STRUKTUR KONTROL

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

  digunakan 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.

CONTOH

```
public class statement_IF {
    public static void main(String [] args){
        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;
        . . .
    }
```

CONTOH

```
public class statement_IF_Else {
    public static void main(String [] args){
        int grade = 68;
        if( grade > 60 ){
            System.out.println("Selamat !");
        } else {
            System.out.println("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;
```

CONTOH

```
public class statement_IF_Else_IF {
    public static void main(String [] args){
        int Nilai = 95;
        if( Nilai > 90 ){
            System.out.println("Sangat Bagus!");
        }
        else if( Nilai > 60 ){
            System.out.println("Bagus!");
        }
        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.

CONTOH

```
public class statement_Switch {
    public static void main(String[] args) {
        char nilai = 'A';
        switch (nilai) {
            case 'C' : System.out.println("Nilai Anda Cukup");
            break;
            case 'B' : System.out.println("Nilai Anda Baik");
            break;
            case 'A' : System.out.println("Nilai Anda sangat Baik");
            break;
            default  : System.out.println("Nilai Anda Kosong");
            break;
        }
    }
}
```

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 ){
      statementI;
      statement2;
      . . .
}
```

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

CONTOH 1

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

```
public class statment_Do_While {
    public static void main(String [] args){
        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;Step
Ekspresi)
{
    statement1;
    statement2;
    . . .
}
```

- 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.println(i);
    i++;
}
```

FOR LANJUT

```
public class statement_For_2 {
    public static void main(String[] args) {
    int [] a = {1,2,3,4};
    for(int x = 0; x < a.length; x++) // basic for loop
        System.out.println(a[x]);
    for(int n : a) // enhanced for loop
        System.out.println(n);
    }
}
```

LATIHAN

```
Nama    : ---------------------
NPM     : 000000000000
Kelas   : A/B/C/D/E
=================================================
            APLIKASI NILAI AKADEMIK
=================================================
      Masukan Nilai Tugas1: 95
      Masukan Nilai Tugas2: 95
      Masukan Nilai Tugas3: 100
      Masukan Nilai Tugas4: 100
=================================================
      Nilai Tugas           : 97
      Masukan Nilai UTS     : 90
      Masukan Nilai UAS     : 95
=================================================
      Nilai Akhir           : 94
=================================================
      Anda Berhasil dengan Sangat Baik
                Grade = A
          ****************************
             ************************
```