

**LAPORAN PRAKTIKUM PEKAN 5**



**MATA KULIAH ALGORITMA DAN PEMROGRAMAN**

**DOSEN PENGAMPU :**

**WAHYUDI, S.T M.T**

**OLEH :**

**MUHAMMAD HANS NAFIS**

**NIM : 2511532027**

**FAKULTAS TEKNOLOGI INFORMASI**

**DEPARTEMEN INFORMATIKA**

**UNIVERSITAS ANDALAS**

**2025**

## **KATA PENGANTAR**

Puji syukur kepada Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Laporan Praktikum Pekan 5 ini tepat pada waktunya. Laporan ini disusun untuk memenuhi salah satu tugas praktikum mata kuliah Algoritma dan Pemrograman, dengan topik pembahasan mengenai *for looping* pada Bahasa Pemrograman Java.

Tujuan dari praktikum ini adalah untuk memahami dan mengimplementasikan konsep perulangan (*looping*), khususnya struktur *for loop* dalam bahasa pemrograman. Melalui praktikum ini, mahasiswa diharapkan mampu menggunakan perulangan *for* untuk menyelesaikan berbagai permasalahan yang melibatkan proses berulang secara efektif dan efisien.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun demi perbaikan di masa mendatang. Akhir kata, penulis mengucapkan terima kasih kepada Bapak Wahyudi S.T. M.T. selaku dosen pengampu mata kuliah Algoritma dan Pemrograman, asisten laboratorium, serta teman-teman praktikum dan pihak lain yang turut mendukung penulisan laporan ini.

Kamis, 30 Oktober 2025

Muhammad Hans Nafis

## DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI .....	ii
BAB 1 .....	1
PENDAHULUAN .....	1
1.1 Latar Belakang.....	1
1.2 Tujuan Praktikum .....	1
1.3 Manfaat Praktikum .....	1
BAB 2 .....	3
PEMBAHASAN.....	3
2.1 <i>For</i> .....	3
2.2 <i>Nested For</i> .....	6
BAB 3 .....	9
KESIMPULAN .....	9
DAFTAR PUSTAKA .....	10
LAMPIRAN .....	11

# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Dalam dunia pemrograman, proses pengulangan atau *looping* merupakan salah satu konsep dasar yang sangat penting. *Looping* memungkinkan sebuah program untuk mengeksekusi perintah tertentu secara berulang tanpa harus menulis kode yang sama berkali-kali. Dalam bahasa pemrograman Java, terdapat beberapa jenis perulangan seperti *for*, *while*, dan *do-while*.

Struktur perulangan *for* sering digunakan ketika jumlah pengulangan sudah diketahui sebelumnya. Dengan menggunakan *for*, programmer dapat menulis program yang lebih efisien, ringkas, dan mudah dipahami. Pemahaman yang baik mengenai struktur perulangan ini sangat penting untuk menyelesaikan berbagai permasalahan dalam pemrograman, seperti perhitungan matematis, pengolahan data, maupun pembuatan algoritma yang kompleks. Oleh karena itu, praktikum ini dilakukan untuk memahami konsep, sintaks, dan penerapan perulangan *for* dalam bahasa Java secara tepat.

### 1.2 Tujuan Praktikum

Tujuan dari pelaksanaan praktikum antara lain sebagai berikut :

1. Memahami konsep dasar perulangan *for* dan mengetahui cara kerja struktur *for* dalam bahasa Java.
2. Mengimplementasikan perulangan *for* untuk menyelesaikan permasalahan sederhana dalam program Java.
3. Melatih kemampuan logika dan berpikir sistematis dalam menyusun algoritma menggunakan perulangan.

### 1.3 Manfaat Praktikum

Manfaat dari pelaksanaan praktikum antara lain sebagai berikut :

1. Mahasiswa dapat menulis program Java yang menggunakan struktur perulangan *for* secara efektif.
2. Mahasiswa mampu Memahami penerapan perulangan dalam berbagai kasus, seperti menampilkan deret angka, menghitung total nilai, atau membuat pola tertentu.

3. Mahasiswa dapat mengembangkan keterampilan berpikir logis dan analitis dalam menyelesaikan masalah pemrograman.

## BAB 2

### PEMBAHASAN

#### 2.1 *For*

Dalam bahasa pemrograman Java, *for loop* adalah salah satu bentuk struktur perulangan (*looping*) yang digunakan untuk mengeksekusi satu atau beberapa pernyataan (*statements*) secara berulang, selama kondisi tertentu terpenuhi. Struktur ini sangat berguna ketika jumlah pengulangan sudah diketahui sebelumnya (*counted loop*).

Sintaks *for* :

```
for (inisialisasi; test; update) {  
    statement;  
}
```

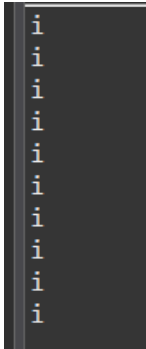
Bagian inisialisasi digunakan untuk menginisialisasi variabel penghitung yang akan digunakan dalam perulangan. Inisialisasi hanya dilakukan sekali. Bagian *test* merupakan ekspresi logika yang akan diperiksa sebelum setiap iterasi. Jika bernilai *true*, blok pernyataan di dalam *loop* akan dijalankan. Jika bernilai *false*, perulangan akan berhenti. Bagian *update* digunakan untuk mengubah nilai variabel penghitung setelah setiap satu kali iterasi selesai, biasanya dengan menambah nilai/*increment* (*variabel++*) atau mengurangi nilai/*decrement* (*variabel--*).

Beberapa contoh kode program *for* :

```
for (int i = 1 ; i <= 10; i++) {  
    System.out.println("i");  
}  
  
}
```

Variabel *i* diinisialisasi dengan 1. Selama nilai *i* kecil sama dengan 10, blok pernyataan di dalam kurung kurawal akan dijalankan secara berulang. Setelah setiap iterasi selesai, nilai *i* akan bertambah 1 (*i++*). Pada blok pernyataan, *i* diberi tanda kutip untuk menampilkan teksn”i” ke layar sebanyak 10 kali, satu di setiap baris

Output :



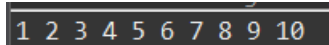
```
i
i
i
i
i
i
i
i
i
i
```

Gambar 2.1

```
public static void main(String[] args) {
    for (int i = 1; i <= 10; i++) {
        System.out.print(i+" ");
    }
}
```

Sama seperti sebelumnya, namun disini yang ditampilkan adalah nilai i yang diikuti dengan spasi pada satu baris horizontal.

Output :



```
1 2 3 4 5 6 7 8 9 10
```

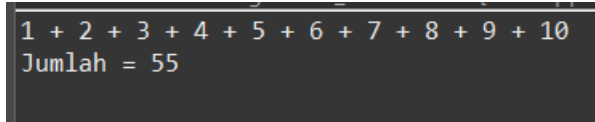
Gambar 2.2

```
int jumlah=0;
for (int i = 1; i<=10; i++) {
    System.out.print(i);
    jumlah= jumlah+i;
    if (i<10) {
        System.out.print(" + ");
    }
}
System.out.println();
System.out.println("Jumlah = "+jumlah);
}
```

Variabel jumlah diinisialisasi dengan 0 untuk menyimpan nilai penjumlahan sementara. Perulangan *for* digunakan dengan variabel penghitung i yang dimulai dari satu, dan akan terus bertambah satu per iterasi hingga bernilai 10. Program menampilkan nilai i dan menjumlahkan nilai i ke dalam variabel jumlah. If (i<10) berfungsi untuk menampilkan tanda + di antara angka-angka, kecuali setelah angka

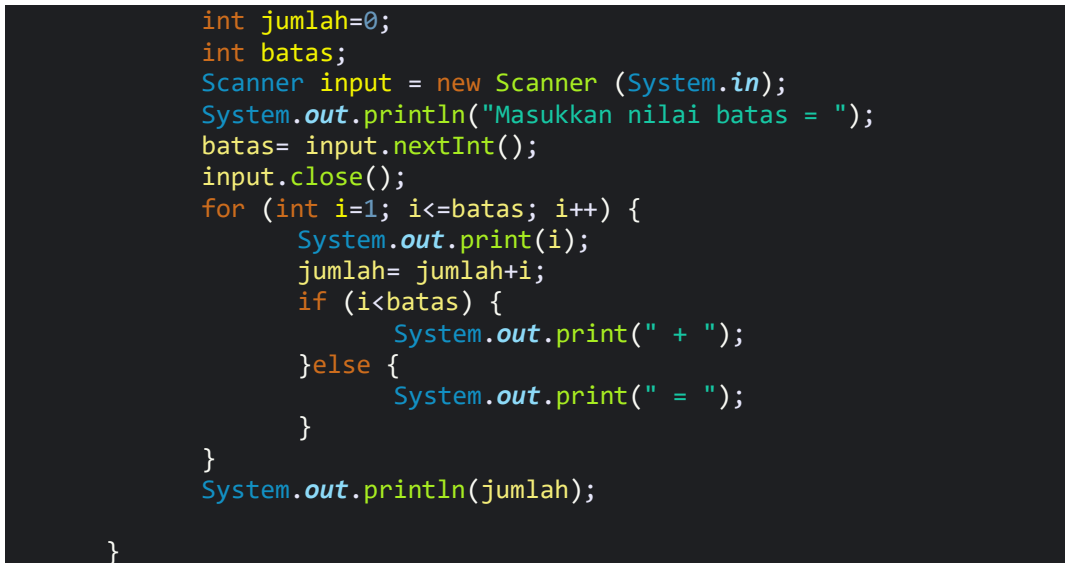
terakhir. Setelah seluruh perulangan selesai dijalankan, program menampilkan baris baru dan mencetak total penjumlahan.

Output :



```
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
Jumlah = 55
```

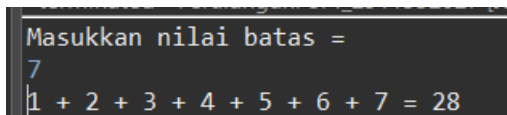
Gambar 2.3



```
int jumlah=0;
int batas;
Scanner input = new Scanner (System.in);
System.out.println("Masukkan nilai batas = ");
batas= input.nextInt();
input.close();
for (int i=1; i<=batas; i++) {
    System.out.print(i);
    jumlah= jumlah+i;
    if (i<batas) {
        System.out.print(" + ");
    }else {
        System.out.print(" = ");
    }
}
System.out.println(jumlah);
}
```

Program mendeklarasikan variabel jumlah dengan nilai awal 0 untuk menyimpan hasil penjumlahan, serta variabel batas yang akan diisi oleh input pengguna. Kemudian, program meminta pengguna memasukkan nilai batas melalui Scanner. Struktur perulangan digunakan untuk mengulangi proses penjumlahan, mulai dari i=1 hingga i sama dengan batas. Pada setiap iterasi, nilai i dicetak ke layar dan ditambahkan ke dalam variabel jumlah. If (i<batas) berfungsi untuk menampilkan tanda + di antara bilangan, dan jika sudah mencapai bilangan terakhir, program menampilkan tanda =. Setelah perulangan selesai, program mencetak nilai akhir dari variabel jumlah.

Output :



```
Masukkan nilai batas =
7
1 + 2 + 3 + 4 + 5 + 6 + 7 = 28
```

Gambar 2.4



## 2.2 Nested For

*Nested for* adalah perulangan (*loop*) *for* di dalam perulangan *for* lainnya. Dengan kata lain, satu perulangan berada di dalam perulangan lain, sehingga ketika perulangan luar (*outer loop*) berjalan satu kali, perulangan dalam (*inner loop*) dapat berjalan beberapa kali.

Sintaks *nested for* :

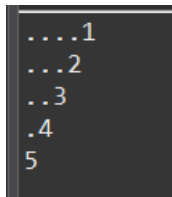
```
for (inisialisasi1; test1; update1) {  
    for (inisialisasi2; test2; update2) {  
        statement2;  
    }  
    statement1;  
}
```

Beberapa contoh kode program *nested for* :

```
for (int line = 1; line <= 5; line++) {  
    for (int j = 1; j <= (-1 * line + 5); j++) {  
        System.out.print(".");  
    }  
    System.out.print(line);  
    System.out.println();  
}
```

Perulangan pertama menggunakan variabel *line* yang berjalan dari 1 hingga 5, dan setiap nilai *line* mewakili baris beberapa yang sedang dicetak. Di dalamnya terdapat perulangan kedua dengan variabel *j*, yang bertugas mencetak karakter titik (.) sebanyak  $(-1 * \text{line} + 5)$  kali. Artinya, pada baris pertama ( $\text{line} = 1$ ), akan dicetak 4 titik, pada baris kedua 3 titik, dan seterusnya hingga pada baris kelima tidak ada titik sama sekali. Setelah perulangan dalam selesai, program mencetak nilai *line* menggunakan `System.out.print(line);` dan membuat baris baru, agar hasil berikutnya dicetak di baris selanjutnya.

Output :



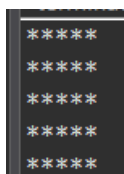
```
....1
...2
..3
.4
5
```

Gambar 2.5

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 5; j++) {
        System.out.print("*");
    }
    System.out.println();
    // to end the line
}
```

Perulangan pertama berfungsi untuk mengatur jumlah baris yang akan dicetak, dalam hal ini sebanyak 5 baris. Di dalam perulangan tersebut terdapat perulangan kedua, yang bertugas mencetak lima buah karakter bintang (\*) pada setiap baris. Perintah `System.out.print("*");` digunakan untuk mencetak bintang tanpa pindah ke baris baru, sehingga semua bintang pada satu baris akan muncul berdampingan. Setelah perulangan `j` selesai, program mencetak bintang lainnya pada baris berikutnya.

Output :



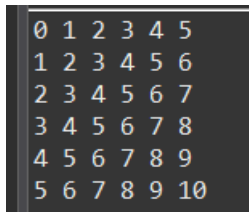
```
*****
*****
*****
*****
*****
```

Gambar 2.6

```
for (int i = 0; i <= 5; i++) {
    for (int j = 0; j <= 5; j++) {
        System.out.print(i+j+ " ");
    }
    System.out.println();
    // to end the line
}
```

Perulangan pertama menggunakan variabel *i* sebagai penghitung luar, sedangkan perulangan kedua menggunakan variabel *j* sebagai penghitung dalam. Nilai *i* dimulai dari 0 hingga 5 ( $i \leq 5$ ), dan untuk setiap nilai *i*, perulangan *j* juga berjalan dari 0 hingga 5 ( $j \leq 5$ ). Pada setiap iterasi bagian dalam, program akan mengeksekusi perintah `System.out.print(i + j + " ");`, yang berarti mencetak hasil penjumlahan antara nilai *i* dan *j*, diikuti oleh satu spasi, tanpa pindah baris. Setelah perulangan *j* selesai, output dicetak di baris baru. Akibatnya, output program ini akan membentuk sebuah tabel angka di mana setiap baris menunjukkan hasil penjumlahan antara nilai *i* (baris) dan *j* (kolom).

Output :



```
0 1 2 3 4 5
1 2 3 4 5 6
2 3 4 5 6 7
3 4 5 6 7 8
4 5 6 7 8 9
5 6 7 8 9 10
```

Gambar 2.7

### **BAB 3**

#### **KESIMPULAN**

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa perulangan *for* pada bahasa Java merupakan struktur kontrol yang digunakan untuk menjalankan perintah secara berulang dengan jumlah iterasi yang telah ditentukan. Dengan menggunakan perulangan *for*, proses yang bersifat repetitif dapat dilakukan secara lebih efisien tanpa perlu menulis kode berulang kali. Melalui praktikum ini, penulis memahami cara kerja inisialisasi, *test*, dan *update* pada *for*, serta mampu mengimplementasikannya dalam berbagai kasus, seperti menampilkan deret bilangan, menghitung hasil operasi berulang, dan membentuk pola tertentu menggunakan logika pemrograman.

Sebagai saran, disarankan mahasiswa lebih aktif berlatih menulis dan menjalankan program secara mandiri di luar jam praktikum. Hal ini penting karena keterampilan pemrograman hanya dapat dikuasai dengan sering berlatih. Dengan demikian, diharapkan pada praktikum berikutnya mahasiswa dapat lebih cepat memahami materi, meminimalisir kesalahan sintaks, serta mampu mengembangkan program yang lebih kompleks dan bermanfaat.

## DAFTAR PUSTAKA

- [1] <https://www.petanikode.com/java-perulangan/> [Diakses 30 Oktober 2025]
- [2] <https://www.duniailkom.com/tutorial-belajar-java-perulangan-for-bahasa-java/>  
[Diakses 30 Oktober 2025]
- [3] <https://www.programiz.com/java-programming/nested-loop> [Diakses 30  
Oktober 2025]
- [4] [https://www.w3schools.com/java/java\\_for\\_loop\\_nested.asp](https://www.w3schools.com/java/java_for_loop_nested.asp) [Diakses 30  
Oktober 2025]

## **LAMPIRAN**

Gambar 2.1 .....	4
Gambar 2.2 .....	4
Gambar 2.3 .....	5
Gambar 2.4 .....	5
Gambar 2.5 .....	7
Gambar 2.6 .....	7
Gambar 2.7 .....	8