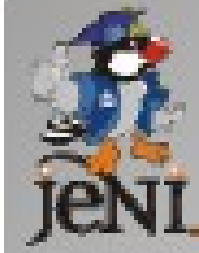
JARDIKNAS

Struktur kontrol

Pengenalan Pemrograman 1

Versi 2.0



tujuan

Pada akhir pembahasan, peserta diharapkan mampu untuk:

- Menggunakan struktur kontrol keputusan (**if**, **else**, **switch**), untuk memilih bagian dari program yang akan dieksekusi.
- Menggunakan struktur kontrol pengulangan (**while**, **do-while**, **for**), untuk melakukan pengulangan eksekusi program atau code, sejumlah yang telah ditentukan.
- Menggunakan *branching statement* (**break**, **continue**, **return**) untuk mengarahkan alur program atau code.

Struktur kontrol

Struktur kontrol

- Digunakan untuk mengatur susunan proses eksekusi *statement-statement* di dalam program.
-

Struktur kontrol mempunyai dua tipe:

- Struktur kontrol keputusan

  Digunakan untuk memilih bagian dari code yang akan dieksekusi.

- Struktur kontrol pengulangan

  digunanakan untuk mengeksekusi bagian tertentu sesuai dengan jumlah angka pengulangannya.

Struktur Kontrol Keputusan

Struktur kontrol keputusan

digunakan untuk memilih dan mengeksekusi block tertentu dari code yang dapat berpindah ke bagian lain.

Tipe-tipe:

- statement-if
- statement-if-else
- statement-if-else if

statement-if

- statement-if
  - Menspesifikasikan sebuah statement (atau block dari code) yang akan dieksekusi jika dan hanya jika statement boolean bernilai true.
- Form statement-if:

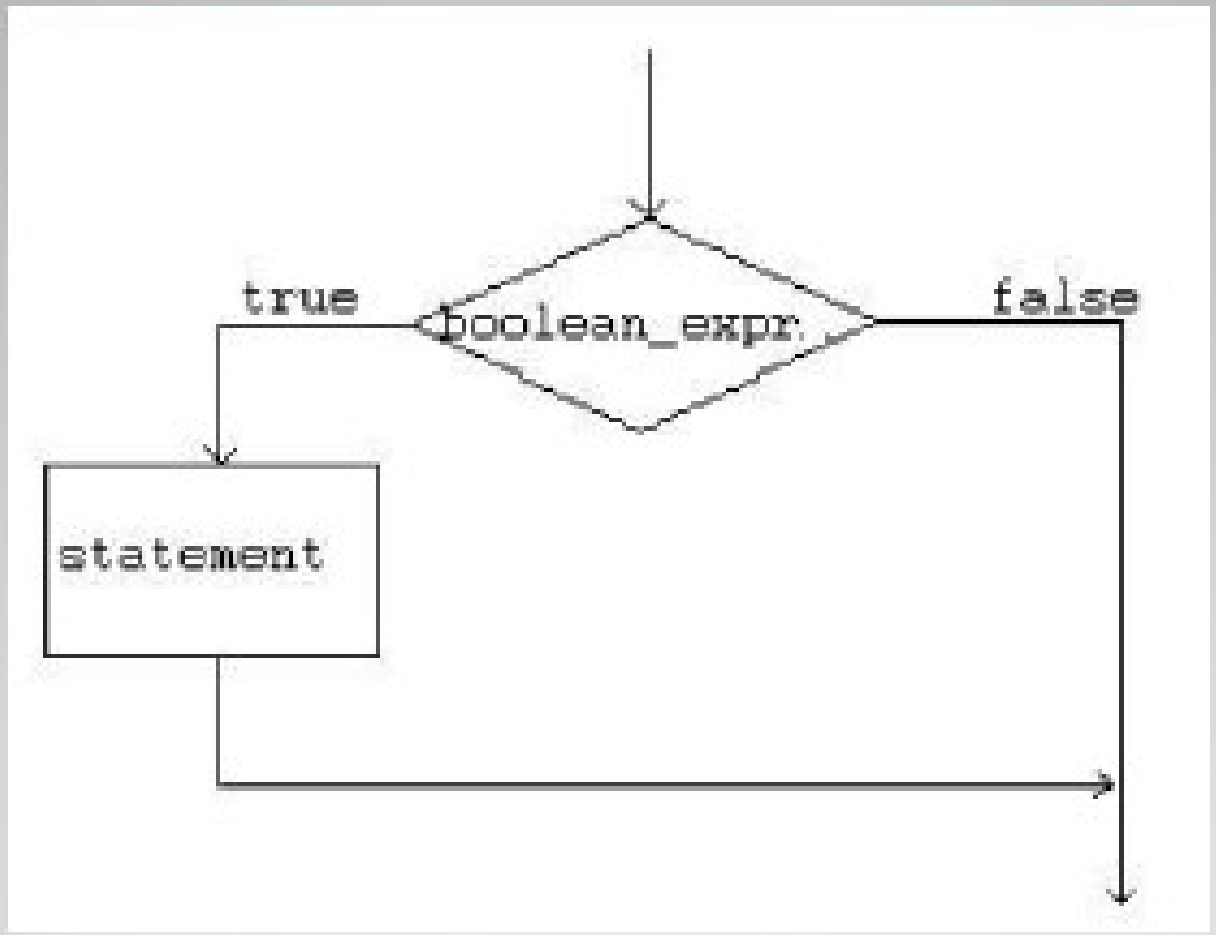
```
if( boolean_ekspresi )
        statement;
```

atau

```
if( boolean_ekspresi ){
        statement 1;
        statement 2;
}
```

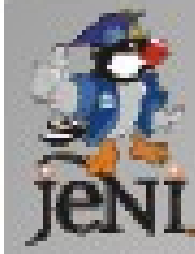
  - dimana,
    - boolean_ekspresi sama dengan boolean ekspresi atau boolean variabel.

if-statement Flowchart



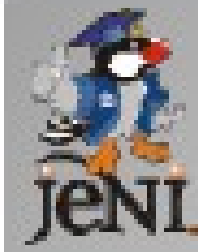
Contoh 1

```
int grade = 68;
  if( grade > 60 )
       System.out.println("Selamat!");
```



Contoh 2

```
int grade = 68;
  if( grade > 60 ){
   System.out.println("Selamat!");
   System.out.println("Anda Berhasil!");
}
```



Panduan penulisan program

1. **Ekspresi Boolean** merupakan bagian dari sebuah statement yang harus dieveluasi ke sebuah nilai boolean. Hal tersebut berarti bahwa eksekusi dari kondisi harus memiliki nilai true atau false.

2. statement dalam blok-if.

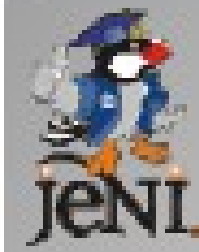
Contoh,

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

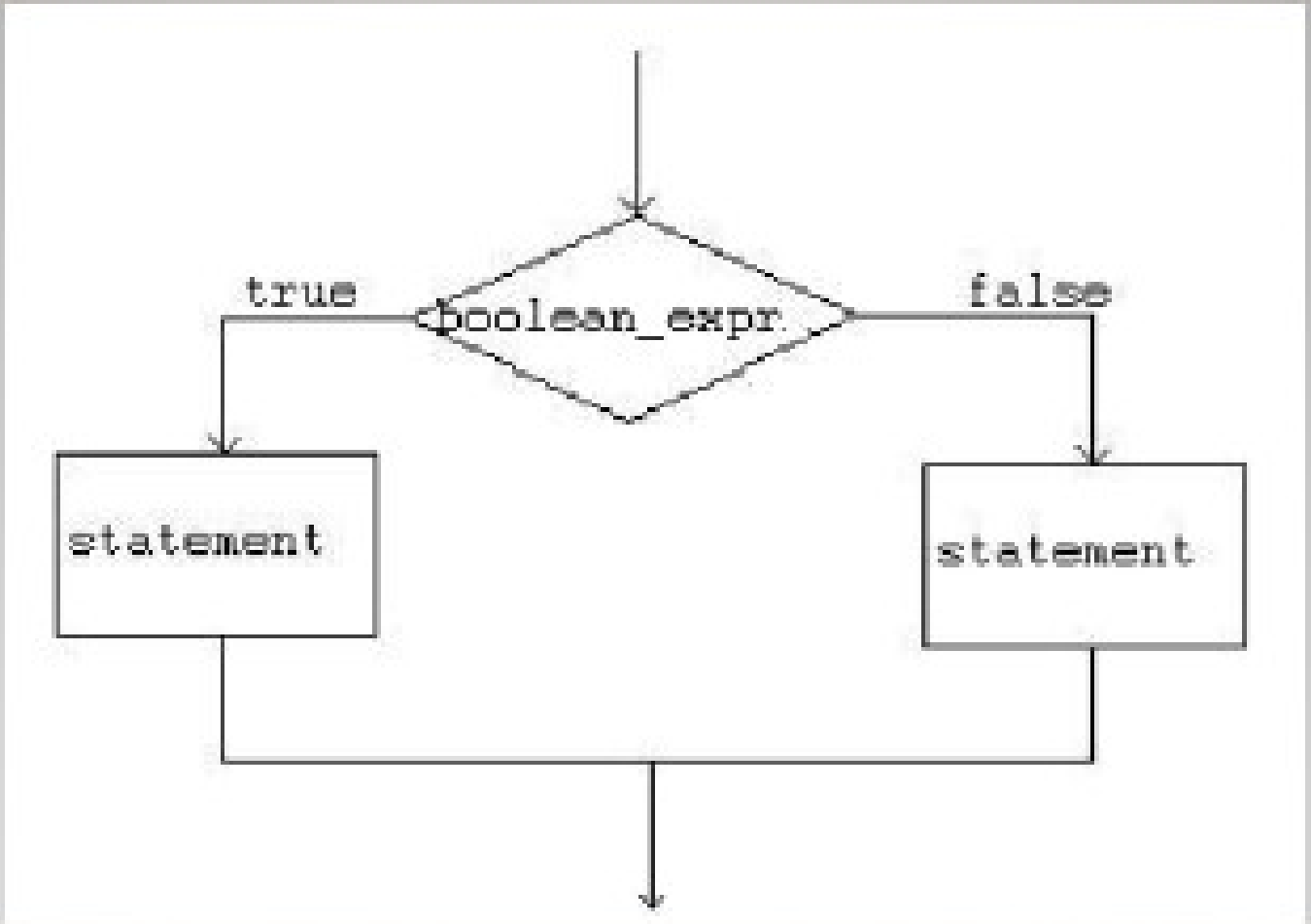
statement if-else

- statement if-else
  - Digunakan ketika kita akan mengeksekusi sebuah statement jika kondisinya true, dan statement yang lain jika berkondisi false.
- Form statement if-else:

```
if( boolean_ekspresi ){
        statement1;
        statement2;
        . . .
}
else{
        statement3;
        statement4;
        . . .
}
```

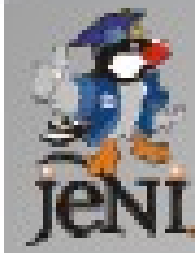


Flowchart



Contoh 1

```
int grade = 68;

if( grade > 60 )
   System.out.println("Selamat!");
else
   System.out.println("Maaf Anda gagal");
```



Contoh 2

```
int grade = 68;

if( grade > 60 ){
   System.out.println("Selamat!");
   System.out.println("Anda berhasil!");
}
else{
   System.out.println("Maaf Anda gagal");
}
```

Panduan Penulisan Program

1. Untuk menghindari kesalahan, selalu letakkan statement-statement dari blok if atau if-else didalam tanda {}.

2. Anda dapat memiliki blok if-else berantai. Artinya Anda dapat memilki blok if-else yang lain didalam blok if-else yang lain.

   Contoh,

```
if( boolean_ekspresi ){
    if( boolean_ekspresi ){
            //statement
    }
}
else{
     // statement
}
```

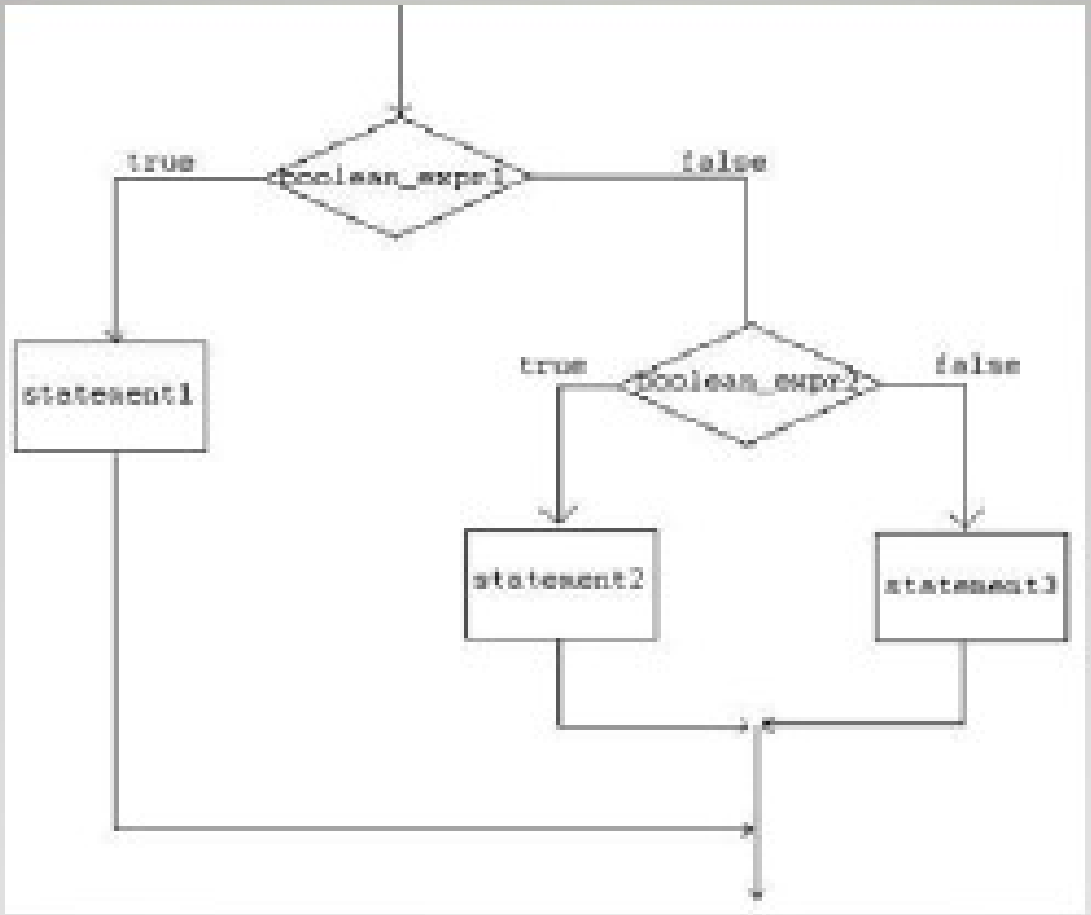
statement if-else-else if

- statement pada klausa else dari sebuah blok if-else dapat menjadi struktur if-else yang lain.
- Struktur ini memperbolehkan kita untuk membuat pilihan yang lebih kompleks.
- Form statement if-else-else if:

```
if( boolean ekspresi1 )
        statement1;
else if( boolean ekspresi2 )
        statement2;
else
        statement3;
```


jeNI

Flowchart


true
boolean_expr1
false
statement1
true
boolean_expr2
false
statement2
statement3

Contoh

```
int grade = 68;

if( grade > 90 ){
   System.out.println("Sangat Bagus!");
}
else if( grade > 60 ){
   System.out.println("Sangat Bagus!");
}
else{
   System.out.println("Maaf Anda gagal");
}
```

Kesalahan Umum

1. Kondisi didalam statement if-statement tidak ditentukan dalam nilai boolean.
   Contoh,

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

Angka variabel tidak ditentukan sebagai nilai boolean.

2. Menulis elseif sebagai ganti dari else if.

Kesalahan Umum

3. Menggunakan = pengganti dari == sebagai pembanding. Contoh,

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

Seharusnya ditulis,

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

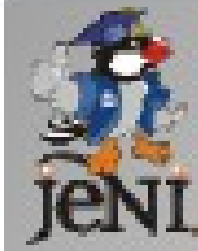
Contoh Program

```
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("Bagus!" );
      }
      else if( (grade < 80) && (grade >= 60)){
           System.out.println("Belajar lagi!" );
      }
      else{
                System.out.println("Maaf, Anda gagal.");


       }
   }
}
```

statement-switch

- Switch
  - Memperbolehkan percabangan pada multiple outcomes.
- Form statement-switch:

```
switch( switch_ekspresi ){
        case case_pilihan1:
                statement1;//
                statement2;//blok 1
                break;
        case case_pilihan2:
                statement1;//
                statement2;//blok 2
                break;
                        :
        default:
                statement1;//
                statement2;//blok n
}
```



statement-switch

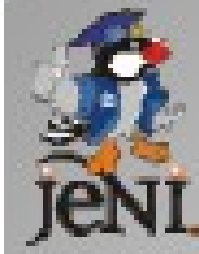
- Dimana,
  - ekspresi switch
    - Merupakan integer atau karakter ekspresi
  - case_pilihan1, case_pilihan2 dan yang lainnya,
    - merupakan integer unique atau karakter tetap.

statement-switch

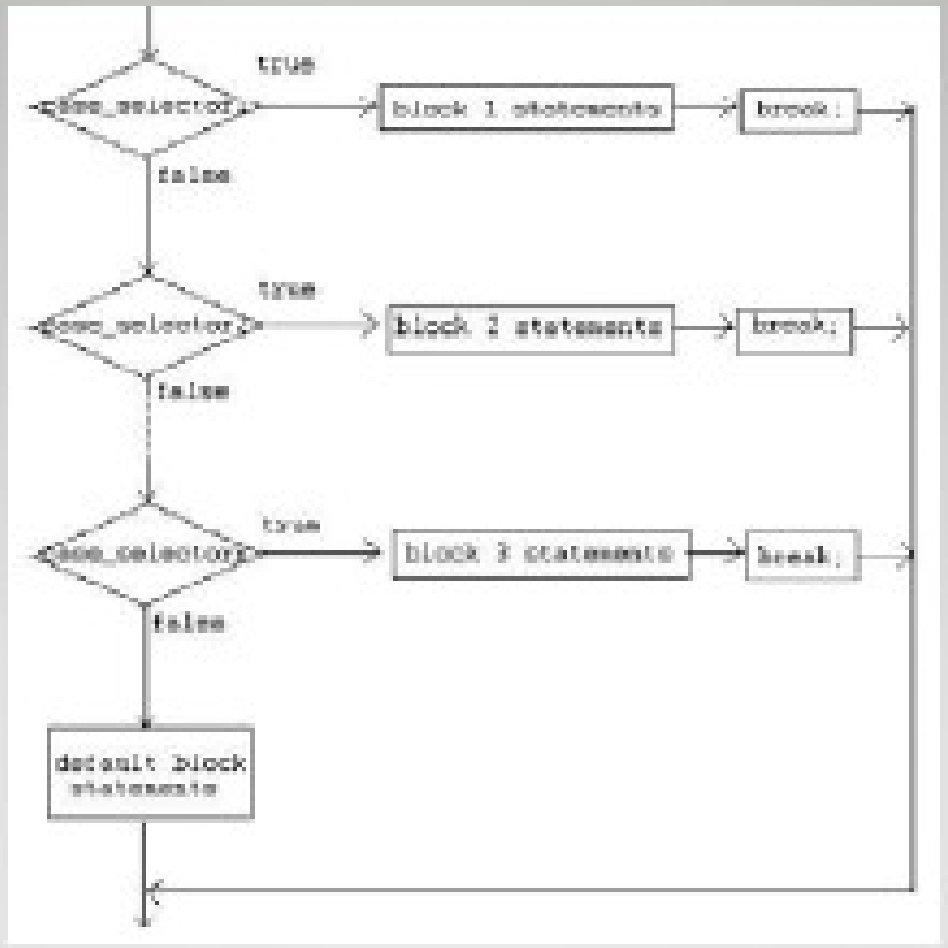
- Ketika sebuah switch digunakan,
  - Java akan menilai ekspresi switch, kemudian berpindah ke case yang pilihan dari pemilih sesuai dengan nilai dari ekspresi.
  - Program mengeksekusi statement yang diminta dari point sebuah case sampai statement break dibaca, kemudian pindah ke statement awal setelah membaca akhir dari struktur switch.
  - Jika tidak ada case yang sesuai, maka blok default akan dieksekusi. Catatan, bahwa bagian default merupakan pilihan.

statement-switch

- CATATAN:
  - Tidak sama dengan statement-if, statement multiple dieksekusi pada statement-switch, tanpa membutuhkan statement percabangan (braches statement).
  - Ketika sebuah case pada statement-switch sesuai, semua statement yang ada didalam case tersebut akan dieksekusi. Tidak hanya itu, statement yang berhubungan dengan case tersebut juga akan dieksekusi.
  - Untuk mencegah program dari pengeksekusian statement pada case sebelumnya, kita menggunakan statement-break sebagai statement akhir.



Flowchart


true
case selector
block 1 statements
break;
false
true
case selector
block 2 statements
break;
false
true
case selector
block 3 statements
break;
false
default block
statements

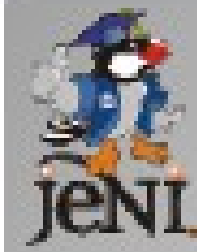
Contoh

```
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("Bagus!" );
             break;
                                        case 80:
             System.out.println("Belajar lagi!" );
             break;
                                        default:
             System.out.println("Maaf, Anda gagal.");
        }
    }
}
```

Panduan Penulisan Program

1. Penentuan penggunaan statement-if atau statement-switch berdasarkan pada requirement output program.

2. Sebuah statement-if dapat digunakan untuk membuat keputusan berdasarkan pada deretan dari nilai atau kondisi, dimana statement-switch dapat membuat keputusan hanya berdasar kepada single integer atau nilai karakter. Juga, nilai yang disediakan untuk setiap statement-case harus berbeda (unique).



Struktur Kontrol Pengulangan

- Struktur kontrol pengulangan
  - Pada statement Java, kita dapat menentukan angka pengulangan yang akan dilakukan,
- Tipe:
  - Pengulangan-while
  - Pengulangan-do-while
  - Pengulangan-for

Pengulangan-while

- Pengulangan while
  - Merupakan statement atau blok dari statement yang diulang selama kondisinya sesuai.
- Form pengulangan while:

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

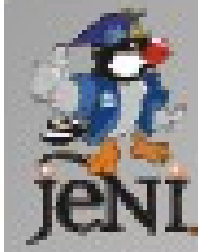
  - statement didalam pengulangan while akan dieksekusi selama boolean_ekspresi bernilai true.

Contoh 1

```
int x = 0;

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

Contoh 2

```
//Pengulangan tanpa batas
  while(true)
        System.out.println("hello");
```



Contoh 3

```
//Tanpa pengulangan
// statement yang tidak pernah dieksekusi
while (false)
        System.out.println("hello");
```

statement-do-while

- statement-do-while
  - Sama dengan pengulangan-while
  - statement didalam pengulangan do-while akan dieksekusi beberapa kali selama kondisinya sesuai dengan ekspressi yang diberikan.
  - Hal utama yang membedakan antara pengulangan while dan do-while:
    - statement didalam pengulangan do-while loop setidaknya dieksekusi satu kali.

- Form pengulangan-do-while:

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

Contoh 1

```
int x = 0;

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

Contoh 2

```
//pengulangan tanpa batas
do{
  System.out.println("hello");
} while (true);
```

Contoh 3

```
//satu kali pengulangan
// statement dieksekusi satu kali
do
  System.out.println("hello");
while (false);
```

Petunjuk Penulisan Program

1. Kesalahan pemrograman secara umum terjadi, ketika lupa menulis semi-colon setelah ekspresi while pada saat menggunakan pengulangan do-while

```
do{
   ...
 }while(boolean_ekspresi)//SALAH->lupa semicolon;
```

2. Sama halnya dengan pengulangan while, pastikan bahwa pengulangan do-while akan diakhiri dengan semicolon.

Pengulangan-for

- Pengulangan-for
  - Digunakan untuk mengeksekusi code yang bernilai sama, berulang-ulang.
- Form pengulangan-for:

```
for(InisialisasiEkspresi;KondisiPengulangan;StepEkspresi)
{
    statement1;
    statement2;
    . . .
}
```

  - dimana,

InisialisasiEkspresi	–meninisialisasi variabel pengulangan.
KondisiPengulangan	– membandingkan variabel pengulangan dengan nilai limit.
StepEkspresi	- memperbarui variabel pengulangan.

Contoh

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

code diatas sama dengan pengulangan-while dibawah ini.

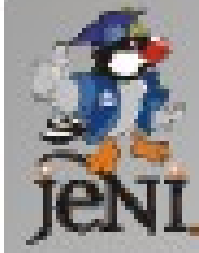
```
int i = 0;
while( i < 10 ){
     System.out.print(i);
     i++;
}
```

For lanjut

- int [] a = {1,2,3,4};
- for(int x = 0; x < a.length; x++) // basic for loop
- System.out.print(a[x]);
- for(int n : a) // enhanced for loop
- System.out.print(n);

Branching statement

- statement branching dapat digunakan untuk mengatur flow dari pengeksekusian program.
- Java menyediakan tiga statement branching:
  - break
  - continue
  - return.



Unlabeled break statement

Unlabeled break

- Mengakhiri statement switch
- Juga dapat digunakan untuk mengakhiri pengulangan for, while, atau do-while

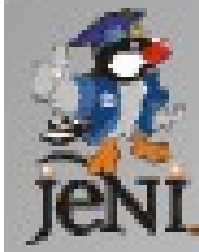
Contoh

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

String  searchName = "Ysa";
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 + " ditemukan!" );
else System.out.println( searchName + " tidak ditemukan." );
```

labeled break statement

labeled break statement

- Mengakhiri sebuah statement, yang diidentifikasi oleh spesifikasi label pada statement break.
- Pada slide berikut terdapat contoh, untuk mencari sebuah nilai pada array dua dimensi. Pengulangan melewati dua array. Ketika nilainya ditemukan, sebuah labeled break mengakhiri statement labeled search, yang digunakan diluar pengulangan.



Contoh

```
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 + " ditemukan!" );
else System.out.println(searchNum + " tidak ditemukan!");
```

Unlabeled Continue statement

unlabeled continue statement

Pindah ke akhir dari bagian pengulangan dan memberikan nilai boolean ekspresi yang mengontrol pengulangan tersebut, pada dasarnya perpindahan merupakan pengingat(remainder) dari iterasi yang berasal dari pengulangan.

Contoh

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

for( int i=0; i<names.length; i++ ){
   if( !names[i].equals("Beah") ){
     continue;        //pindah ke statement berikutnya
   }
   count++;
}
System.out.println("Inilah "+count+" Beah pada daftar");
```

Contoh

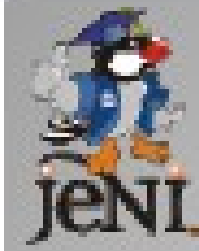
```
outerLoop:
for( int i=0; i<5; i++ ){
   for( int j=0; j<5; j++ ){
     System.out.println("Inside for(j) loop"); //pesan1
     if( j == 2 ) continue outerLoop;
   }
   System.out.println("Inside for(i) loop"); //pesan2
}
```

Pada code di atas, **pesan2** tidak ditampilkan, karena ada continue outerloop statement, yang digunakan untuk iterasi(skip).

Return statement

Return statement

- Digunakan untuk keluar dari method.
- Mengikuti kontrol return dari statement pada method yang memanggilnya.



Return statement

Return value

Memberi nilai (atau sebuah ekspresi yang menghitung sebuah nilai) setelah keyword **return.**

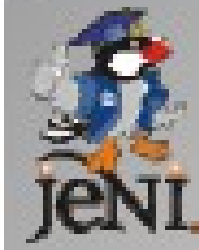
Contoh,

```
return ++count;
```

atau

```
return "Hello";
```

Tipe data dari nilai dikembalikan oleh return harus sama dengan tipe dari pendeklarasian nilai dari method yang memanggilnya.



Return statement

Ketikan sebuah method dideklasikan sebagai void, gunakan form yang tidak menggunakan nilai return.

Contoh,

```
return;
```

kesimpulan

- Struktur Kontrol Keputusan
  - if
  - if-else
  - if – else if
  - Switch
- Struktur Kontrol Pengulangan
  - while
  - do-while
  - For
- statement Branching
  - break
  - continue
  - return