


# Bab 4. Struktur Keputusan

## OBJEKTIF :

1. Mahasiswa mampu memahami struktur kontrol dengan menggunakan *statement if*, *statement if-else*, *statement if Tersarang*, *statement if-else-if*, operator logis, dan *statement switch*.
2. Mahasiswa mampu mempelajari perbandingan antara karakter, *floating point*, dan *string*.
3. Mahasiswa mampu menggunakan *software IDE NetBeans* untuk membuat program dengan menggunakan *statement if*, *statement if-else*, *statment if Tersarang*, *statement if-else-if*, operator logis, dan *statement switch*.

## 4.1 Statement *if*

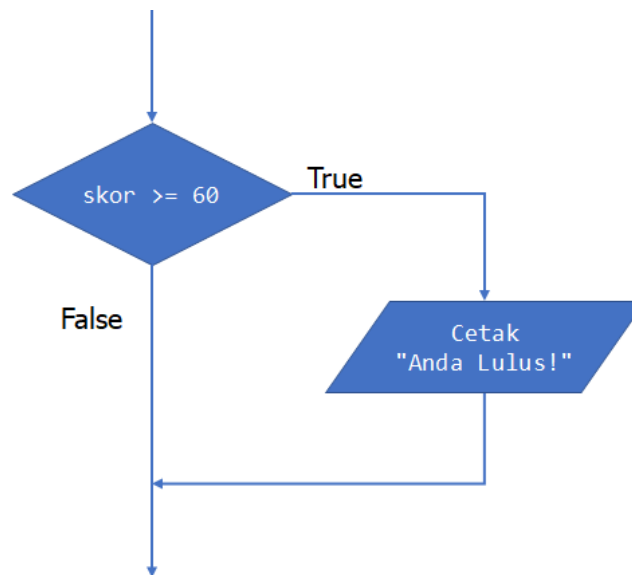
Program-program yang kita tulis sebelumnya berjalan secara sekuensial, yaitu *statement-statement* dieksekusi berurut, satu per satu dari *statement* pertama hingga terakhir. Program dengan alur eksekusi berurut ini disebut sebagai program dengan struktur sekuensial. Gambar berikut mengilustrasikan langkah-langkah eksekusi program dengan struktur sekuensial.



```
public class LuasPersegiPanjang
{
    public static void main(String[] args)
    {
        Langkah 1 → double panjang, lebar, luas;
        Langkah 2 → panjang = 10;
        Langkah 3 → lebar = 5;
        Langkah 4 → luas = panjang * lebar;
        Langkah 5 → System.out.print("Luas adalah " + luas);
    }
}
```

Struktur sekuensial tidak cukup untuk mengimplementasikan sebagian besar algoritma. Banyak algoritma yang membutuhkan program mengeksekusi sekelompok *statement* ketika suatu kondisi terpenuhi. Ini dapat dilakukan dengan struktur keputusan. Disebut struktur keputusan, karena program membuat keputusan apakah mengeksekusi suatu kelompok *statement* atau tidak berdasarkan terpenuhi atau tidaknya suatu kondisi.

Contoh sederhana dari struktur keputusan adalah program yang menampilkan teks "Anda lulus" hanya ketika skor lebih besar atau sama dengan 60 ( $\text{skor} \geq 60$ ). Struktur keputusan dari program ini mempunyai *flowchart* seperti gambar berikut:



Pada *flowchart* di atas, bentuk belah ketupat menandakan kondisi berupa pertanyaan: apakah skor lebih besar atau sama dengan 60? Kondisi ini dapat bernilai *True* (benar) atau *False* (salah). Jika kondisi bernilai *True*, maka program bercabang ke kanan dan menampilkan `Anda Lulus!` dan jika kondisi bernilai *False*, maka program melanjutkan eksekusi ke *statement* berikutnya.

Untuk menuliskan program dengan struktur keputusan seperti *flowchart* di atas kita menggunakan *statement* `if` seperti berikut:

```

if (skor >= 60)
{
    system.out.println("Anda Lulus!");
}
  
```

Pada contoh *statement* `if` di atas, `skor >= 60` adalah sebuah kondisi. Kondisi ini diuji dengan membandingkan nilai yang disimpan dalam variabel `skor`. Jika nilai `skor` lebih besar atau sama dengan 60 maka program mengeksekusi *statement* `system.out.println("Anda Lulus!")` dan jika nilai `skor` kurang daripada 60 maka program melewati *statement* tersebut.

Kondisi yang dituliskan di dalam tanda kurung setelah keyword `if` (dalam contoh di atas: `skor >= 60`) adalah ekspresi *Boolean*. Ekspresi *Boolean* adalah ekspresi yang dievaluasi ke nilai `true` atau `false`.

*statement* `if` mempunyai format penulisan seperti terlihat pada gambar berikut:

```

if (EkspresiBoolean)
{
    statement
    statement
    ...
}
  
```

Kondisi yang berupa ekspresi Boolean (ekspresi yang dievaluasi ke nilai `true` atau `false`).

Kita dapat menuliskan lebih dari satu *statement* dalam body *statement* `if`. *Statement*-*statement* ini dieksekusi ketika *EkspresiBoolean* dievaluasi ke `true`.

## Operator Relasional

Salah satu cara untuk menuliskan ekspresi *Boolean* adalah dengan menggunakan operator relasional. Operator relasional menentukan apakah terdapat suatu relasi tertentu diantara dua nilai. Sebagai contoh, operator relasional lebih besar ( $>$ ) menentukan apakah nilai di ruas kirinya lebih besar daripada nilai di ruas kananya. Tabel berikut mendaftar operator-operator relasional yang terdapat dalam Java.

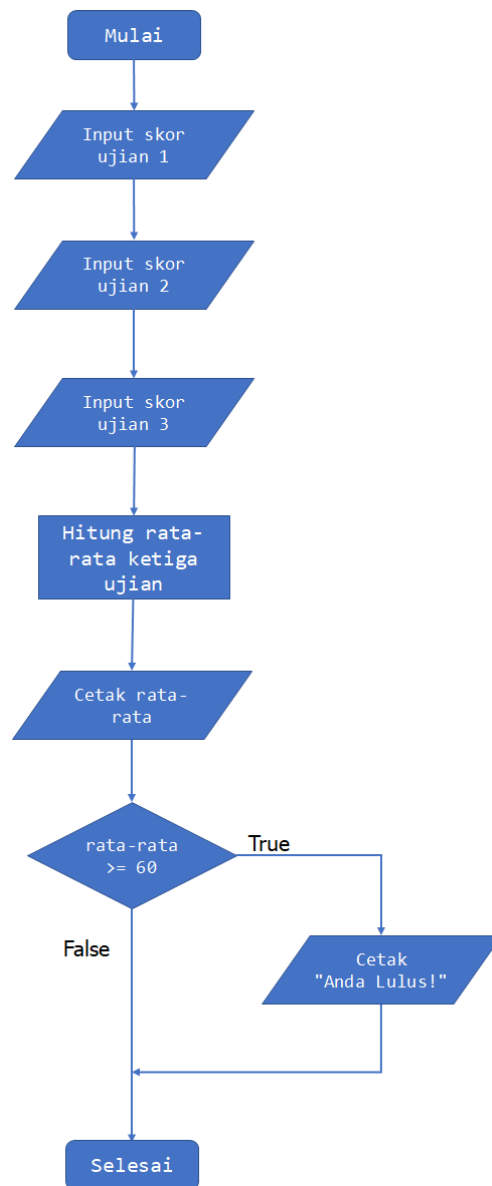
Operator Relasional	Contoh	Arti
$<$	<code>x &lt; y</code>	Apakah <code>x</code> lebih kecil dari <code>y</code> ?
$<=$	<code>x &gt;= y</code>	Apakah <code>x</code> lebih kecil dari atau sama dengan <code>y</code> ?
$>$	<code>x &gt; y</code>	Apakah <code>x</code> lebih besar dari <code>y</code> ?
$>=$	<code>x &gt;= y</code>	Apakah <code>x</code> lebih besar atau sama dengan <code>y</code> ?
$==$	<code>x == y</code>	Apakah <code>x</code> sama dengan <code>y</code> ?
$!=$	<code>x != y</code>	Apakah <code>x</code> tidak sama dengan <code>y</code> ?

**Catatan.** Perhatikan simbol `==` yang digunakan untuk membandingkan apakah nilai kedua operand sama. Kesalahan yang sering dilakukan programmer pemula adalah menggunakan `=` untuk menuliskan ekspresi *Boolean* yang seharusnya menggunakan `==`.

Perlu diperhatikan bahwa semua operator relasional memerlukan dua operand yang dituliskan di ruas kiri dan ruas kanan operator.

## Contoh Program dengan *Statement if*

Misalkan kita menulis sebuah program yang meminta pengguna memasukkan tiga nilai ujian dan program menghitung rata-rata dari ketiga nilai ujian tersebut. Jika rata-rata lebih besar dari 60, program menampilkan pesan "Anda lulus!". Program yang akan kita tuliskan mempunyai *flowchart* seperti berikut:



Kita dapat menuliskan program dengan *flowchart* di atas seperti berikut:

#### **Program (KelulusanUjian.java)**

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan statement if.
*/
public class KelulusanUjian
{
    public static void main(String[] args)
    {
        double skor1;      // Untuk menyimpan skor ujian ke-1
        double skor2;      // Untuk menyimpan skor ujian ke-2
        double skor3;      // Untuk menyimpan skor ujian ke-3
        double rerata;      // Untuk menyimpan rata-rata skor ujian

        Scanner keyboard = new Scanner(System.in);    // Buat object Scanner

        // Minta skor 1
        System.out.print("Masukkan skor ujian 1: ");
        skor1 = keyboard.nextDouble();
```

```

// Minta skor 2
System.out.print("Masukkan skor ujian 2: ");
skor2 = keyboard.nextDouble();

// Minta skor 3
System.out.print("Masukkan skor ujian 3: ");
skor3 = keyboard.nextDouble();

rerata = (skor1 + skor2 + skor3) / 3; // Hitung rata-rata tiga skor

// Tampilkan rata-rata ujian
System.out.printf("Rata-rata Ujian Anda: %.2f\n", rerata);

// Jika skor lebih besar dari 60, tampilkan pesan bahwa pengguna lulus
if (rerata > 60)
{
    System.out.println("Selamat!");
    System.out.println("Anda Lulus!");
}
}
}

```

#### Output Program (KelulusanUjian.java)

```

Masukkan skor ujian 1: 87.5
Masukkan skor ujian 2: 82.33
Masukkan skor ujian 3: 74.75
Rata-rata Ujian Anda: 81.53
Selamat!
Anda Lulus!

```

## Operator Aritmatika dan Operator Relasional

Kita dapat menuliskan ekspresi aritmatika pada ruas kanan dan kiri dari operator relasional, misalkan seperti berikut:

$$x + 2 > y * 4 / 2$$

Ekspresi di atas dievaluasi dengan mengevaluasi masing-masing ekspresi aritmatika pada ruas kanan, `x + 5` dan ruas kiri, `2 * y`, terlebih dahulu lalu kedua nilainya diuji. Misalkan, `x` menyimpan nilai 4 dan `y` menyimpan nilai 2, maka ekspresi di atas akan menguji `9 > 2` yang dievaluasi ke `true`.

Sebagai contoh penggunaan operator aritmatika dan operator relasional pada ekspresi *Boolean* dalam *statement if*, kita dapat mengganti kondisi pada contoh program `KelulusanUjian.java` sebelumnya dengan menghitung rata-rata ujian dalam ekspresi *Boolean* seperti terlihat pada kode berikut:

```

if ((skor1 + skor2 + skor3) / 3 >= 60)
{
    System.out.println("Selamat!");
    System.out.println("Anda Lulus!");
}

```

## 4.2 statement `if-else`

statement `if` dapat diekstensi dengan menambahkan klausa `else` untuk membuat program mengeksekusi sekelompok statement lain ketika kondisi tidak terpenuhi. statement `if` dengan klausa `else` ini disebut dengan statement `if-else`.

statement `if-else` mempunyai format penulisan seperti berikut:

```
if (EkspresiBoolean)
{
    statement
    statement
    ...
}
else
{
    statement
    statement
    ...
}
```

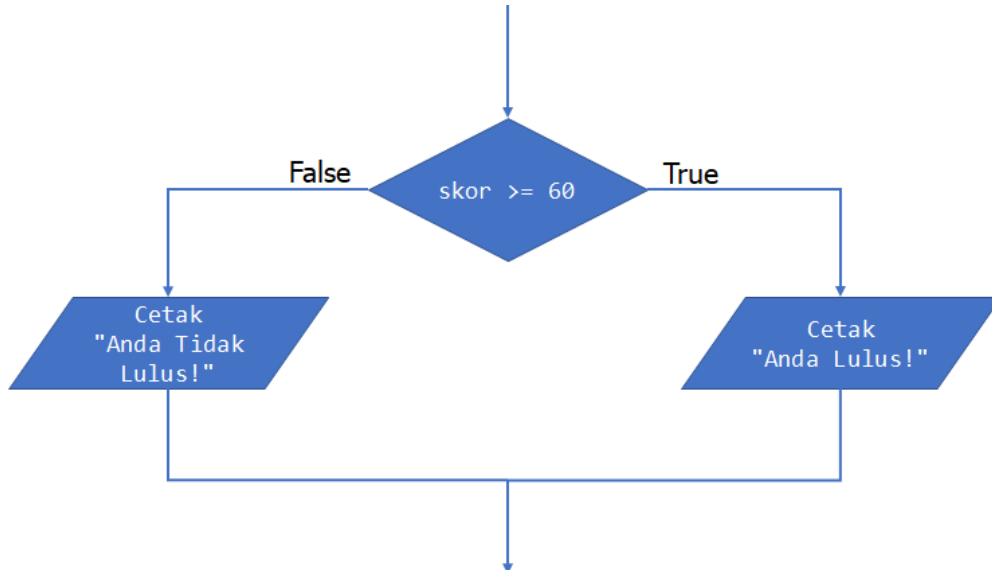
Kondisi yang berupa ekspresi Boolean (ekspresi yang dievaluasi ke nilai true atau false).

Statement-statement ini dieksekusi ketika *EkspresiBoolean* dievaluasi ke true.

Statement-statement ini dieksekusi ketika *EkspresiBoolean* dievaluasi ke false.

Sebagai contoh, misalkan kita mengembangkan program contoh kita pada subbab sebelumnya, yang sebelumnya menampilkan pesan lulus saat kondisi skor  $\geq 60$ , sekarang program juga menampilkan pesan tidak lulus ketika kondisi skor  $< 60$ .

Program kita sekarang mempunyai logika seperti pada *flowchart* berikut:



Kita dapat menggunakan statement `if-else` seperti berikut untuk mengimplementasikan *flowchart* di atas:

```
if (skor >= 60)
{
    System.out.println();
}
else
{
    System.out.println();
}
```

Program `kelulusanUjian.java` dapat kita tulis ulang menggunakan *statement if-else* menjadi seperti berikut:

**Program (KelulusanUjian2.java)**

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan statement if-else.
*/
public class KelulusanUjian2
{
    public static void main(String[] args)
    {
        double skor1;        // Untuk menyimpan skor ujian ke-1
        double skor2;        // Untuk menyimpan skor ujian ke-2
        double skor3;        // Untuk menyimpan skor ujian ke-3
        double rerata;        // Untuk menyimpan rata-rata skor ujian

        Scanner keyboard = new Scanner(System.in);    // Buat object Scanner

        // Minta skor 1
        System.out.print("Masukkan skor ujian 1: ");
        skor1 = keyboard.nextDouble();

        // Minta skor 2
        System.out.print("Masukkan skor ujian 2: ");
        skor2 = keyboard.nextDouble();

        // Minta skor 3
        System.out.print("Masukkan skor ujian 3: ");
        skor3 = keyboard.nextDouble();

        rerata = (skor1 + skor2 + skor3) / 3; // Hitung rata-rata tiga skor

        // Tampilkan rata-rata ujian
        System.out.printf("Rata-rata Ujian Anda: %.2f\n", rerata);

        // Jika skor lebih besar dari 60, tampilkan pesan bahwa pengguna lulus
        if (rerata > 60)
        {
            System.out.println("Selamat!");
            System.out.println("Anda Lulus!");
        }
        else
        {
            System.out.println("Anda Tidak Lulus!");
            System.out.println("Silahkan mengulang semester depan.");
        }
    }
}
```

**Output Program (KelulusanUjian2.java)**

Input 1

```
Masukkan skor ujian 1: 66.5
Masukkan skor ujian 2: 50.4
Masukkan skor ujian 3: 55.3
Rata-rata Ujian Anda: 57.40
Anda Tidak Lulus!
Silahkan mengulang semester depan.
```

#### Input 2

```
Masukkan skor ujian 1: 98.75
Masukkan skor ujian 2: 76.5
Masukkan skor ujian 3: 70.5
Rata-rata Ujian Anda: 81.92
Selamat!
Anda Lulus!
```

## Contoh Lain *Statement* `if-else`

Salah satu kegunaan *statement* `if-else` yang sering digunakan dalam program adalah untuk mengvalidasi input pengguna untuk mencegah program *crash* (berhenti tiba-tiba). Misalkan, program yang melakukan pembagian dua angka akan *crash* ketika kita melakukan pembagian dengan 0. Kita dapat mencegah program *crash* dengan menuliskan *statement* `if-else` yang mencegah kalkulasi pembagian ketika pengguna memasukkan angka pembagi 0, seperti yang dicontohkan pada program berikut:

#### **Program (Pembagian.java)**

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan statement if-else
    untuk validasi input pengguna.
*/
public class Pembagian
{
    public static void main(String[] args)
    {
        double num1, num2;    // Operand pembagian
        double hasilBagi;     // Hasil pembagian

        // Buat sebuah object Scanner untuk input keyboard
        Scanner keyboard = new Scanner(System.in);

        // Minta angka pertama (terbagi)
        System.out.print("Masukkan angka terbagi: ");
        num1 = keyboard.nextDouble();

        // Minta angka kedua (pembagi)
        System.out.print("Masukkan angka pembagi: ");
        num2 = keyboard.nextDouble();

        // validasi angka pembagi tidak sama nol dan tampilkan pesan
        // program tidak dapat melakukan pembagian dengan nol
        if (num2 == 0)
        {
```



```

        System.out.println("Pembagian dengan nol tidak mungkin.");
        System.out.println("Silahkan jalankan kembali program dan ");
        System.out.println("masukkan angka pembagi selain nol.");
    }
    else
    {
        hasilBagi = num1 / num2;
        System.out.print("Angka " + num1);
        System.out.print(" dibagi dengan " + num2);
        System.out.println(" menghasilkan " + hasilBagi);
    }
}
}

```

### Output Program (Pembagian.java)

#### Input 1

```

Masukkan angka terbagi: 6
Masukkan angka pembagi: 0
Pembagian dengan nol tidak mungkin.
Silahkan jalankan kembali program dan
masukkan angka pembagi selain nol.

```

#### Input 2

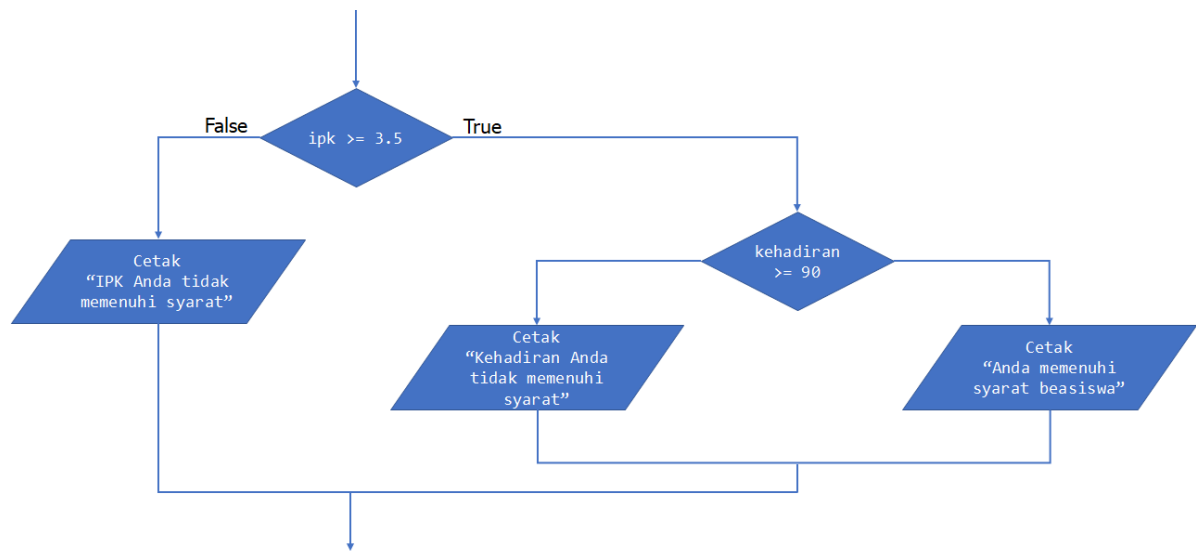
```

Masukkan angka terbagi: 6
Masukkan angka pembagi: 2
Angka 6.0 dibagi dengan 2.0 menghasilkan 3.0

```

## 4.3 Statement `if` Tersarang

Kita dapat menuliskan *statement* `if` di dalam *statement* `if` lainnya. *statement* `if` di dalam *statement* `if` ini disebut dengan *statement* `if` tersarang (nested `if`). Contoh persoalan yang membutuhkan *statement* `if` tersarang adalah sebagai berikut: misalkan kita menulis sebuah program yang menentukan apakah seorang mahasiswa terkualifikasi sebagai penerima beasiswa atau tidak. Untuk terkualifikasi sebagai penerima beasiswa, mahasiswa harus mempunyai IPK lebih besar atau sama dengan 3.5 dan kehadiran kelas lebih besar dari 90 persen. Kita merancang program tersebut dengan *flowchart* berikut:



Flowchart di atas dapat dituliskan dengan `statement if` tersarang seperti berikut:

```
if (ipk >= 3.5)
{
    if (kehadiran >= 90)
    {
        System.out.println("Anda memenuhi syarat beasiswa.");
    }
    else
    {
        System.out.println("Kehadiran Anda tidak memenuhi syarat.");
    }
}
else
{
    System.out.println("IPK Anda tidak memenuhi syarat.");
}
```

Kode program lengkap kualifikasi beasiswa ini dapat dituliskan seperti berikut:

#### **Program (Beasiswa.java)**

```
/*
    Program ini mendemonstrasikan statement if tersarang
*/

import java.util.Scanner;

public class Beasiswa
{
    public static void main(String[] args)
    {
        double ipk;
        int kehadiran;

        Scanner keyboard = new Scanner(System.in);

        // Minta nilai ipk
        System.out.print("Masukkan IPK: ");
        ipk = keyboard.nextDouble();
    }
}
```

```

// Minta kehadiran
System.out.print("Masukkan kehadiran (persen): ");
kehadiran = keyboard.nextInt();

// statement if tersarang untuk menentukan apakah
// mahasiswa memenuhi syarat beasiswa
if (ipk >= 3.5)
{
    if (kehadiran >= 90)
    {
        System.out.println("Anda memenuhi syarat beasiswa.");
    }
    else
    {
        System.out.println("Kehadiran Anda tidak memenuhi syarat.");
    }
}
else
{
    System.out.println("IPK Anda tidak memenuhi syarat.");
}
}
}

```

### Output Program (Beasiswa.java)

#### Input 1

```

Masukkan IPK: 3.4
Masukkan kehadiran (persen): 70
IPK Anda tidak memenuhi syarat.

```

#### Input 2

```

Masukkan IPK: 3.6
Masukkan kehadiran (persen): 80
Kehadiran Anda tidak memenuhi syarat.

```

#### Input 3

```

Masukkan IPK: 3.9
Masukkan kehadiran (persen): 95
Anda memenuhi syarat beasiswa.

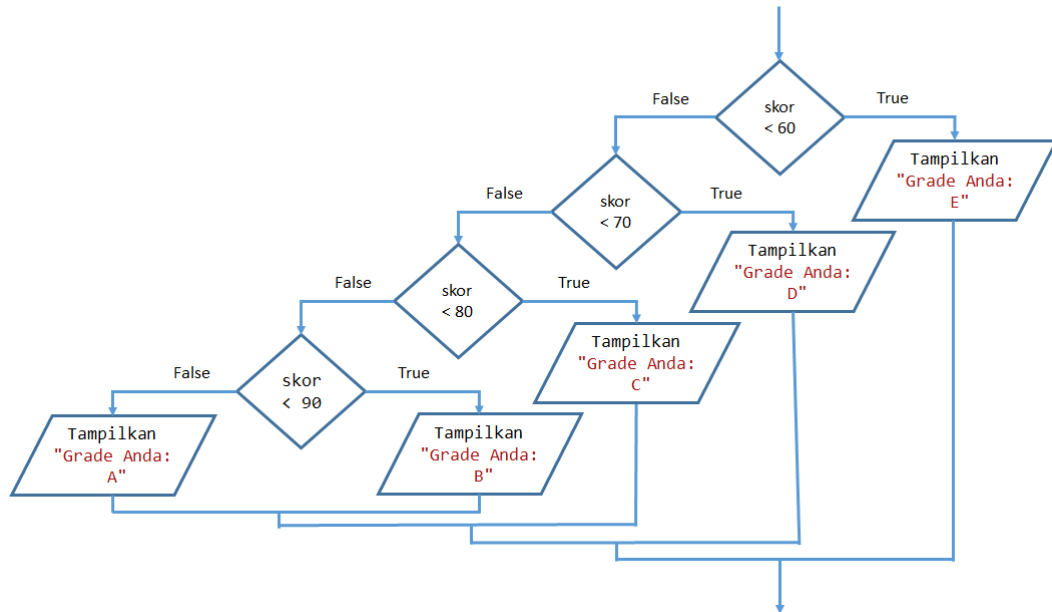
```

## Contoh Lain **Statement if** Tersarang

Misalkan, Anda ditugaskan untuk membuat sebuah program yang mengkonversi nilai ujian menjadi nilai huruf menurut skala pada tabel berikut:

Skor Ujian	Grade
> 90	A
80 - 89	B
70 - 79	C
60 - 69	D
< 60	E

Flowchart dari program dapat kita gambarkan seperti pada gambar berikut:



Program yang mengkonversi nilai ujian menjadi nilai huruf ini dapat dituliskan seperti berikut:

#### Program (GradeHuruf.java)

```

/*
    Program ini meminta pengguna untuk memasukkan skor ujian
    dan menampilkan grade huruf untuk skor tersebut. Program
    ini menggunakan struktur keputusan tersarang untuk menentukan
    grade huruf.
*/

import java.util.Scanner;

public class GradeHuruf
{
    public static void main(String[] args)
    {
        double skor;

        Scanner keyboard = new Scanner(System.in);

        // Minta skor ujian
        System.out.print("Masukkan nilai ujian: ");
        skor = keyboard.nextDouble();

        // Tampilkan nilai huruf
    }
}
  
```

```

    if (skor < 60)
    {
        System.out.println("Grade Anda: E");
    }
    else
    {
        if (skor < 70)
        {
            System.out.println("Grade Anda: D");
        }
        else
        {
            if (skor < 80)
            {
                System.out.println("Grade Anda: C");
            }
            else
            {
                if (skor < 90)
                {
                    System.out.println("Grade Anda: B");
                }
                else
                {
                    System.out.println("Grade Anda: A");
                }
            }
        }
    }
}
}
}
}
}

```

#### Output Program (GradeHuruf.java)

##### Input 1

```

Masukkan nilai ujian: 91
Grade Anda: A

```

##### Input 2

```

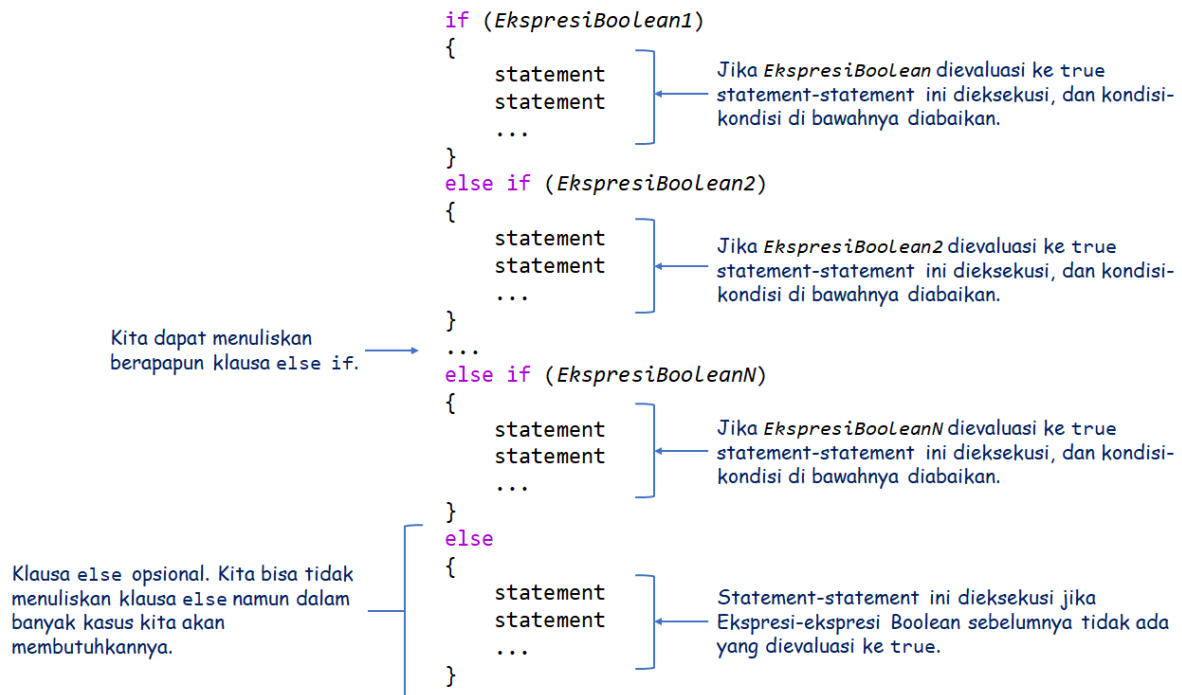
Masukkan nilai ujian: 59
Grade Anda: E

```

## 4.4 Statement `if-else-if`

Di bagian sebelumnya kita dapat menggunakan *statement* `if` tersarang untuk menguji rangkaian kondisi-kondisi. Penulisan *statement* `if` tersarang tersebut membuat kode sulit dibaca. Untungnya, Java menyediakan *statement* `if-else-if` yang dapat digunakan untuk menyederhanakan penulisan `if` tersarang yang menguji rangkaian kondisi-kondisi.

Format penulisan *statement* `if-else-if` adalah seperti berikut:



Ketika `statement if-else-if` dieksekusi, pertama-tama program menguji `EkspresiBoolean1`. Jika `EkspresiBoolean1` menghasilkan `true`, maka `statement-statement` yang berada dalam tanda kurung kurawal setelahnya dieksekusi dan program mengabaikan kondisi-kondisi di bawahnya. Namun, jika `EkspresiBoolean1` menghasilkan `false`, program ke klausa `else if` berikutnya dan menguji `EkspresiBoolean2`. Jika `EkspresiBoolean2` menghasilkan `true`, program mengeksekusi `statement-statement` dalam kurung kurawal setelahnya dan mengabaikan kondisi-kondisi di bawahnya. Proses ini berlanjut, dari kondisi paling atas sampai bawah, sampai dengan program menemukan ekspresi `Boolean` yang menghasilkan `true`. Jika tidak ada ekspresi `Boolean` yang menghasilkan `true`, `statement-statement` yang berada dalam klausa `else` dieksekusi.

Klausa `else` terakhir, yang tidak mempunyai `keyword if` dan tidak mempunyai kondisi adalah opsional. Kita bisa tidak menuliskan klausa `else` ini, namun dalam banyak kasus kita akan membutuhkannya.

Program berikut menggunakan `statement if-else-if` untuk menulis ulang program `GradeHuruf.java`:

### Program (GradeHuruf2.java)

```
import java.util.Scanner;

/*
    Program ini meminta pengguna untuk memasukkan skor ujian
    dan menampilkan grade huruf untuk skor tersebut. Program
    ini menggunakan statement if-else-if untuk menentukan
    grade huruf.
*/
public class GradeHuruf2
{
    public static void main(String[] args)
    {
        double skor;

        Scanner keyboard = new Scanner(System.in);

        // Minta skor ujian
    }
}
```

```

        System.out.print("Masukkan nilai ujian: ");
        skor = keyboard.nextDouble();

        // Tampilkan nilai huruf
        if (skor < 60)
        {
            System.out.println("Grade Anda: E");
        }
        else if (skor < 70)
        {
            System.out.println("Grade Anda: D");
        }
        else if (skor < 80)
        {
            System.out.println("Grade Anda: C");
        }
        else if (skor < 90)
        {
            System.out.println("Grade Anda: B");
        }
        else
        {
            System.out.println("Grade Anda: A");
        }
    }
}

```

### Output Program (GradeHuruf2.java)

#### Input 1

```

Masukkan nilai ujian: 87
Grade Anda: B

```

#### Input 2

```

Masukkan nilai ujian: 61
Grade Anda: D

```

Hal yang perlu diperhatikan ketika kita menggunakan statement `if-else-if` adalah program memulai pengujian kondisi dari `EksresiBoolean1` dan melanjutkan pengujian ke `Ekspresi Boolean` selanjutnya, namun ketika program menemukan kondisi yang `true` pada klausa `if` atau salah satu klausa `else if` maka program tidak melanjutkan pengujian ke kondisi-kondisi selanjutnya.

## 4.5 Operator Logis

Java menyediakan dua operator logis, `&&` dan `||` yang digunakan untuk mengkombinasi dua ekspresi *Boolean* menjadi satu ekspresi. Selain kedua operator logis tersebut, Java juga mempunyai operator `!` yang membalik nilai kebenaran dari sebuah ekspresi *Boolean*. Tabel berikut menjelaskan ketiga operator tersebut.

Operator	Arti	Contoh Ekspresi	Arti Ekspresi
<code>&amp;&amp;</code>	Operator <b>AND</b> . Mengkombinasikan dua ekspresi Boolean menjadi satu gabungan ekspresi. Kedua subekspresi harus <code>true</code> agar gabungan ekspresi <code>true</code> .	<code>x &gt; y &amp;&amp; a &lt; b</code>	Apakah x lebih besar dari y DAN apakah a lebih kecil dari b?
<code>  </code>	Operator <b>OR</b> . Mengkombinasikan dua ekspresi Boolean menjadi satu gabungan ekspresi. Salah satu atau kedua subekspresi harus <code>true</code> agar gabungan ekspresi <code>true</code> .	<code>x == y    a == b</code>	Apakah x sama dengan y ATAU apakah a sama dengan b?
<code>!</code>	Operator <b>NOT</b> . Merupakan operator unary (membutuhkan hanya satu operand). Operator ini membalik nilai kebenaran dari operand. Jika operand bernilai <code>true</code> , maka operator ini menjadikannya bernilai <code>false</code> , dan sebaliknya.	<code>!(x &gt; y)</code>	Apakah ekspresi <code>x &gt; y</code> TIDAK benar?

### Operator `&&`

Operator `&&` disebut sebagai operator logis AND. Operator ini mengkombinasi dua ekspresi *Boolean* menjadi sebuah ekspresi *Boolean* yang hanya `true` ketika kedua subekspresi *Boolean* bernilai `true`. Berikut adalah contoh *statement if* yang menggunakan operator `&&`:

```
if (temperatur < 20 && menit > 12)
{
    System.out.println("Temperatur dalam zona berbahaya.");
}
```

Dalam *statement* ini dua ekspresi *Boolean*, `temperatur < 20` dan `menit > 12` dikombinasi menjadi sebuah ekspresi. Sebuah pesan akan ditampilkan jika `temperatur` kurang dari 20 **DAN** `menit` lebih besar dari 12. Jika salah satu atau kedua ekspresi *Boolean* bernilai `false`, maka keseluruhan ekspresi adalah `false` dan pesan tidak ditampilkan.

Tabel berikut adalah tabel kebenaran untuk operator `&&`. Tabel ini mendaftar semua kemungkinan kombinasi nilai dua ekspresi yang dihubungkan dengan operator `&&`.

Ekspresi	Nilai Ekspresi
<code>true &amp;&amp; false</code>	<code>false</code>
<code>false &amp;&amp; true</code>	<code>false</code>
<code>false &amp;&amp; false</code>	<code>false</code>
<code>true &amp;&amp; true</code>	<code>true</code>

Seperti terlihat pada tabel, kedua ekspresi yang dikombinasi oleh operator `&&` harus bernilai `true` untuk kombinasi ekspresi dievaluasi ke `true`.



Operator `&&` dapat digunakan untuk menyederhanakan program yang menggunakan *statement if* tersarang. Program berikut adalah versi berbeda dari program `Beasiswa.java` yang ditulis menggunakan operator `&&`.

#### Program (*BeasiswaAnd.java*)

```
import java.util.Scanner;
/*
    Program ini menentukan apakah seorang mahasiswa
    terqualifikasi untuk menerima beasiswa berdasarkan
    ipk dan kehadiran. Program ini menggunakan operator
    logis && untuk kondisi kualifikasi.
*/
public class BeasiswaAnd
{
    public static void main(String[] args)
    {
        double ipk;
        int kehadiran;

        Scanner keyboard = new Scanner(System.in);

        // Minta nilai ipk
        System.out.print("Masukkan IPK: ");
        ipk = keyboard.nextDouble();

        // Minta kehadiran
        System.out.print("Masukkan kehadiran (persen): ");
        kehadiran = keyboard.nextInt();

        // statement if tersarang untuk menentukan apakah
        // mahasiswa memenuhi syarat beasiswa
        if (ipk >= 3.5 && kehadiran >= 90)
        {
            System.out.println("Anda memenuhi syarat beasiswa.");
        }
        else
        {
            System.out.println("Anda tidak memenuhi syarat.");
        }
    }
}
```

#### Output Program (*BeasiswaAnd.java*)

```
Masukkan IPK: 3.55
Masukkan kehadiran (persen): 91
Anda memenuhi syarat beasiswa.
```

## Operator `||`

Operator `||` disebut sebagai operator logis OR. Operator ini mengkombinasi dua ekspresi *Boolean* menjadi sebuah ekspresi *Boolean* yang dievaluasi ke `true` ketika salah satu atau kedua subekspresi *Boolean* bernilai `true`. Berikut adalah contoh *statement if* yang menggunakan operator `||`:

```

if (temperatur < 20 || temperatur > 100)
{
    System.out.println("Temperatur dalam zona berbahaya");
}

```

Kode di atas akan menampilkan pesan jika kondisi `temperatur` kurang dari 20 **ATAU** kondisi `temperatur` lebih dari 100. Jika salah satu atau kedua kondisi bernilai `true`, maka keseluruhan ekspresi akan bernilai `true`.

Tabel berikut adalah tabel kebenaran untuk operator `||`. Tabel ini mendaftar semua kemungkinan kombinasi nilai dua ekspresi yang dihubungkan dengan operator `||`.

Ekspresi	Nilai Ekspresi
<code>true    false</code>	<code>true</code>
<code>false    true</code>	<code>true</code>
<code>false    false</code>	<code>false</code>
<code>true    true</code>	<code>true</code>

Pada tabel dapat dilihat, ekspresi kombinasi dengan operator `||` bernilai `true` jika salah satu atau kedua subekspresi bernilai `true` dan hanya bernilai `false` jika kedua ekspresi bernilai `false`.

Program di bawah mencontohkan penggunaan operator `||`. Program adalah program aplikasi pinjaman yang menentukan apakah seseorang memenuhi syarat pengajuan pinjaman berdasarkan lama kerja dan gaji. Pinjaman diberikan jika seseorang mempunyai gaji lebih besar dari Rp. 7.500.000,- ATAU lama kerja lebih besar dari 5 tahun.

#### Program (Pinjaman.java)

```

import java.util.Scanner;

/*
    Program ini menentukan apakah seseorang memenuhi syarat
    pengajuan pinjaman berdasarkan gaji dan lama kerja.
    Program ini menggunakan operator ||.
*/
public class Pinjaman
{
    public static void main(String[] args)
    {
        double gaji;
        int lamaKerja;

        Scanner keyboard = new Scanner(System.in);

        // Dapatkan gaji pengguna
        System.out.print("Masukkan gaji Anda: ");
        gaji = keyboard.nextDouble();

        // Dapatkan lama kerja pengguna
        System.out.print("Lama kerja (tahun): ");
        lamaKerja = keyboard.nextInt();
    }
}

```

```
// Tentukan apakah pengguna dapat mengajukan pinjaman
if (gaji >= 7500000 || lamakerja > 5)
{
    System.out.println("Anda dapat mengajukan pinjaman.");
}
else
{
    System.out.println("Anda tidak memenuhi syarat mengajukan
pinjaman");
}
}
```

### Output Program (Pinjaman.java)

```
Masukkan gaji Anda: 7650000
Lama kerja (tahun): 2
Anda dapat mengajukan pinjaman.
```

## Operator !

Operator **!** disebut sebagai operator logis NOT. Operator ini hanya membutuhkan satu operand dan melakukan negasi terhadap nilai kebenaran dari operand tersebut. Dengan kata lain, jika sebuah ekspresi bernilai `true`, operator **!** mengubahnya menjadi bernilai `false`. Berikut adalah contoh penggunaan operator **!** dalam *statement if*:

```
if (!(temperatur > 100))
{
    System.out.println("Ini di bawah temperatur maksimum.");
}
```

Kode di atas akan menampilkan pesan jika `temperatur` TIDAK lebih besar dari 100.

Tabel berikut adalah tabel kebenaran untuk operator **!**.

Ekspresi	Nilai Ekspresi
<code>!true</code>	<code>false</code>
<code>!false</code>	<code>true</code>

## 4.6 Membandingkan Karakter, *Floating Point*, dan String

Terdapat hal-hal yang perlu diperhatikan ketika membandingkan data bertipe karakter, *floating point* dan `string`. Pada bagian ini kita akan membahas perbandingan antar karakter-karakter, antar *floating-floating point*, dan antar *string-string*.

### Membandingkan Karakter

Kita dapat menggunakan operator relasional untuk menguji data berupa karakter. Sebagai contoh misalkan `ch` adalah variabel `char`, kode berikut menguji apakah karakter yang disimpan dalam `ch` adalah sama dengan `'A'`:

```
if (ch == 'A')
{
    System.out.println("Huruf pada ch adalah A.");
}
```

Kita juga dapat menggunakan operator-operator relasional lainnya untuk dengan data `char`. Ketika kita menuliskan ekspresi *Boolean* dengan operator relasional antara dua tipe data `char` yang dilakukan komputer adalah membandingkan nilai Unicode dari karakter tersebut.

Kode berikut mencontohkan penggunaan operator relasional terhadap dua tipe data `char`:

```
char ch = 'A';
if (ch < 'B')
{
    System.out.println("ch lebih kecil dari B");
}
```

Kode di atas akan mencetak:

```
ch lebih kecil dari B
```

Ini karena nilai Unicode dari karakter `'A'` adalah 65 sedangkan nilai Unicode dari karakter `'B'` adalah 66.

### Membandingkan *Floating Point*

Komputer tidak dapat melakukan kalkulasi angka *floating point* secara akurat. Hasil kalkulasi *floating point* umumnya mempunyai *rounding error* (kesalahan pembulatan). Sehingga, ketika kita membandingkan hasil kalkulasi operasi aritmatika pada *floating point* kita harus memperhitungkan *rounding error* ini.

Sebagai contoh, perhatikan kode berikut:

```
double r = Math.sqrt(2.0);

if (r * r == 2.0)
{
    System.out.println("Math.sqrt(2.0) dikuadratkan sama dengan 2.0");
}
else
{
    System.out.println("Math.sqrt(2.0) dikuadratkan tidak sama dengan 2.0");
    System.out.println("Math.sqrt(2.0) = " + r * r);
}
```

Output dari kode di atas adalah seperti berikut:

```
Math.sqrt(2.0) dikuadratkan tidak sama dengan 2.0
Math.sqrt(2.0) = 2.0000000000000004
```

Output dari kode di atas tidak sesuai dengan ekspektasi kita, karena secara matematika logika dari kode di atas sudah benar: karena  $r = \sqrt{2.0}$  maka  $r^2$  atau `r * r` haruslah sama dengan 2.0. Akan tetapi, program mengkalkulasi `r * r` menjadi `2.0000000000000004`. Perbedaan `0.0000000000000004` adalah *rounding error* dari kalkulasi ini.

Untuk membandingkan apakah dua nilai *floating point* adalah sama, kita menuliskan kode yang membandingkan apakah kedua nilai tersebut cukup dekat, yaitu dengan membandingkan apakah besaran dari selisih keduanya kurang dari suatu batas. Secara matematika, kita menulis bahwa `x` dan `y` adalah cukup dekat jika

$$|x - y| < \epsilon$$

dimana  $\epsilon$  adalah angka yang sangat kecil.  $\epsilon$  adalah huruf latin epsilon, huruf yang digunakan untuk menotasikan angka yang sangat kecil. Umumnya,  $\epsilon$  ditetapkan dengan nilai  $10^{-14}$  ketika membandingkan angka `double`.

Kode di atas dapat kita perbaiki dengan memperhitungkan *rounding error* menjadi seperti berikut:

```
final double EPSILON = 1E-14;

double r = Math.sqrt(2.0);
if (Math.abs(r * r - 2.0) < EPSILON)
{
    System.out.println("Math.sqrt(2.0) dikuadratkan adalah 2.0");
}
else
{
    System.out.println("Math.sqrt(2.0) dikuadratkan tidak sama dengan 2.0");
}
```

Kode di atas akan memberikan *output*:

```
Math.sqrt(2.0) dikuadratkan adalah 2.0
```

## Membandingkan String

Kita telah melihat pada subbab-subbab sebelumnya bahwa data numerik dan data karakter dapat dibandingkan menggunakan operator relasional. Namun, kita tidak dapat menggunakan operator relasional untuk membandingkan *object* `String`. Ingat bahwa variabel yang dideklarasikan sebagai `String` tidak menyimpan nilai `String` namun menyimpan alamat memori yang mereferensi ke *object* `String` sebenarnya. Ketika kita menggunakan operator relasional untuk membandingkan variabel `String`, yang dilakukan program adalah membandingkan nilai alamat memori dari *object* `String` bukan isi dari *object* `String`. Sebagai contoh, perhatikan program berikut.

### Program (PerbandinganString.java)

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan penggunaan operator relasional yang salah
    untuk membandingkan nilai dua string.
*/
public class PerbandinganString
{
    public static void main(String[] args)
    {
        String nama1, nama2;

        Scanner keyboard = new Scanner(System.in);

        // Minta nama1
        System.out.print("Masukkan nama 1: ");
        nama1 = keyboard.nextLine();

        // Minta nama2
        System.out.print("Masukkan nama 2: ");
        nama2 = keyboard.nextLine();

        // statement if-else untuk menentukan apakah
        // kedua nama sama
        if (nama1 == nama2)
        {
            System.out.println("Nama 1: " + nama1);
            System.out.println("Nama 2: " + nama2);
            System.out.println("Kedua nama sama.");
        }
        else
        {
            System.out.println("Nama 1: " + nama1);
            System.out.println("Nama 2: " + nama2);
            System.out.println("Kedua nama tidak sama.");
        }
    }
}
```

### Output Program (PerbandinganString.java)

```
Masukkan nama 1: Budi
Masukkan nama 2: Budi
Nama 1: Budi
Nama 2: Budi
Kedua nama tidak sama.
```

Pada *output* dapat dilihat, meskipun pengguna memasukkan dua nama yang sama, program tetap menganggapnya dua nama yang berbeda. Ini karena, ketika program membandingkan `nama1` dan `nama2` dengan operator `==`, program membandingkan alamat memori *object* `String` dari *input* pertama dengan alamat memori *object* `String` dari *input* kedua. Dan, karena kedua alamat memori *object* `String` berbeda, maka program menganggapnya tidak sama.

Untuk membandingkan apakah dua *object* `String` menyimpan nilai `String` yang sama, kita menggunakan *method* `equals`. Misalkan, jika variabel `nama1` dan variabel `nama2` keduanya adalah variabel referensi `String`, kita menuliskan ekspresi berikut untuk membandingkan apakah nilai `String` yang direferensikan oleh kedua variabel tersebut sama:

```
nama1.equals(nama2)
```

*Method* `equals` memberikan nilai `true` jika `nama1` dan `nama2` adalah `String` bernilai sama dan memberikan nilai `false` jika keduanya tidak sama. Program `PerbandinganString.java` dapat kita perbaiki menjadi seperti pada program berikut.

#### **Program (PerbandinganStringEquals.java)**

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan penggunaan method equals untuk
    membandingkan dua string.
*/

public class PerbandinganStringEquals
{
    public static void main(String[] args)
    {
        String nama1, nama2;

        Scanner keyboard = new Scanner(System.in);

        // Minta nama1
        System.out.print("Masukkan nama 1: ");
        nama1 = keyboard.nextLine();

        // Minta nama2
        System.out.print("Masukkan nama 2: ");
        nama2 = keyboard.nextLine();

        // statement if-else untuk menentukan apakah
        // kedua nama sama
        if (nama1.equals(nama2))
        {
            System.out.println("Nama 1: " + nama1);
            System.out.println("Nama 2: " + nama2);
            System.out.println("Kedua nama sama.");
        }
    }
}
```

```

    }
    else
    {
        System.out.println("Nama 1: " + nama1);
        System.out.println("Nama 2: " + nama2);
        System.out.println("Kedua nama tidak sama.");
    }
}
}

```

#### Output Program (PerbandinganStringEquals.java)

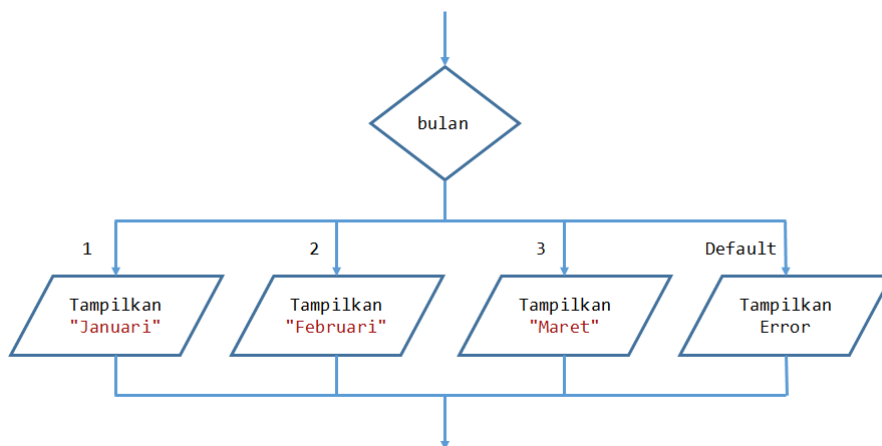
```

Masukkan nama 1: Budi
Masukkan nama 2: Budi
Nama 1: Budi
Nama 2: Budi
Kedua nama sama.

```

## 4.7 Statement switch

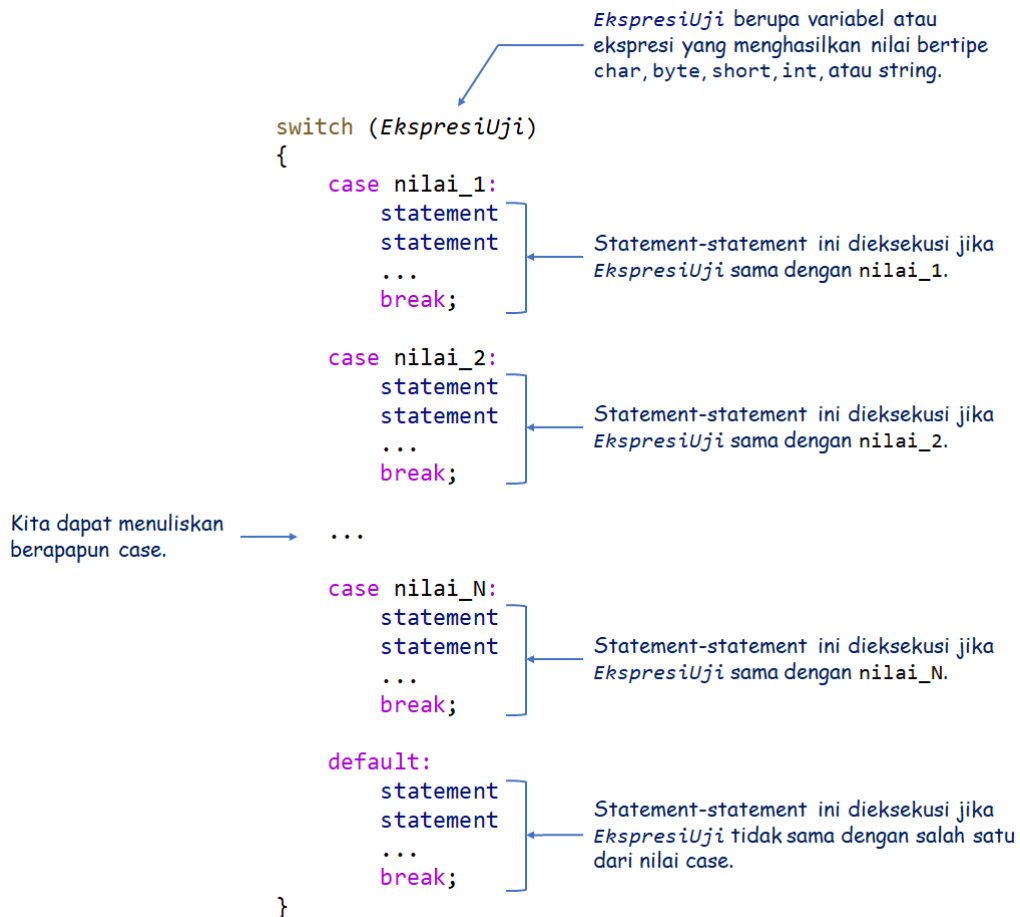
*statement* `switch` memungkinkan program untuk menguji nilai dari sebuah variabel atau sebuah ekspresi dan kemudian menggunakan nilai tersebut untuk menentukan sekelompok *statement* yang dieksekusi. *Flowchart* berikut mengilustrasikan logika dari sebuah contoh *statement* `switch`.



Dalam *flowchart*, bentuk belah ketupat menunjukkan `bulan`, yang merupakan nama dari sebuah variabel. Jika variabel `bulan` berisi nilai 1, maka program menampilkan `Januari`. Jika variabel `bulan` berisi nilai 2, maka program menampilkan `Februari`. Jika variabel `bulan` berisi nilai 3, maka program menampilkan `Maret`. Jika variabel `bulan` berisi bukan nilai 1, 2, atau 3, maka aksi yang dilabeli dengan `Default` dieksekusi, yaitu menampilkan `Error: Bulan salah`.

Bentuk umum penulisan *statement* `switch` adalah sebagai berikut:





Baris pertama `statement switch` diawali dengan `keyword switch`, yang diikuti oleh `EkspresiUji` yang ditulis dalam tanda kurung. `EkspresiUji` adalah sebuah variabel atau sebuah ekspresi yang memberikan nilai `char`, `byte`, `short`, `int`, atau nilai `string`.

Di dalam body dari `statement switch` yang dituliskan dalam tanda kurung kurawal, adalah satu atau lebih bagian `case` (kasus). Setiap bagian `case` dimulai dengan `keyword case` yang diikuti oleh sebuah nilai, lalu diikuti dengan titik dua. Setiap bagian `case` berisi satu atau lebih `statement`, yang diikuti oleh `statement break`. Setelah semua bagian `case`, sebuah bagian `default` opsional dapat dituliskan.

Ketika `statement switch` dieksekusi, nilai dari `EkspresiUji` dibandingkan dengan nilai pada setiap bagian `case` dari atas ke bawah. Jika program menemukan sebuah nilai `case` yang cocok dengan nilai `EkspresiUji`, program mengeksekusi `statement-statement` pada bagian `case` tersebut sampai dengan `statement break`. `statement break` menyebabkan program keluar dari `statement switch`, sehingga program tidak menguji bagian `case` selanjutnya. Jika `EkspresiUji` tidak ada yang cocok dengan nilai dari semua `case`, maka program mengeksekusi `statement-statement` dalam bagian `default`.

Sebagai contoh, kode berikut melakukan operasi yang sama seperti pada *flowchart* sebelumnya. Asumsikan variabel `bulan` adalah variabel `int`.

```
switch (bulan)
{
    case 1:
        System.out.println("Januari");
        break;

    case 2:
        System.out.println("Februari");
```

```

        break;

    case 3:
        System.out.println("Maret");
        break;

    default:
        System.out.println("Error: Bulan salah");
        break;
}

```

Pada contoh *statement switch* di atas, *Ekspresi Uji* adalah variabel `bulan`. Nilai variabel `bulan` akan dievaluasi dan salah satu dari aksi berikut akan terjadi:

- Jika nilai dari `bulan` adalah 1, program akan ke `case 1:` dan mengeksekusi `System.out.println("Januari");`. Kemudian, program mengeksekusi *statement break* yang menyebabkan program keluar dari *statement switch*.
- Jika nilai dari `bulan` adalah 2, program akan ke `case 2:` dan mengeksekusi `System.out.println("Februari");`. Kemudian, program mengeksekusi *statement break* yang menyebabkan program keluar dari *statement switch*.
- Jika nilai dari `bulan` adalah 3, program akan ke `case 3:` dan mengeksekusi `System.out.println("Maret");`. Kemudian, program mengeksekusi *statement break* yang menyebabkan program keluar dari *statement switch*.
- Jika nilai dari `bulan` bukan 1, 2, atau 3, program akan ke `default:` dan mengeksekusi `System.out.println("Error: Bulan salah");`. Kemudian, program mengeksekusi *statement break* yang menyebabkan program keluar dari *statement switch*.

*statement switch* dapat digunakan sebagai alternatif dari *statement if-else-if* yang membandingkan variabel atau ekspresi yang sama ke beberapa nilai-nilai berbeda. Sebagai contoh, *statement switch* pada contoh sebelumnya bekerja sama seperti *statement if-else-if* berikut:

```

if (bulan == 1)
{
    System.out.println("Januari");
}
else if (bulan == 2)
{
    System.out.println("Februari");
}
else if (bulan == 3)
{
    System.out.println("Maret");
}
else
{
    System.out.println("Error: Bulan salah");
}

```

Program berikut memperlihatkan bagaimana *statement switch* sederhana bekerja.

### Program (DemoSwitch.java)

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan statement switch.
*/
public class DemoSwitch
{
    public static void main(String[] args)
    {
        int bulan; // Untuk menyimpan angka bulan

        Scanner keyboard = new Scanner(System.in);

        // Minta angka bulan ke pengguna
        System.out.print("Masukkan angka bulan (1 s.d 3): ");
        bulan = keyboard.nextInt();

        // statement switch untuk menampilkan nama bulan sesuai angka bulan
        switch (bulan)
        {
            case 1:
                System.out.println("Januari");
                break;

            case 2:
                System.out.println("Februari");
                break;

            case 3:
                System.out.println("Maret");
                break;

            default:
                System.out.println("Angka bulan yang Anda masukkan salah.");
                break;
        }
    }
}
```

### Output Program (DemoSwitch.java)

#### Input 1

```
Masukkan angka bulan (1 s.d 3): 3
Maret
```

#### Input 2

```
Masukkan angka bulan (1 s.d 3): 8
Angka bulan yang Anda masukkan salah.
```

## case Tanpa *Statement* break

*statement* `break` pada setiap bagian `case` digunakan untuk membuat program keluar dari *statement* `switch`. Tanpa *statement* `break`, program akan mengeksekusi *statement-statement* dalam bagian-bagian `case` di bawahnya setelah menemukan nilai `case` yang cocok. Sebagai contoh, program berikut adalah modifikasi program `DemoSwitch.java` tanpa *statement* `break`.

### Program (*TanpaBreak.java*)

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan statement switch
    tanpa statement break dalam setiap bagian case.
*/
public class TanpaBreak
{
    public static void main(String[] args)
    {
        int bulan; // Untuk menyimpan angka bulan

        Scanner keyboard = new Scanner(System.in);

        // Minta angka bulan ke pengguna
        System.out.print("Masukkan angka bulan (1 s.d 3): ");
        bulan = keyboard.nextInt();

        // statement switch untuk menampilkan nama bulan sesuai angka bulan
        switch (bulan)
        {
            case 1:
                System.out.println("Januari");

            case 2:
                System.out.println("Februari");

            case 3:
                System.out.println("Maret");

            default:
                System.out.println("Angka bulan yang Anda masukkan salah.");
        }
    }
}
```

### Output Program (*TanpaBreak.java*)

#### Input 1

```
Masukkan angka bulan (1 s.d 3): 1
Januari
Februari
Maret
Angka bulan yang Anda masukkan salah.
```

### Input 2

```
Masukkan angka bulan (1 s.d 3): 2
Februari
Maret
Angka bulan yang Anda masukkan salah.
```

### Input 3

```
Masukkan angka bulan (1 s.d 3): 3
Maret
Angka bulan yang Anda masukkan salah.
```

Dapat dilihat pada *output* program di atas, jika pengguna memasukkan nilai 1, program mengeksekusi *statement* dalam bagian `case 1:` karena nilai `bulan` cocok dengan nilai `case 1:`, namun karena tidak ada *statement break* dalam bagian `case 1:` maka program melanjutkan mengeksekusi *statement-statement* dalam `case` selanjutnya termasuk bagian `default`.

Terkadang bagian `case` tanpa *statement break* adalah hal yang kita inginkan. Misalkan, program berikut meminta pengguna untuk memilih menu paket. Pilihan yang tersedia adalah A, B, dan C. *statement switch* digunakan untuk mengenali pilihan-pilihan tersebut dalam huruf besar maupun huruf kecil.

### Program (PilihanMenu.java)

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan statement switch
    untuk memilih menu A, B, C. Program menerima pilihan
    dalam huruf besar dan huruf kecil.
*/
public class PilihanMenu
{
    public static void main(String[] args)
    {
        String input; // Untuk menyimpan input pengguna
        char menu;     // Untuk menyimpan pilihan menu

        Scanner keyboard = new Scanner(System.in);

        // Minta angka bulan ke pengguna
        System.out.print("Masukkan paket yang diinginkan (A, B, atau C): ");
        input = keyboard.nextLine();
        menu = input.charAt(0);

        // *statement* switch untuk menampilkan nama bulan sesuai angka bulan
        switch (menu)
        {
            case 'a':
            case 'A':
                System.out.println("Paket A");
                System.out.println("Harga paket: Rp.15.000");
                break;

            case 'b':
```

```

        case 'B':
            System.out.println("Paket B");
            System.out.println("Harga paket: Rp.25.000");
            break;

        case 'c':
        case 'C':
            System.out.println("Paket C");
            System.out.println("Harga paket: Rp.30.000");
            break;

        default:
            System.out.println("Pilihan yang Anda masukkan salah.");
            break;
    }
}

```

### **Output Program (PilihanMenu.java)**

#### *Input 1*

```

Masukkan paket yang diinginkan (A, B, atau C): B
Paket B
Harga paket: Rp.25.000

```

#### *Input 2*

```

Masukkan paket yang diinginkan (A, B, atau C): b
Paket B
Harga paket: Rp.25.000

```

## REFERENSI

- [1] Horstmann, Cay S. 2012. *Big Java: Late Objects, 1st Edition*. United States of America: John Wiley & Sons, Inc.
- [2] Gaddis, Tony. 2016. *Starting Out with Java: From Control Structures through Objects (6th Edition)*. Boston: Pearson.