Algoritma dan Pemrograman Komputer 1

Bab 7: Perulangan (Looping)

Aslam Pandu Tasminto – 5002241025
M. Ma'ruf Qomaruddin Kafi – 5002241095

October 20, 2025

Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember

Daftar Isi

Pengenalan Perulangan

Konsep Perulangan dalam Java

Definisi

Perulangan (looping) adalah struktur kontrol yang digunakan untuk mengeksekusi blok kode secara berulang-ulang selama kondisi tertentu terpenuhi.

Jenis Perulangan dalam Java

- `for` loop - ketika jumlah perulangan diketahui
- `while` loop - ketika jumlah perulangan tidak pasti
- `do-while` loop - minimal 1x eksekusi dijamin

Manfaat Perulangan

- Mengurangi duplikasi kode
- Efisiensi dalam penulisan program
- Memudahkan proses iterasi data

FOR Loop

Sintaks dan Struktur FOR Loop

Struktur Dasar FOR Loop

```
for (inisialisasi; kondisi; perubahan) {
    // pernyataan yang diulang
}

```

Komponen	Deskripsi
`inisialisasi`	Inisialisasi variabel counter (contoh: `int i = 0`)
`kondisi`	Kondisi untuk melanjutkan perulangan (contoh: `i < 5`)
`perubahan`	Perubahan nilai counter tiap iterasi (contoh: `i++`)
`pernyataan`	Blok kode yang dieksekusi berulang kali

Tabel 1: Komponen-komponen dalam FOR Loop

Contoh FOR Loop Sederhana

Contoh Program FOR Loop Dasar

```
1 public class ForLoop {
2     public static void main(String[] args) {
3         // Menampilkan "Selamat datang" 5 kali
4         for (int i = 0; i < 5; i++) {
5             System.out.println("Selamat datang");
6         }
7     }
8 }
9
```

Output Program

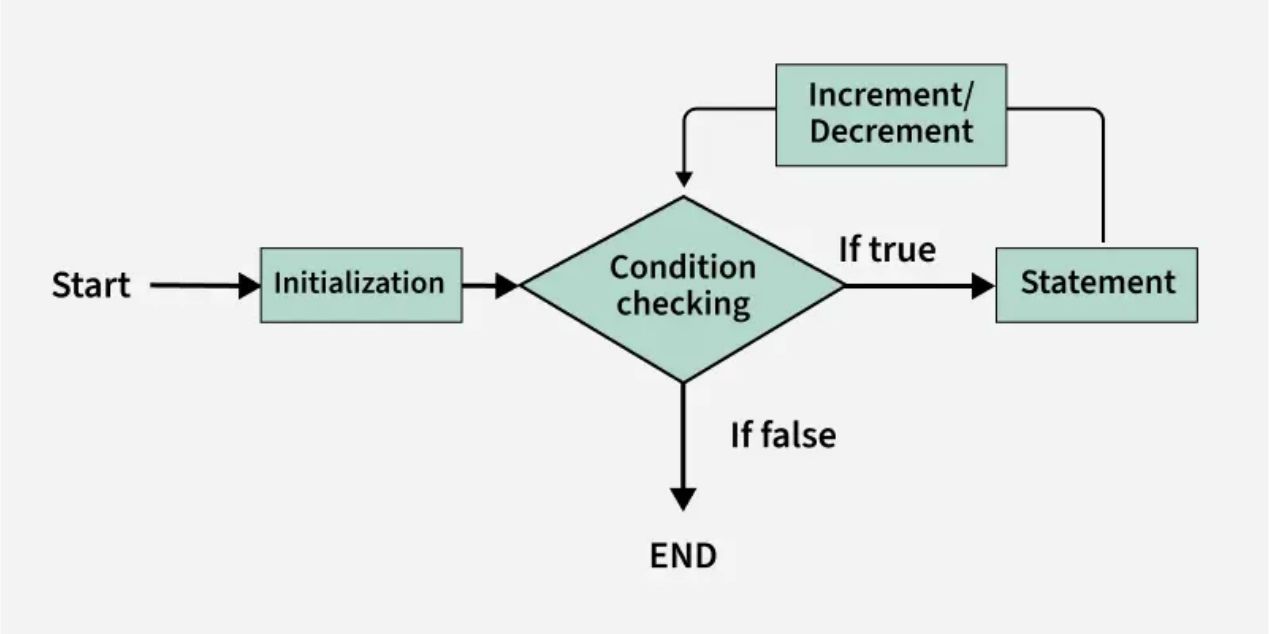
```
Selamat datang
Selamat datang
Selamat datang
Selamat datang
Selamat datang
```

Variasi FOR Loop

FOR Loop dengan Pola Berbeda

```
1  public class ForVariasi {
2      public static void main(String[] args) {
3          // Decrement
4          for(int i = 5; i > 0; i--) {
5              System.out.println(i);
6          }
7
8          // Kelipatan 2
9          for(int j = 0; j < 10; j = j + 2) {
10             System.out.println(j);
11         }
12
13         // Perkalian
14         for(int k = 1; k < 17; k = k * 2) {
15             System.out.println(k);
16         }
17     }
18 }
19
```

Flowchart FOR Loop



Gambar 1: Alur eksekusi FOR Loop dalam Java

Sumber: GeeksforGeeks - Java Loops

WHILE Loop

Sintaks dan Struktur WHILE Loop

Struktur Dasar WHILE Loop

```
1 while (kondisi) {
2     // pernyataan yang diulang
3 }
4
```

Komponen	Deskripsi
`kondisi`	Ekspresi boolean yang dievaluasi sebelum setiap iterasi
`pernyataan`	Blok kode yang dieksekusi berulang selama kondisi true

Tabel 2: Komponen-komponen dalam WHILE Loop

Karakteristik WHILE Loop

- Kondisi diperiksa **sebelum** eksekusi blok pernyataan
- Jika kondisi false sejak awal, blok tidak akan dieksekusi sama sekali
- Cocok untuk kasus dimana jumlah iterasi tidak diketahui
- Pastikan ada mekanisme untuk mengubah kondisi atau akan terjadi **infinite loop**!

Contoh WHILE Loop

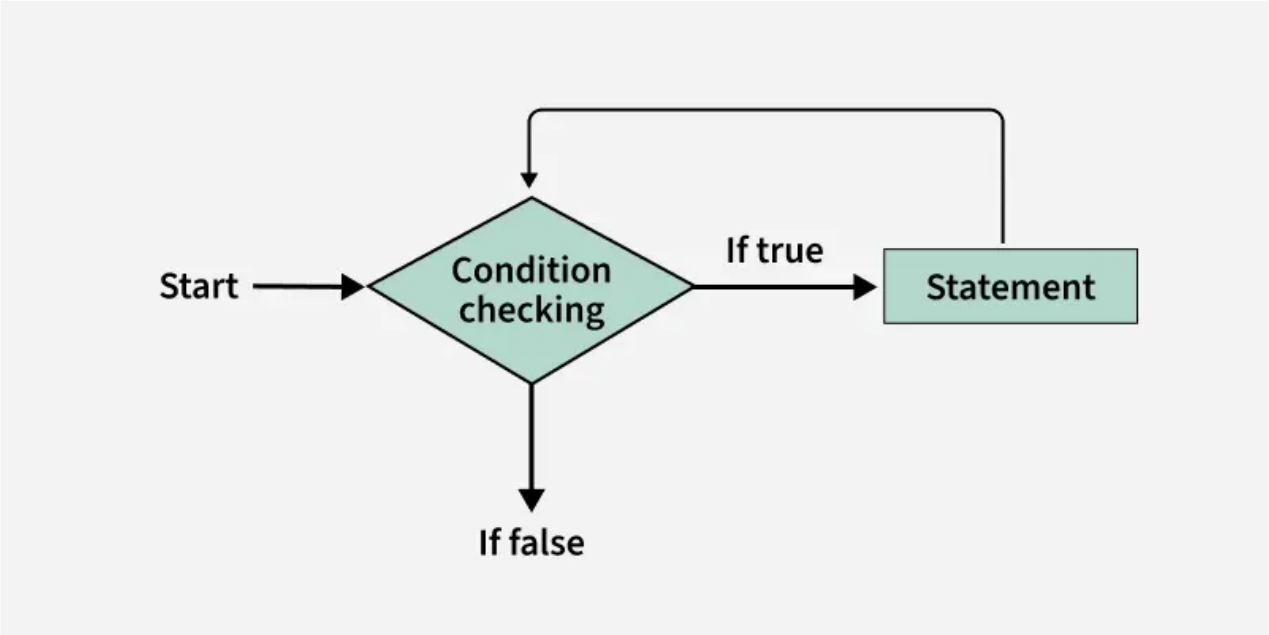
Contoh Program WHILE Loop

```
1  public class WhileLoop {
2      public static void main(String[] args) {
3          int i = 0;
4
5          // Menampilkan "Selamat datang" 5 kali
6          while (i < 5) {
7              System.out.println("Selamat datang");
8              i++;  // Jangan lupa increment!
9          }
10     }
11 }
12
```

Output Program

```
Selamat datang

Selamat datang

Selamat datang

Selamat datang

Selamat datang
```

Flowchart WHILE Loop



Gambar 2: Alur eksekusi WHILE Loop dalam Java

Sumber: GeeksforGeeks - Java Loop

DO-WHILE Loop

Sintaks dan Struktur DO-WHILE Loop

Struktur Dasar DO-WHILE Loop

```
1 do {
2     // pernyataan yang diulang
3 } while (kondisi);
4
```

Komponen	Deskripsi
`do`	Keyword yang menandai awal blok pernyataan
`pernyataan`	Blok kode yang dieksekusi minimal sekali
`while (kondisi)`	Kondisi yang diperiksa setelah eksekusi blok

Tabel 3: Komponen-komponen dalam DO-WHILE Loop

Karakteristik DO-WHILE Loop

- Blok pernyataan dieksekusi **minimal sekali**
- Kondisi diperiksa **setelah** eksekusi blok pernyataan
- Cocok untuk kasus yang membutuhkan eksekusi minimal sekali sebelum pengecekan

Contoh DO-WHILE Loop

Contoh Program DO-WHILE Loop

```
1  public class DoWhileLoop {
2      public static void main(String[] args) {
3          int i = 0;
4
5          // Menampilkan "Selamat datang" 5 kali
6          do {
7              System.out.println("Selamat datang");
8              i++;
9          } while (i < 5);
10     }
11 }
12
```

Output Program

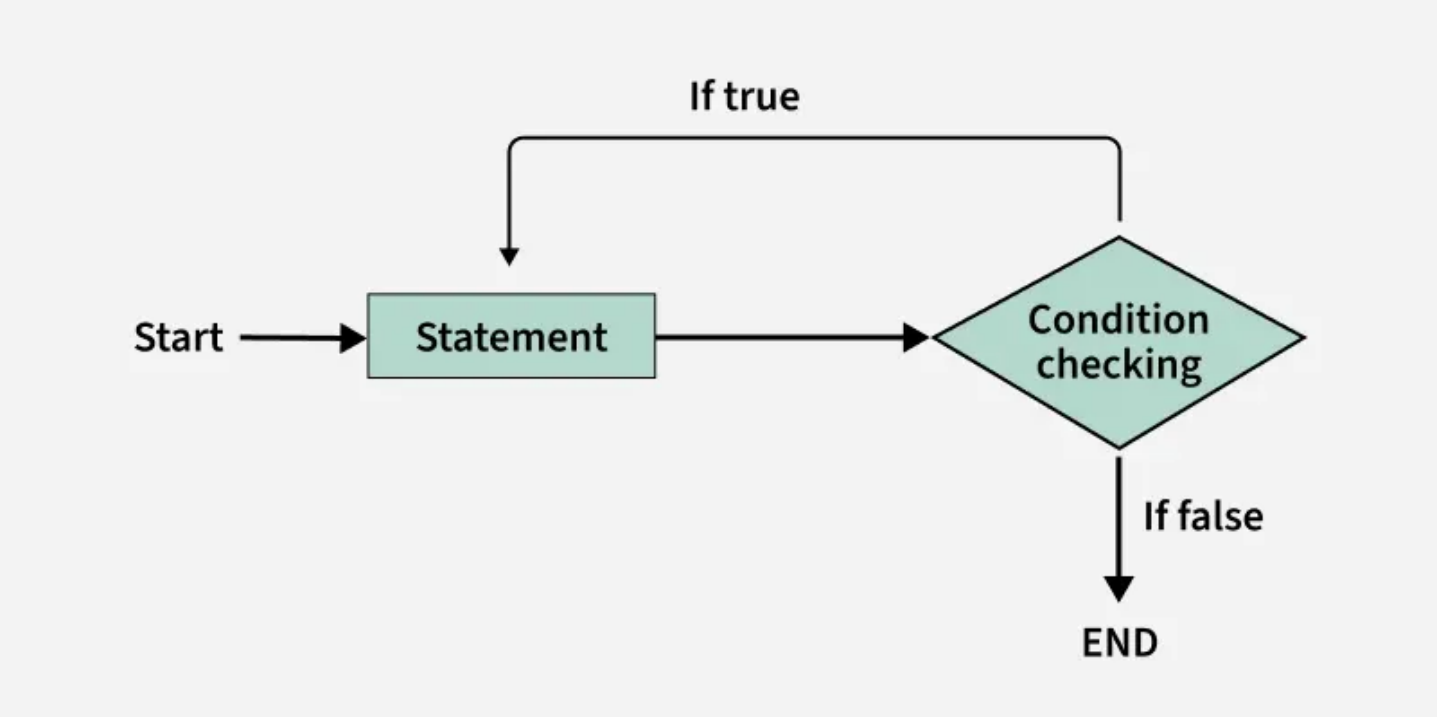
```
Selamat datang
Selamat datang
Selamat datang
Selamat datang
Selamat datang
```

Perbedaan WHILE vs DO-WHILE

Contoh Kondisi Awal False

```java
public class PerbedaanLoop {
    public static void main(String[] args) {
        boolean kondisi = false;
        int x = 0, y = 0;

        // WHILE - tidak dieksekusi
        while(kondisi) {
            x++;
        }

        // DO-WHILE - dieksekusi sekali
        do {
            y++;
        } while(kondisi);

        System.out.println("x = " + x); // Output: x = 0
        System.out.println("y = " + y); // Output: y = 1
    }
}

```

Flowchart DO-WHILE Loop



Gambar 3: Alur eksekusi DO-WHILE Loop dalam Java

Sumber: GeeksforGeeks - Java Loop

Perbandingan Jenis Perulangan

Perbandingan FOR, WHILE, dan DO-WHILE

Kriteria	FOR	WHILE	DO-WHILE
Pengecekan kondisi	Awal	Awal	Akhir
Jumlah eksekusi minimal	0	0	1
Inisialisasi counter	Dalam statement	Diluar loop	Diluar loop
Cocok untuk	Iterasi diketahui	Iterasi tidak pasti	Minimal 1x eksekusi
Contoh penggunaan	Array, deret	Input validation	Menu program

Tabel 4: Perbandingan Jenis Perulangan dalam Java

Kapan Menggunakan Masing-masing Loop?

FOR Loop

```
1 // Ketika jumlah iterasi diketahui
2 for(int i = 0; i < array.length; i++) {
3     System.out.println(array[i]);
4 }
5
```

WHILE Loop

```
1 // Ketika jumlah iterasi tidak pasti
2 while(scanner.hasNextInt()) {
3     int num = scanner.nextInt();
4     // process number
5 }
6
```

DO-WHILE Loop

```
1 // Ketika perlu eksekusi minimal sekali
2 do {
3     System.out.print("Masukkan password: ");
4     password = scanner.nextLine();
5 } while(!isValidPassword(password));
6
```

Nested Loops

Konsep Nested Loops

Definisi

Nested loops (perulangan bersarang) adalah perulangan yang berada di dalam perulangan lainnya.

Contoh Program Nested FOR Loop

```
1  public class NestedFor {
2      public static void main(String[] args) {
3          for(int i = 0; i < 3; i++) {
4              System.out.println("Iterasi luar: " + i);
5
6              for(int j = 0; j < 2; j++) {
7                  System.out.println("  Iterasi dalam: " + j);
8              }
9          }
10     }
11 }
12
```

Nested Loops untuk Pola Bintang

Contoh Program Segitiga Bintang

```
1  public class PolaBintang {
2      public static void main(String[] args) {
3          int tinggi = 5;
4
5          for(int i = 1; i <= tinggi; i++) {
6              for(int j = 1; j <= i; j++) {
7                  System.out.print("*");
8              }
9              System.out.println();
10         }
11     }
12 }
13
```

Output Program

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

Loop Control Statements

BREAK Statement

Fungsi BREAK

Menghentikan eksekusi loop secara paksa dan keluar dari loop.

Contoh Program BREAK dalam Loop

```
1  public class BreakExample {
2      public static void main(String[] args) {
3          for(int i = 0; i < 10; i++) {
4              if(i == 5) {
5                  break;  // Keluar loop ketika i = 5
6              }
7              System.out.println("i = " + i);
8          }
9          System.out.println("Loop selesai");
10     }
11 }
12
```

Output Program

```
i = 0
i = 1
i = 2
i = 3
i = 4
Loop selesai
```

CONTINUE Statement

Fungsi CONTINUE

Melewati iterasi saat ini dan melanjutkan ke iterasi berikutnya.

Contoh Program CONTINUE dalam Loop

```
1  public class ContinueExample {
2      public static void main(String[] args) {
3          for(int i = 0; i < 10; i++) {
4              if(i % 2 == 0) {
5                  continue;  // Lewati bilangan genap
6              }
7              System.out.println("i = " + i);
8          }
9      }
10 }
11
```

Output Program

```
i = 1
i = 3
i = 5
i = 7
i = 9
```

Enhanced FOR Loop

Enhanced FOR Loop (For-Each)

Konsep For-Each

Penyederhaan loop khusus untuk iterasi melalui array

Konsep array akan dipelajari di Modul 10, ini hanya sebagai pengantar

Contoh Program For-Each dengan Array

```
1  public class ForEachExample {
2      public static void main(String[] args) {
3          int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
4
5          // Enhanced for loop
6          for(int number : numbers) {
7              System.out.println(number);
8          }
9      }
10 }
11
```

Keunggulan For-Each

- Lebih sederhana dan mudah dibaca
- Tidak perlu mengelola index manual
- Tidak mungkin terjadi index out of bounds (index yang melebihi batas)

Best Practices

Best Practices Penggunaan Loop

Tips dan Rekomendasi

- **Pilih loop yang tepat** sesuai kebutuhan
- **Hindari infinite loops** dengan memastikan kondisi berakhir
- **Gunakan nama variabel yang bermakna** untuk counter
- **Minimalkan perhitungan dalam kondisi** loop
- **Gunakan enhanced for** ketika iterasi array
- **Hati-hati dengan nested loops**

Kesalahan Umum

- Lupa increment counter (loop tak berakhir)
- Kesalahan perhitungan satu angka
- Modifikasi counter dalam badan loop
- Loop bersarang yang terlalu dalam

Latihan

Latihan 1: Deret Fibonacci

Soal 1: Deret Fibonacci

Buat program yang meminta input berupa integer positif n, kemudian menampilkan n suku pertama deret Fibonacci.

Rumus Fibonacci:

- F(1) = 1
- F(2) = 1
- F(n) = F(n-1) + F(n-2) untuk n >2

Contoh: Input 7 → Output: 1, 1, 2, 3, 5, 8, 13

Latihan 2: Bilangan Prima

Soal 2: Cek Bilangan Prima

Buat program yang meminta input berupa integer positif, kemudian menentukan apakah bilangan tersebut prima atau bukan.

Ciri bilangan prima:

- Hanya habis dibagi 1 dan dirinya sendiri
- Bilangan >1
- Contoh: 2, 3, 5, 7, 11, 13, ...

Contoh:

- Input 7 $\rightarrow$ Output: "7 adalah bilangan prima"
- Input 9 $\rightarrow$ Output: "9 bukan bilangan prima"

Latihan 3: Pola Segitiga Angka

Soal 3: Segitiga Angka

Buat program yang meminta input berupa integer n, kemudian menampilkan pola segitiga angka seperti berikut:

Contoh n = 4

Output:

```
1
12
123
1234
```

Latihan 4: Menghitung Faktorial

Soal 4: Menghitung Faktorial

Buat program yang meminta input berupa integer positif n, kemudian menghitung dan menampilkan nilai n! (n faktorial).

Rumus Faktorial:

- $n! = 1 \times 2 \times 3 \times ... \times n$
- $0! = 1$ (menurut definisi)
- Tidak terdefinisi untuk bilangan negatif

Contoh:

- $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$
- $0! = 1$

Kesimpulan

Kesimpulan

Inti Bab 7: Perulangan (Looping)

- **3 jenis loop**: FOR, WHILE, DO-WHILE
- **FOR**: jumlah iterasi diketahui
- **WHILE**: kondisi kompleks
- **DO-WHILE**: eksekusi minimal sekali
- **Control**: BREAK (berhenti) dan CONTINUE (lewati)
- **Nested loops**: pola dan data multidimensi
- **Enhanced for**: iterasi array lebih mudah

Referensi

Referensi

Referensi:

- **Modul Praktikum Algoritma dan Pemrograman - Modul 7**
  Departemen Matematika FSAD ITS
- **Oracle Java Tutorials - Control Flow Statements**
  `https://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html`
- **GeeksforGeeks - Loops in Java**
  `https://www.geeksforgeeks.org/loops-in-java/`

Terima Kasih

Pertanyaan dan Diskusi