BAB VIII

PERULANGAN MEMAKAI FOR

8.1. Pengantar

Cara ketiga membentuk perulangan adalah dengan memakai perintah **for (…; ….; ….).** Perintah perulangan **for** mempunyai fungsi dan struktur mirip dengan perintah perulangan **while**. Dengan demikian perintah perulangan for tidak menambah kehebatan bahasa pemrogaman Java. Meskipun demikian, ada banyak kasus persoalan di mana penyelesaian programnya secara intuitif lebih mudah diselesaikan memakai **for**. Selain itu, banyak pemrogram merasa lebih nyaman memakai **for** ketimbang **while** karena semua kendali perulangan dapat diletakkan di bagian awal perulangan.

8.2. Perulangan Memakai for ( )

Perintah perulangan memakai **for** memang lebih mudah ditulis dibanding memakai perintah **while**. Secara umum perintah while mempunyai struktur berikut:

```
inisiasi variable perulangan;
while ( syarat perulangan ) {
        blok-pernyataan
        ubah nilai variabel perulangan;
}
```

Berikut adalah contoh perhitungan saldo simpanan setelah mendapat bunga tahunan selama 5 tahun.

```
tahun = 0; // inisiasi nilai variabel perulangan
while ( tahun < 5 ) { // syarat perulangan
      bunga = modal * persenBunga; // hitung besar bunga
      modal = modal + bunga; // hitung modal plus bunga
      System.out.println(modal);
      tahun++; // lakukan pembaharuan nilai variabel perulangan
}
```

Perulangan di atas dapat dibuat memakai perintah **for** yang ekuivalen sebagai berikut:

```
for ( tahun = 0; tahun < 5; tahun++ ) {
    bunga = modal * persenBunga; // hitung besar bunga
    modal = modal + bunga; // hitung modal plus bunga
    System.out.println(modal);
}
```

Perhatikan bahwa inisiasi variabel perulangan, syarat perulangan, serta perubahan variabel perulangan semuanya diletakkan di bagian awal perulangan mengikuti kata **for** yang ditaruh di dalam tanda kurung biasa. Hal ini menjadikan perintah perulangan memakai **for** lebih mudah ditulis dan dibaca.

Perintah perulangan loop dikerjakan *compiler* sebagaimana dengan perintah while. Inisiasi variabel perulangan dikerjakan sekali di awal. Pengecekan syarat perulangan dikerjakan setiap kali sebelum mengeksekusi blok perulangan. Apakah blok perulangan akan diekseskui atau tidak tergantung apakah syarat perulangan masih memenuhi (bernilai *true*) atau tidak (bernailai *false*). Sedangkan pengubahan nilai variabel perulangan akan dilakukan setiap kali setelah mengeksekusi blok perulangan.

Secara umum, struktur perintah perulangan memakai **for** adalah sebagai berikut:

```
for ( <inisiasi variabel perulangan> ; <syarat perulangan> ; <update variabel perulangan>) {
    <blok-pernyataan>
    }
```

Apabila blok pernyataan hanya memuat satu perintah/pernyataan maka *bisa tidak perlu* diletakkan ke dalam tanda kurung kurawal.

Perhatikan bahwa <syarat perulangan> haruslah berupa ekspresi Boolean yang mempunyai nilai kebenaran true atau false. Sedangkan <inisiasi variabel perulangan> biasanya berupa pernyataan penugasan tetapi secara umum dapat berupa sebarang pernyataan. Sementara itu, <update variabel perulangan> dapat berupa sebarang pernyataan tetapi biasanya berupa penambahan atau pengurangan nilai variabel perulangan meskipun dapat juga sebarang pernyataan. Ketiga penyataan dalam perintah for ini boleh kosong sehingga hanya berbentuk **for (; ;).** Hal ini berarti bahwa kita memiliki perintah perulangan terus-menerus tanpa henti seperti

halnya dalam bentuk **while (true).** Untuk itu, supaya perulangan berhenti maka di dalamnya harus ada perintah semisal **break**.

Perintah perulangan **for(; ;)** paling sering digunakan dalam situasi di mana jumlah perulangan ditentukan berdasar nilai bilangan bulat yang terletak di antara nilai awal dan nilai akhir tertentu. Perulangan seperti ini disebut dengan perulangan hitungan atau *counting loop*. Secara umum bentuk perulangan tersebut adalah sebagai berikut:

```
for ( < variabel > = < min > ; < variabel > <= < max > ; <variabel>++ ) {
        <blok-pernyataan>
}
```

Di dalam bentuk perulangan di atas, < min > dan < max > berupa ekspresi bilangan bulat atau sebuah konstanta bulat. Oleh karena itu selama perulangan bariabel akan berisi/bernilai masing-masing min, min+1, min+2, min +3, . . . , max. Nilai-nilai dalam <variabel> biasanya digunakan dalam <blok-pernyaataan> di perulangan tersebut.

Berikut adalah contoh sederhana penggunaan bentuk perulangan hitungan di atas untuk mencetak bilangan bulat dari 1 sampai dengan 20.

```
for (int i = 1 ; i <= 20 ; i++ ) {
        System.out.println ( i );
}
```

Karena perulangan di atas hanya memuat satu perintah maka dapat dituliskan tanpa harus memakai kurung kurawal sehingga cukup menjadi :

```
for (int i = 1 ; i <= 20 ; i++ ) System.out.println ( i );
```

Jika sebaliknya kita justru ingin mencetak bilangan bulat secara menurun dari 20 ke 1 maka perintah perulangan di atas cukup dimodifikasi menjadi :

```
for (int i = 20 ; i >= 1 ; i--) System.out.println ( i );
```

Sebenarnya struktur resmi dari perulangan memakai for () mengijinkan bahwa kendali perulangan dalam inisiasi variabel maupun di bagian perubahan nilai variabel dapat memuat lebih dari satu perintah yang masing-masing dipisahkan dengan koma. Berikut adalah contoh struktur perulangan

tersebut yang berfungsi untuk mencetak sekaligus bilangan bulat dari 1 sampai dengan 20 baik secara menaik maupun secara menurun.

```
for (int i = 1, j = 20 ; i <= 20 ; i++, j-- ) {
        System.out.printf (“5%”,  i );
        System.out.printf ( “5%”, j );
        System.out.println();
}
```

Berikut adalah contoh pemakaian perulangan memakai for () untuk menampilkan bilangan genap positif antara 2 sampai dengan 20. Perhatikan bahwa ada banyak cara untuk yang dapat dilakukan dan di sini diberikan 3 cara yang berbeda:

```
// Ada sepuluh bilangan genap yang ditampilkan
// Gunakan for untuk menyimpan  1, 2, …, 10 lalu kalikan masing-masing dengan
//  bilangan bulat 2 sehingga  2*1, 2*2, ... 2*10.

for (N = 1; N <= 10; N++) {
        System.out.println( 2*N );
}


// Gunakan langsung perintah for() untuk menghitung 2, 4, ..., 20
// yakni menambah N dengan 2

for (N = 2; N <= 20; N = N + 2) {
        System.out.println( N );
}


// Gunakan semua bilangan dari 2, 3, 4, ..., 19, 20,
// tetapi yang dicetak hanya yang berupa bilangan genap

for (N = 2; N <= 20; N++) {
        if ( N % 2 == 0 ) // Bila N genap maka cetaklah N
        System.out.println( N );
}
```

8.3. Praktikum

1. Jalankan program berikut dan kuasai sintaknya

```
public class ForDemo {
public static void main(String[] args){
for (int i=1; i<11; i++){
        System.out.println("Nilai i adalah : " + i);
        }
}
}
```

2. Jalankan dan analisalah mengapa outputnya menjadi seperi itu

```
public class CobaFor {
 public static void main(String[] args) {
        String bintang="";
        for(int i=1; i < 11; i++){
                bintang=bintang+"*";
                System.out.println(bintang);
                }
        }
}
```

3. Jalankan program untuk menjumlah bilangan bulat positif kelipatan 5 berikut

```
        public class CobaLagiFor {
         public static void main(String[] args) {
        int jumlah=0;
        for(int i=0; i <= 100; i=i+5){
                jumlah = jumlah + i;
                }
        System.out.println("Jumlah 5+10+ …+100 = "+jumlah);
        }
}
```

4. Buat program memakai for yang dapat menghitung tahanan (resistor) total dari N buah tahanan yang dimasukkan lewat keyboard. Sebelum tahanan total dihitung user ditanya oleh program apakah tahanan akan disusun seri ataukah paralel. Jika r1, r2, …., rN adalah tahanan yang dibaca komputer maka tahanan total (rt) dihitung dengan rumus

   Jika tahanan disusun seri maka rt = r1 + r2 + . . . + rN

   Jika tahanan disusun paralel maka 1/rt = 1/r1 + . . . + 1/rN

5. Buat program memakai perulangan for untuk menghitung besarnya beberapa nilai yang dihitung dari N buah data real (double) yang dimasukkan lewat keyboard. Beberapa nilai tersebut antara lain

   a) Jumlah kuadrat data yakni Σx_i^2

   b) Jumlah akar data yakni $\Sigma (\sqrt{x_i})$

   c) Jumlah 1/data untuk data yang tidak sama dengan nol yakni $\Sigma (1/x_i)$

6. Buat program yang dapat menampilkan deret Fibonacci sebanyak N suku memakai perulangan for . Berikut adalah 20 suku deret Fibonacci :

   1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946
   Deret angka ini diawali dengan dua angka 1 lalu diikuti dengan 2 dan kemudian penjumlahan dari kedua angka menghasilkan deretan angka yang berikutnya. 1+2 muncul angka 3, lalu 2+3 muncul angka 5, kemudian 3+5 muncul angka 8 dan seterusnya.

7. Buat program untuk menghitung nilai $y = a x^2 + b x + c$ untuk x mulai dari x1 sampai dengan x2 dengan pertambahan 0.25 di mana x1 dan x2 adalah akar dari persamaan $a x^2 + b x + c = 0$. Nilai dari koefisien a, b dan c dimasukkan lewat keyboard. Nilai y hanya dihitung apabila persamaan tersebut mempunyai dua akar yakni diskriminan (D) > 0. Tampilan hasil program Anda adalah seperti berikut

Dari persamaan y = … x2 + ….x + ….
Akar-akarnya adalah …… dan …….
Nilai y untuk beberapa x antara akar pertama dan kedua adalah

---

x	*y = ….x2 + …x + …..*
……	……
……	……
……	……
……	……

---

8. Buat program untuk menampilkan formasi bintang berikut memakai for

```
*****
*****
*****
*****
*****
```

9. Buat program untuk menampilkan formasi bintang berikut memakai for

```
*
**
***
****
*****
```

10. Buat program untuk menampilkan formasi bintang berikut memakai for

```
*****
****
***
**
 *
```