Bab 3

Perulangan

Tujuan

1. Memahami Konsep Perulangan dalam Pemrograman
2. Menggunakan Perulangan untuk Control Flow program
3. Memahami penggunaan Jump Statement

Perulangan merupakan konsep pemrograman untuk melakukan perintah secara berulang berdasarkan kondisi tertentu tanpa perlu menuliskan kodenya secara berulang. Dalam Java terdapat 3 jenis perulangan yaitu *for loop*, *while loop*, dan *do while loop*.

3.1 For Loop

For loop atau juga dikenal sebagai *Counted Loop* adalah perulangan yang jumlah perulangannya terhitung, perulangan ini terdiri dari inisialisasi, kondisi, dan iterasi.

```
for (inisialisai; kondisi; iterasi) {
    // body: perintah yang dijalankan
}
```

Drawing 15: Struktur For Loop

Berikut contoh program menggunakan for loop:

```
public static void main (String[] args) {
    int a = 0;
    for (int i = 0; i < 10; i++) {
         System.out.print(a + " ");
         a += 2;
    }
}
```

Drawing 16: Program Untuk Mencetak 10 Digit Bilangan Kelipatan 2

Variabel `int i` di atas adalah *iterator* atau penyebab terjadinya perulangan, dengan inisialisasi nol, kemudian perulangan akan berlangsung selama `i` kurang dari 10, dan iterasi dilakukan dengan meningkatkan nilai `i` sebanyak 1 poin.

3.2 While Loop

While Loop atau *Uncounted Loop* merupakan bentuk lain dari perulangan dengan struktur yang lebih sederhana dari *for loop* dan perulangan ini lebih cocok digunakan ketika perulangan tersebut membutuhkan kondisi yang tidak perlu dihitung. Struktur dasar While Loop hanya terdiri dari Kondisi dan *body*.

```
public static void main (String[] args) {
    Scanner in = new Scanner(System.in);
    int answer = 9;
    boolean guess = false;
    while (!guess) {
         int try = in.nextInt();
         guess = try == answer;
    }
    System.out.println(“end”);
}
```

Drawing 17: Program Untuk Menebak Bilangan

3.3 Do While Loop

Do While adalah *while loop* yang pengecekan kondisinya dilakukan setelah perintah pada bagian *body* dijalankan, sehingga sekalipun kondisi tidak terpenuhi, minimal program akan berjalan sekali sebelum akhirnya diterminasi. Berikut Struktur dasar *do while loop.*

```
public static void main (String[] args) {
    int i = 10;
    do {
         System.out.print(i);
         i++;
    } while (i < 10);
    System.out.println(“end”);
}
```

Drawing 18: Program Akan Tetap Mencetak Angka 10 Satu Kali

3.4 Jump Statement

Jump Statement adalah perintah untuk mengubah alur suatu program, entah itu memberhentikan satu blok (***break***), melangkahi blok selanjutnya (***continue***) atau keluar dan mengembalikan nilai suatu method (***return***), salah satu contoh penggunaan *jump statement* adalah *break* yang terdapat pada *switch case*.

3.4.1 Break

Seperti namanya, *break* berfungsi untuk menghentikan alur sebuah program, namun yang dihentikan hanya satu blok yang membungkus *break* tersebut, contoh penggunaan *break* dapat dilihat pada *switch case.*

3.4.2 Continue

Perintah *Continue* akan melangkahi perintah yang ada setelahnya, umumnya *continue* digunakan di dalam perulangan, yang mana ketika terdapat *continue*, maka program akan kembali melakukan iterasi selanjutnya tanpa menjalankan perintah setelah *continue*.

```
public static void main (String[] args) {
    Scanner in = new Scanner(System.in);
    int answer = 9;
    boolean guess = false;
    while (!guess) {
          int try = in.nextInt();
          if (try == 9) {
                continue;
          }
          guess = try == answer;
    }
    System.out.println(“end”);
}
```

Drawing 19: Akan Terjadi Infinity Loop Pada Program di atas

3.4.3 Return

Return merupakan perintah untuk mengembalikan nilai dan keluar dari suatu *method*, jika di *methodA* terdapat *return*, lalu *methodB* memanggil *methodA*, maka alur program akan dikembalikan ke *methodB*. Untuk lebih lanjut akan dibahas pada bab 5.