BAB VII

PERULANGAN MEMAKAI DO – WHILE

7.1. Pengantar

Perulangan dapat juga dibentuk memakai perintah pasangan **do** dan **while**. Perulangan memakai perintah ini memiliki struktur yang mirip dengan perulangan memakai **while**. Perbedaan utama antara perulangan memakai **while** dan **do –while** terletak pada dua hal. Hal pertama menyangkut kapan perulangan dilakukan dan kedua menyangkut letak syarat perulangan dituliskan.

Perulangan memakai **do-while** akan otomatis mengerjakan blok perintah yang diulang sekali dahulu, baru kemudian syarat perulangan dievaluasi apakah masih *true* atau sudah *false.* Jika syarat perulangan masih *true* maka blok perintah tersebut akan diulang terus sampai syarat perulangan bernilai *false*. Hal ini agak berbeda dengan perintah **while** karena di perintah **while**, *compiler* akan langsung mengecek syarat perulangan apakah bernilai *true* atau *false*. Jika syarat bernilai *true* maka blok perulangan akan dieksekusi. Dengan demikian bisa saja blok perulangan tidak akan pernah dieksekusi ketika syarat perulangan langsung bernilai *false*. Hal inilah yang membedakan dengan perulangan memakai **do-while** karena bila memakai **do-while** maka blok perulangan pasti akan dieksekui minimal sekali. Karena blok perulangan otomatis diesekusi minimal sekali maka perulangan memakai **do – while** akan menempatkan syarat perulangan di akhir blok perintah yang diulang.

Perintah perulangan memakai do-while berguna untuk mengatasi masalah yang terkait dengan penjaminan validitas atau kebenaran pemasukan data oleh user. Misalnya dalam program statistika sederhana akan dibuat program yang lebih tangguh maka sebelum pemasukan data dilakukan, banyak data (N) yang akan dioleh harus dipastikan terlebih dahulu bahwa nilainya berupa bilangan bulat positif. Untuk itu dapat digunakan perintah **do-while** ini yakni **do (kerjakan)** membaca N dari keyboard **while** (selama) N masih belum positif. Meskipun demikian, pada dasarnya setiap perulangan yang dibentuk memakai while dapat dinyatakan memakai do-while dan sebaliknya, seperti yang nanti akan dicontohkan. Akibatnya kapan anda memakai **while**

atau **do – while** lebih sering dilatarbelakangi manakah perintah yang lebih Anda kuasai karena menggunakan perintah yang betul=betul Anda kuasai akan mengurangi resiko kesalahan program.

7.2. DO – WHILE

Perulangan memakai **do – while** mempunyai struktur atau format sbb:

```
do {
  blok perulangan
 }  while ( ekspresi boolean);
```

Perhatikan akan adanya tanda titik-koma (;) di akhir ekspresi boolean. Hal ini merupakan keharusan karena **do-while** adalah satu kesatuan perintah yang harus diakhiri titik koma.

Perintah perulangan memakai **do – while** di atas akan mengakibatkan *compiler* mengerjakan blok perulangan sekali terlebih dahulu. Setelah itu *compiler* akan mengecek apakah ekspresi Boolean yang berada di samping **while** bernilai *true* atau *false*. Jika ekspresi Boolean bernilai *true* maka blok perulangan akan dikerjakan lagi, demikian seterusnya sampai ekspresi boolean bernilai *false*.

Andaikan dalam program statistik sederhana yang pernah anda buat, Anda harus memastikan bahwa banyak data yang akan diolah yakni N harus berupa bilangan bulat positif maka pembacaan data N tersebut dapat dilakukan memakai perulangan **do – whule** berikut:

```
Scanner tombol = new Scanner(System.in);
do {
        System.out.print("Masukkan banyak data : ");
        N = tombol.nextInt();
} while ( N <= 0);
```

Perhatikan bahwa perintah membaca N dari keyborad di atas dapat juga dinyatakan memakai perintah **while** seperi berikut:

```
 Scanner tombol = new Scanner(System.in);
System.out.print("Masukkan banyak data : ");
N = tombol.nextInt();
while ( N <= 0) {
        System.out.print("Masukkan banyak data lagi yang lebih dari nol : ");
        N = tombol.nextInt();
}
```

Dengan demikian, secara umum perintah **do – while** berikut

```
do      {
  blok perulangan
}  while ( ekspresi boolean);
```

Dapat diubah ke perintah yang ekuivalen memakai while berikut

```
{
blok perulangan
}
while ( ekspresi boolean){
  blok perulangan
}
```

Demikian pula sebaliknya, perulangan memakai while dengan bentuk

```
while (ekspresi boolean) {
        blok perulangan
}
```

Dapat diubah ke bentuk yang ekuivalen memakai do – while berikut

```
if (ekpresi boolean) {
        blok perulangan
do      {
        blok perulangan
} while (ekspresi boolean);
}
```

7.3. Perintah break

Perulangan yang dibentuk memakai **while** maupun **do -while** akan berakhir jika ekspresi boolean yang ada bernilai *false*. Akan tetapi perulangan juga dapat dihentikan dengan cara lain yakni memakai perintah **break** seperti halnya yang dipakai dalam perintah **switch-case**. Jika *compiler* bertemu dengan **break** maka perintah setelah perulangan yang selanjutnya akan dikerjakan. Dengan demikian, perintah membaca N sampai dengan bernilai lebih dari nol dapat dibentuk memakai perintah **while** berikut

```
Scanner tombol = new Scanner(System.in);
while ( true ) { // seolah berputar tanpa henti
        Sytem.out.print("Masukkan banyak data : ");
        N = tombol.nextInt();
        if (N > 0)
                break;
        Sytem.out.print("Data yang Anda masukkan harus lebih besar dari nol ! ");
}
// apabila break dikerjakan maka compiler mengerjakan perintah setelah baris ini
```

7.4. Contoh-contoh

1. Program untuk menampilkan angka 1, 2, 3, ….., 15 di mana satu baris berisi satu angka.

```
public class Looping1 {
public static void main(String[] args) {
        int  bilangan = 1;
        do {
                System.out.println(bilangan);
                bilangan = bilangan + 1;
        } while ( bilangan < 16 );
    }
}
```

2. Program untuk menampilkan angka 1, 2, 3, ….., 20 dalam satu baris

```
public class Looping2 {
public static void main(String[] args) {
        int  bilangan = 1;
         do {
                System.out.print(bilangan+ “,     “);
                bilangan = bilangan + 1;
         } while ( bilangan < 21 );
    }
}
```

3. Program untuk menghitung jumlah dari 5 + 10 + …. + 100

```
public class Looping3 {
public static void main(String[] args) {
    int jumlah = 0, bilangan = 0;
    do {
        jumlah = jumlah + bilangan;
        bilangan = bilangan + 5;
        } while ( bilangan <= 100 );
        System.out.println(“ Jumlah dari 5+10+…+100 =  “+  jumlah);
    }
}
```

7.5. Praktikum

1. Buat program yang dapat menentukan jumlah setiap jenis kualitas mangga berdasarkan beratnya. Ada 3 jenis kualitas mangga yakni ‘BIASA’, ‘BAGUS’ dan ‘UNGGUL’. Ketiga jenis ini menentukan apakah mangga tersebut akan diekspor, dijual di dalam negeri atau akan dibuat juice. Program Anda pertama-tama membaca jumlah mangga yang akan diproses lalu secara berulang program membaca berat mangga satu persatu. Berdasarkan berat tersebut program menentukan ada berapa mangga yang berkualitas BIASA, B AGUS dan UNGGUL memakai ketentuan berikut:

   BIASA : berat kurang dari 500 gram

   BAGUS : berat lebih besar atau sama dengan 500 gram tetapi kurang dari 750 gram

   UNGGUL : berat lebih dari 750 gram.

   Karena berat mangga harus positif (> 0) maka program Anda harus membuat perintah untuk mengulang pemasukan berat mangga apabila berat yang dimasukkan user masih belum positif memakai perintah do-while.

2. Deret Fibonacci berbentuk demikian : 0, 1, 1, 2, 3, 5, 8, 13, …. yakni bilangan ke n diperoleh dari jumlah bilangan ke (n-1) dan bilangan ke (n-2) sehingga bilangan 8 berasal dari 5 + 3. Buat program yang dapat menampilkan sejumlah N buah deret Fibonacci yang pertama di mana besarnya N dimasukkan lewat keyboard. Misalnya N diisi 9 maka hasil/ouput program Anda adalah 0, 1, 1, 2, 3, 5, 8, 13, 21

3. Buat program game sederhana berikut. Game ini akan menebak besarnya bilangan bulat (integer) random yang dihasilkan oleh komputer.

   Perintah bilKom = (int) (100 * Math.random() + 1)

   akan menyimpan hasil bilangan bulat random antara 1 sampai dengan 100 ke memori bernama bilKom.

   Program akan meminta user menebak besarnya bilangan random tersebut. Jika tebakan benar yakni bilUser = bilKom maka program memberi ucapan selamat ‘Anda Menang !!!’ dan game over. Tetapi jika tebakan belum benar maka program akan memberi tahu kepada user apakah tebakannya lebih besar atau lebih kecil dari bilKom dan kemudian program

mempersilahkan user untuk memasukkan tebakannya lagi. Proses ini diulang-ulang tetapi sampai tebakannya benar atau kalau sudah menebak maksimum 10 kali dan tetap tidak benar maka game over dengan pesan ‘Anda Kalah !!!’

Perintah sisiDadu = (int) (6 * Math.random() + 1) akan menghasilkan bilangan random bulat antara 1 sampai dengan 6. Perintah ini dapat dipakai untuk mensimulasikan pelemparan sebuah dadu. Sekarang andaikan Anda punya dua dadu sehingga Anda punya sisDadu1 dan sisiDadu2. Buat program yang dapat menghitung berapa kali lemparan 2 buah dadu dibutuhkan supaya hasil lemparannya berupa 6 dan 6 yakni sisiDadu1 = sisiDadu2 dan keduanya berisi 6. Modifikasi program Anda sehingga program dapat menampilkan jumlah lemparan dadu untuk berapapun hasil sisi dadu yang diinginkan oleh user Anda.