak memiliki blok default.

CATATAN:

- Tidak seperti statement **if**, pada struktur switch statement dieksekusi tanpa memerlukan tanda kurung kurawal (**{ }**).
- Ketika sebuah case pada statement switch menemui kecocokan, semua statement pada case tersebut akan dieksekusi. Tidak hanya demikian, statement lain yang berada pada case yang cocok juga dieksekusi.
- Untuk menghindari program mengeksekusi statement pada case berikutnya, kita menggunakan statement **break** sebagai statement akhir.



Gambar 4: Flowchart Statement Switch

Petunjuk Penulisan Program :

1. *Menentukan penggunaan statement if atau statement switch adalah sebuah keputusan. Anda dapat menentukan yang mana yang akan dipakai berdasarkan kemudahan membaca program dan faktor-faktor yang lain.*
2. *Statement if dapat digunakan untuk membuat keputusan berdasarkan rentang nilai tertentu atau kondisi tertentu, sedangkan statement switch membuat keputusan hanya berdasarkan nilai unique (unik) dari integer atau character.*

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 statement dari Java dimana kita bisa mengeksekusi blok code berulang-ulang dalam kurun nilai tertentu. Ada tiga macam jenis struktur kontrol pengulangan yaitu while, do-while, dan for-loops.

6.3.1 while loop

Statement while loop adalah statement atau blok statement yang diulang-ulang sampai mencapai kondisi yang cocok.

Bentuk statement while,

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

Statement di dalam while loop akan dieksekusi berulang-ulang selama *boolean_expression* bernilai true.

Contoh, pada code 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 looping yang tidak berhenti (**infinite loop**). Sehingga, ketika menggunakan while loop atau bentuk pengulangan yang lain, pastikan Anda memberikan statement yang membuat pengulangan berhenti pada suatu titik.

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. Statement di dalam do-while loop akan dieksekusi beberapa kali selama kondisi bernilai true.

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

Bentuk statement do-while,

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

Statement di dalam do-while loop akan dieksekusi pertama kali, dan dilakukan pengecekan kondisi dari boolean_expression. Jika nilai tersebut belum mencapai nilai yang diinginkan, statement 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 yang tidak berhenti untuk menulis “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 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 titik.*

6.3.3 for loop

Seperti pada struktur pengulangan sebelumnya yaitu melakukan pengulangan eksekusi code beberapa kali.

Bentuk dari for loop,

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

dimana,

InitializationExpression – inisialisasi dari variabel loop.
LoopCondition **-** membandingkan variabel loop pada nilai batas.
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, statement i=0 merupakan inisialisasi dari variabel. Selanjutnya, kondisi i<10 diperiksa. Jika kondisi bernilai true, statement di dalam for loop dieksekusi. Kemudian, statement i++ dieksekusi, dan dilakukan pengecekan kondisi. 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 Branching Statements

Branching statements mengijinkan kita untuk mengatur jalannya eksekusi program. Java memberikan tiga bentuk branching statements: break, continue dan return.

6.4.1 break statement

Statement break memiliki dua bentuk: unlabeled dan labeled.

6.4.1.1 Unlabeled break statement

Unlabeled menghentikan jalannya statement switch. Anda bisa juga menggunakan bentuk unlabeled untuk menghentikan 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 ini, jika string “Yza” ditemukan, pengulangan pada for loop akan dihentikan dan akan melanjutkan ke proses berikutnya.

6.4.1.2 Labeled break statement

Bentuk labeled form dari statement break akan menghentikan statement luar, dimana diidentifikasikan berupa label pada statement break. Program berikut ini akan mencari nilai dalam array dua dimensi. Terdapat dua pengulangan bersarang (nested loop). Ketika sebuah nilai ditemukan, labeled break akan menghentikan statement yang diberi label searchLabel, dimana label ini berada di luar.

```
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!" );
}
```

Statement break menghentikan sementera labeled statement; ia tidak lagi menjalankan flow control pada label. Flow control pada label akan di-transfer secara otomatis mengikuti labeled statement.

6.4.2 Continue statement

Statement continue memiliki dua bentuk: unlabeled dan labeled. Anda dapat menggunakan statement continue untuk melewati pengulangan dari for, while, atau do-while loop yang sedang berjalan.

6.4.2.1 Unlabeled continue statement

Bentuk unlabeled akan melewati akhir statement pada bagian yang dalam dan memeriksa boolean expression yang mengkontrol loop, pada dasarnya akan melewati bagian pengulangan pada 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 labeled akan melanjutkan sebuah statement dengan melewati pengulangan yang sedang berjalan dari loop terluar 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, pesan ke-2 tidak dicetak, karena statement continue akan melewati pengulangan yang sedang berjalan.

6.4.3 Return statement

Statement return digunakan untuk keluar dari sebuah fungsi (method). Statement return memiliki dua bentuk: menggunakan sebuah nilai, dan tidak memberikan nilai.

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

```
return ++count;
```

atau

```
return "Hello";
```

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

```
return;
```

Kita akan membahas lebih lanjut tentang statement return ketika mempelajari tentang fungsi.

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.