

LAPORAN PRAKTIKUM
ALGORITMA PEMOGRAMAN
PERULANGAN FOR PADA PEMOGRAMAN JAVA

Disusun Oleh :

Aliifah Felda Mufarrihati Salwaa

2511531011

Dosen Pengampu :

Dr. Wahyudi, S.T., M.T.

Asisten Praktikum :

Aufan Taufiqurrahman



DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS
TAHUN 2025

KATA PENGANTAR

Laporan praktikum ini disusun sebagai salah satu bentuk pertanggungjawaban kegiatan praktikum algoritma pemograman yang membahas tentang perulangan for pada pemograman Java. Melalui laporan ini penulis dapat memahami materi praktikum secara mendalam dan dapat lebih teliti, teratur, serta memiliki kemampuan yang baik dalam penulisan kode pemograman sesuai kaidah akademik. Sehingga laporan ini dapat menjadi sarana belajar, dokumentasi kegiatan, dan referensi praktikum berikutnya.

Penulis menyadari bahwa laporan ini masih memiliki banyak kekurangan, baik dari isi maupun penyajiannya. Oleh karena itu, saran dan kritik sangat penulis harapkan guna untuk menyempurnakan laporan berikutnya.

Padang, 29 Oktober 2025

Aliifah Felda Mufarrihati Salwaa

2511531011

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI.....	ii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang Praktikum	1
1.2 Tujuan Praktikum.....	1
1.3 Manfaat Praktikum.....	1
BAB II PEMBAHASAN	2
2.1 Perulangan <i>for</i> pada Pemograman Java	2
2.2 Praktikum “PerulanganFor1”	3
2.3 Praktikum “PerulanganFor2”	4
2.4 Praktikum “PerulanganFor3”	4
2.5 Praktikum “PerulanganFor4”	6
2.6 Praktikum “nestedFor0”	8
2.7 Praktikum “nestedFor1”	10
2.8 Praktikum “nestedFor2”	10
BAB III KESIMPULAN	13
3.1 Kesimpulan.....	13
3.2 Saran.....	13
DAFTAR PUSTAKA	14

BAB I

PENDAHULUAN

1.1 Latar Belakang Praktikum

Dalam pemograman, efektivitas dan efisiensi kode sangat dipengaruhi oleh sistem kontrol. Salah satu sistem kontrol dasar yang dapat kita gunakan adalah sistem perulangan. Terdapat tiga sistem perulangan pada pemograman java: *for*, *while*, *do-while*.

Pada praktikum kali ini kita akan lebih membahas tentang perulangan *for* (*for loop*). Perulangan *for* digunakan ketika jumlah iterasi kita ketahui secara pasti. Perulangan *for* umum digunakan pada kasus seperti iterasi koleksi, menghasilkan deret angka, proses teks atau manipulasi string, dan mengulang isi berkas.

Oleh karena itu, praktikum ini dilaksanakan agar mahasiswa dapat memahami dan menerapkan konsep perulangan *for* pada bahasa pemograman Java, sehingga kelak dapat menuliskan kode program dengan efektif dan efisien.

1.2 Tujuan Praktikum

Tujuan dari pelaksanaan praktikum ini adalah sebagai berikut:

1. Mengetahui bagaimana cara penggunaan perulangan *for* pada bahasa pemograman Java.
2. Mengetahui langkah-langkah dalam membuat program Java dengan menggunakan perulangan *for*.
3. Mengetahui hasil dari program Java yang menggunakan perulangan *for*.

1.3 Manfaat Praktikum

Manfaat dari pelaksanaan praktikum ini adalah sebagai berikut:

1. Memahami penulisan kode pemograman Java terutama dalam menggunakan perulangan *for*.
2. Mampu mengimplementasikan perulangan *for* pada pembuatan kode program Java kedepannya.
3. Meningkatkan kemampuan dalam penulisan kode program dalam bahasa Java terutama pada perulangan *for*.

BAB II PEMBAHASAN

2.1. Perulangan *for* pada pemograman Java

Pada pemograman Java, ada enam struktur yang digunakan untuk menentukan alur kontrol normal dalam suatu program. Keenam struktur kontrol tersebut adalah: *block*, *while loop*, *do.while loop*, *for loop*, *if statement*, dan *switch statement*. Masing-masing struktur ini dianggap sebagai satu pernyataan yang dapat berisi satu atau lebih pernyataan lain di dalamnya.

Struktur kontrol yang digunakan pada praktikum kali ini adalah pengulangan *for* (*for loop*). Pengulangan *for* memberikan cara ringkas untuk mengulangi suatu nilai, yang berarti *for loop* bekerja dengan terus melakukan pengulangan hingga kondisi atau rentang tertentu terpenuhi. Bentuk paling sederhana dalam menulis perulangan *for* adalah sebagai berikut:

```
for (initialization; condition; iteration) {statement(s);
```

Dalam bentuk ini, bagian insialisasi (*initialization*) merupakan pernyataan penugasan yang menetapkan nilai awal variabel. Kondisi (*condition*) adalah operator *boolean* yang menentukan apakah perulangan akan dilanjutkan atau tidak. Iterasi (*iteration*) menentukan bagaimana variabel diubah selama perulangan. Bahasa pemograman Java juga menyediakan cara yang lebih efektif dalam menuliskan iterasi, yakni dengan operator *increment* dan *decrement*. Operator *increment* menaikkan nilai operannya sebanyak satu (*variable++*), sementara itu operator *decrement* menurunkan nilai operannya sebanyak satu (*variable--*).

2.2. Praktikum “PerulanganFor1”

```
package pekan5;

public class PerulanganFor1 {

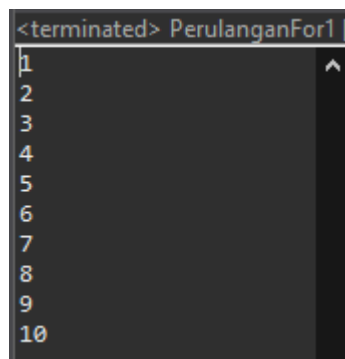
    public static void main(String[] args) {
        for (int i=1; i<=10; i++) {
            System.out.println(i);
        }
    }
}
```

Kode Program 2.1: Praktikum “PerulanganFor1”

Pada kode program ini kita menggunakan bentuk perulangan *for* yang paling sederhana, yakni dengan struktur (inisialisasi; kondisi; iterasi;). Berikut penjelasan singkat tentang penulisan kode program diatas:

1. Pada baris kode 6 kita memasukkan kode program perulangan *for* sederhana dengan format `for (inisialisasi; kondisi; iterasi) {statement;}` dengan:
 - a. Inisialisasi `i = 0;`
 - b. Kondisi `i<=10;`
 - c. Iterasi dengan operator *increment*;
2. Pada baris kode 7 kita memasukkan statement yang akan digunakan selama perulangan *for* terjadi. Disini kita menggunakan *output* `println` untuk mencetak semua hasil `i` dari perulangan *for* yang dilakukan.

Dari langkah-langkah yang telah kita gunakan untuk menuliskan program, maka kita akan mendapatkan *output* seperti gambar 2.1 berikut:



```
<terminated> PerulanganFor1 [ ... ]
1
2
3
4
5
6
7
8
9
10
```

Gambar 2.1: *output* kode porgram “PerulanganFor1”

Output ini kita dapatkan dari menginisialisasi *i* dengan 1, sehingga angka pertama yang tampil adalah 1. Kemudian kita membuat kondisi $i \leq 10$, dimana program akan terus melakukan perulangan hingga kondisi ini bernilai *false* atau $i > 10$. Iterasi yang kita gunakan adalah operasi *increment* yang dimana ini sama dengan $i+1$ setiap kali dilakukan perulangan. Karena faktor-faktor diatas, kita mendapatkan *output* berupa angka 1 hingga 10.

2.3. Praktikum “PerulanganFor2”

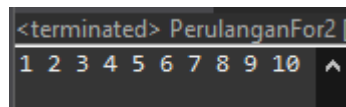
```
package pekan5;

public class PerulanganFor2 {

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

Kode Program 2.2: praktikum “PerulanganFor2”

Kode program praktikum perulangan *for* kedua ini tidak banyak berbeda dibandingkan dengan kode program yang pertama. Seluruh langkah dan penjelasan kode program ini sama dengan kode program 1, tetapi yang membedakan adalah pada sistem *output* dari program, yakni: `System.out.print(i+" ");`. Kode ini akan tetap menghasilkan angka 1 hingga 10 tetapi dalam barisan horizontal seperti berikut:



Gambar 2.2: output kode program “PerulanganFor2”

2.4. Praktikum “PerulanganFor3”

```
package pekan5;

public class PerulanganFor3 {
```

```

public static void main(String[] args) {
    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);
}
}

```

Kode Program 2.3: Praktikum “PerulanganFor3”

Pada kode program ini, kita menggunakan perulangan *for* dengan banyak statement, yakni: sistem *output*, operasi aritmatika, dan statement *if*. Berikut rincian dari kode program diatas:

1. Pada baris kode 6 kita mendeklarasikan sebuah variabel jumlah=0
2. Pada baris kode 7 kita memasukkan perulangan *for* dengan:
 - a. Inisialisasi i=1;
 - b. Kondisi i<=10; dimana program akan terus berjalan atau berulang hingga nilai i<=10 menjadi *false*.
 - c. Iterasi *increment* (i++); dimana nilai i akan terus bertambah selama program pengulangan berjalan.
3. Pada baris kode 8-11, merupakan statement dari perulangan *for* yang kita miliki, yakni berupa:
 - a. Mengeluarkan *output* berupa nilai i
 - b. Mengubah nilai awal jumlah dengan operasi aritmatika jumlah=jumlah+i
 - c. Memasukkan statement *if* dimana jika i<10, maka program akan menghasilkan *output* “+”
4. Pada baris kode 14-15, program akan menghasilkan *output* berupa hasil dari perulangan *for* dan hasil dari variabel jumlah.


```
<terminated> PerulanganFor3 [Java Application] D:\Ecl
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 ^
Jumlah = 55
```

Gambar 2.3: *output* kode program “PerulanganFor3”

Gambar 2.3 merupakan *output* yang dihasilkan dari kode program perulangan *for* ketiga. Dengan menggunakan perulangan *for*, kita bisa membuat suatu baris bilangan dengan operator aritmatika dan mendapatkan hasil operasi dari seluruh bilangan yang ada.

2.5. Praktikum “PerulanganFor4”

```
package pekan5;

import java.util.Scanner;

public class PerulanganFor4 {

    public static void main(String[] args) {
        int jumlah=0;
        int batas;
        Scanner input=new Scanner(System.in);
        System.out.print("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);
    }
}
```

Kode Program 2.4: Praktikum “PerulanganFor4”

Pada kode program kali ini, kita menggunakan *input* dari user untuk menentukan kondisi dari perulangan *for* yang akan kita lakukan. Berikut penjelasan lengkap tentang cara kerja kode program diatas:

1. Pada baris kode 3 kita inisialisasi `java.util.Scanner` untuk melakukan *input*.

2. Pada baris kode 8-9 kita inialisasi variabel `jumlah=0` dan variabel `batas`.
3. Pada baris kode 10-13 kita melakukan proses *input* dari user. Input yang diterima berupa integer yang kemudian disimpan dalam variabel `batas`.
4. Pada baris kode 14 kita menggunakan perulangan *for*, dimana:
 - a. Inialisasi `i=1`;
 - b. Kondisi `i<=batas`; dimana nilai dari variabel `batas` akan berubah tergantung dengan *input* dari user dan perulangan akan selalu berjalan hingga kondisi ini bernilai *false*.
 - c. Iterasi *increment* (`i++`); dimana nilai `i` akan selalu bertambah selama perulangan berlangsung.
5. Baris kode 15-22 merupakan bagian dari statement perulangan *for*, dimana:
 - a. Program akan mencetak *output* berupa nilai `i`.
 - b. Nilai dari variabel `jumlah` akan terus berubah karena menggunakan operator aritmatika penjumlahan.
 - c. Penggunaan statement *if* dimana jika `i<batas`, maka program akan mencetak *output* berupa tanda “+” dan jika lain dari itu maka program akan mencetak *output* berupa tanda “=”
6. Pada baris kode 23 program akan mencetak hasil akhir dari variabel `jumlah`.

```
<terminated> PerulanganFor4 [Java App  
Masukkan nilai batas = 5  
1 + 2 + 3 + 4 + 5 = 15
```

Gambar 2.4: salah satu *output* kode program “PerulanganFor4”

```
<terminated> PerulanganFor4 [Java Application] D:\Eclipse\ec  
Masukkan nilai batas = 10  
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55
```

Gambar 2.5: salah satu *output* kode program “PerulanganFor4”

Dari kedua *output* yang dihasilkan, kita dapat menyimpulkan bahwasanya kode program ini mirip dengan kode program praktikum perulangan *for* ketiga. Namun bedanya, kondisi atau batas pada perulangan disini ditentukan oleh user sehingga *output* yang dihasilkan menjadi beragam.

2.6. Praktikum “nestedFor0”

```
package pekan5;

public class nestedFor0 {

    public static void main(String[] args) {
        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();
        }
    }
}
```

Kode Program 2.5: kode program praktikum “nestedFor0”

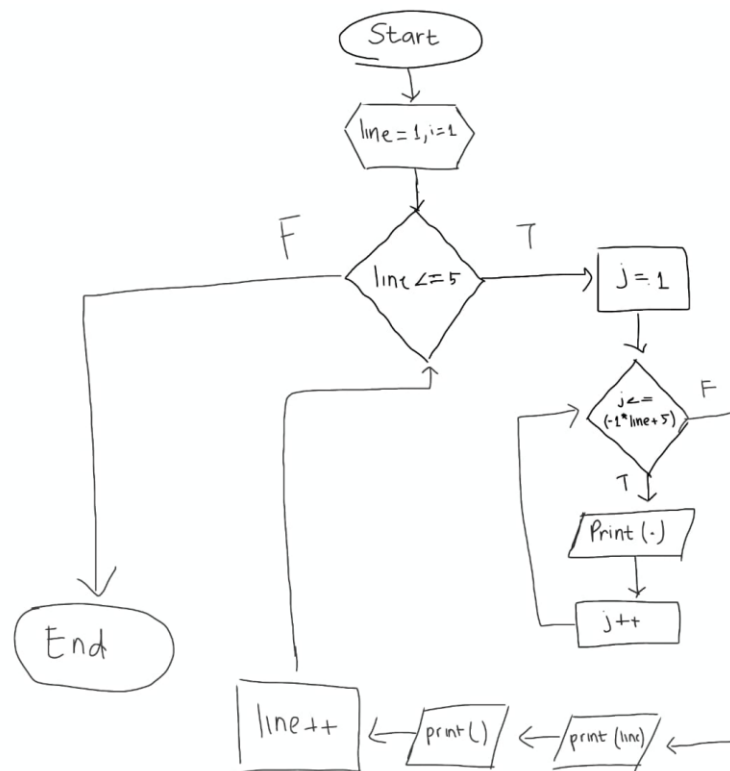
Pada kode pemograman ini, kita menggunakan perulangan *for* di dalam perulangan *for* atau dapat kita sebut dengan *nested for*. Perulangan *for* yang kedua akan dijalankan satu kali untuk setiap iterasi dari perulangan *for* pertama. Berikut penjelasan lengkap tentang kode diatas:

1. Pada baris kode 6 kita menuliskan perulangan *for* yang pertama atau disebut juga sebagai *outer for loop* dengan:
 - a. Inisialisasi variabel `line=1`;
 - b. Kondisi `line<=5`; dimana perulangan akan terus berjalan hingga kondisi ini bernilai *false*.
 - c. Iterasi *increment* variabel `line`; dimana nilai dari variabel `line` akan terus bertambah selama perulangan berlangsung.
2. Pada baris kode 7 kita menuliskan perulangan *for* yang kedua atau disebut juga sebagai *inner for loop* sebagai statement dari *outer for loop* dengan:
 - a. Inisialisai variabel `j=1`;
 - b. Kondisi `j<=(-1*line+5)`; dimana perulangan akan terus berjalan hingga kondisi ini bernilai *false*.
 - c. Iterasi *increment* variabel `j`; dimana nilai `j` akan terus bertambah selama perulangan berlangsung.
3. Baris kode 8 merupakan statement dari *inner for loop* yang merupakan perintah untuk mengeluarkan *output* berupa titik “.”.

```
<terminated> nestedFor0 [Java
....1
...2
..3
.4
5
```

Gambar 2.6: *output* kode program “nestedFor0”

Gambar 2.6 merupakan *output* yang akan dihasilkan oleh kode program *nested for* yang kita gunakan kali ini. Program *nested for* akan mengeksekusi *outer for loop* sehingga mendapatkan nilai dari variabel yang ada pada *outer for loop*. Kemudian, program akan lanjut mengeksekusi *inner for loop* hingga kondisi pada *inner for loop* bernilai *false*. Dari sini, program akan kembali mengeksekusi *outer for loop* dan kembali masuk pada *inner for loop*. Hal ini akan terus berulang hingga kondisi pada *outer for loop* bernilai *false*. Berikut merupakan flowchart dari penggunaan *nested for* agar penjelasan diatas lebih mudah dipahami:



Gambar 2.7: flowchart program “nestedFor0”

2.7. Praktikum “nestedFor1”

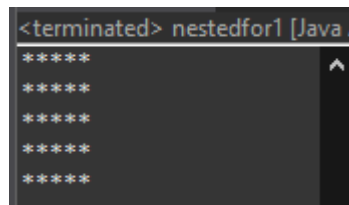
```
package pekan5;

public class nestedfor1 {

    public static void main(String[] args) {
        for (int i = 1; i<=5; i++) {
            for (int j=1; j<=5; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Kode Program 2.6: Praktikum “nestedFor1”

Pada program ini kita masih menggunakan sistem *nested for* dimana terdapat *outer for loop* dan *inner for loop* pada sebuah program. Program ini tidak jauh berbeda jika dibandingkan dengan program “nestedFor0”. Yang menjadi pembeda pada kali ini adalah batas/kondisi pada *inner for loop* dan *output* yang akan dikeluarkan. Langkah-langkah dan penjelasan tentang kode ini sama dengan kode program “nestedFor0”



Gambar 2.8: *output* kode program “nestedFor1”

Output ini kita dapatkan dari kode program “nestedFor1”. *Outer for loop* akan menghasilkan lima baris bintang dan *inner for loop* menghasilkan lima kolom bintang. Hal ini sesuai dengan cara kerja *nested for* dan kondisi yang dimiliki oleh *outer for loop* dan *inner for loop*.

2.8. Praktikum “nestedFor2”

```
package pekan5;

public class nestedFor2 {
```

```

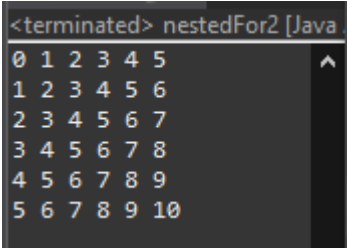
public static void main(String[] args) {
    for (int i = 0; i<=5; i++) {
        for (int j=0; j<=5; j++) {
            System.out.print(i+j+" ");
        }
        System.out.println();
    }
}

```

Kode Program 2.7: praktikum “nestedFor2”

Pada kode program ini kita masih menggunakan *nested for* untuk menghasilkan sebuah *output*. Namun pada program ini kita melakukan operasi aritmatika yang melibatkan variabel pada *outer for loop* dan *inner for loop* di dalam *inner for loop*. Untuk lebih jelasnya, berikut cara kerja dari program diatas:

1. Baris kode 6 adalah *outer for loop* dengan:
 - a. Inisialisasi $i=0$;
 - b. Kondisi $i \leq 5$; dimana program akan terus berjalan hingga kondisi ini bernilai *false*.
 - c. Iterasi *inrement*; dimana nilai variabel i akan terus bertambah selama perulangan berlangsung.
2. Baris kode 7 adalah *inner for loop* dengan:
 - a. Inisialisasi $j=0$;
 - b. Kondisi $j \leq 5$;
 - c. Iterasi *increment*; dimana nilai dari variabel j akan terus bertambah selama perulangan berlangsung.
3. Baris kode 8 merupakan statement dari *inner for loop* dimana statement ini mengandung operasi aritmatika penjumlahan antara variabel i dan j .



```

<terminated> nestedFor2 [Java]
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 2.9: *output* kode program “nestedFor2”

Pada baris pertama *output*, kita memiliki angka 0 1 2 3 4 5. Angka-angka ini berasal dari program perulangan *for* bagian *inner for loop*. Dimana setelah program mengeksekusi nilai pertama dari variabel *i*, program akan masuk pada *inner for loop* dan akan mengeksekusi nilai pertama variabel *j* serta melakukan operasi aritmatika dengan menambahkan variabel *i* dengan variabel *j*. Proses pertambahan ini akan terus berlangsung hingga kondisi dari *inner for loop* bernilai *false* yakni ketika nilai $j > 5$. Hasil dari proses perhitungan pada *inner for loop* ini akan menghasilkan suatu barisan angka dari 1 hingga 5.

Program selanjutnya akan kembali mengeksekusi bagian *outer for loop* dengan menambahkan nilai dari variabel *i*. Kemudian program akan kembali masuk pada bagian *inner for loop* untuk kembali melakukan pertambahan dan akan terus berulang hingga kondisi pada *inner for loop* bernilai *false*. Proses ini akan terus berulang hingga kondisi pada *outer for loop* juga bernilai *false* yakni ketika nilai $i > 5$.

BAB III

KESIMPULAN

3.1. Kesimpulan

Dari seluruh praktek dan landasan teori yang kita miliki, kita dapat memahami bahwasanya sistem perulangan sangat dibutuhkan untuk dapat membuat sebuah kode program menjadi lebih efektif dan efisien. Jika kita ingin melakukan suatu sistem perulangan yang banyak dan panjang tanpa menggunakan *tools* perulangan seperti *for*, kita akan kewalahan dalam menuliskan kode yang sangat panjang. Oleh karena itu, pemahaman dalam sistem perulangan terutama sistem perulangan *for* yang dipelajari saat ini sangat dibutuhkan.

Perulangan *for* sendiri dapat digunakan didalam perulangan *for* yang lain atau disebut juga dengan *nested for*. Dalam penggunaan *nested for*, perulangan akan mengeksekusi nilai awal dari variabel pada *for* pertama (*outer for loop*). Kemudian, perulangan berlangsung pada *for* kedua (*inner for loop*) hingga kondisi pada *inner for loop* bernilai *false*. Setelah itu, program akan kembali mengeksekusi *outer for loop* dan masuk kembali pada *inner for loop*. Kondisi ini akan terus berulang hingga kondisi pada *outer for loop* bernilai *false*.

3.2. Saran

Praktek dalam menggunakan perulangan *for* sedikit rumit apabila tidak terbiasa. Untuk dapat menghasilkan *output* yang kita inginkan, terkadang kita akan bingung bagaimana cara untuk membuat kondisi yang tepat sehingga perulangan tersebut dapat menghasilkan *output* yang tepat juga. Untuk mengatasi hal itu, kita harus bisa memahami dengan baik bagaimana sistem perulangan *for* bekerja. Memahami bagaimana sebuah program itu bekerja dapat kita lakukan dengan terus-menerus melakukan praktek dan melakukan evaluasi ketika mengalami kesalahan atau kebingungan.

DAFTAR PUSTAKA

- [1] H. Schildt, *Java: A Beginner's Guide*, 6th ed. New York, NY: McGraw-Hill Education, 2014
- [2] “Loops in Java,” *GeeksforGeeks* [Daring]. Tersedia pada: <https://www.geeksforgeeks.org/java/loops-in-java/> [Diakses: 29-Okt-2025]
- [3] “For Loop in Programming,” *GeeksforGeeks* [Daring]. Tersedia pada: <https://www.geeksforgeeks.org/dsa/for-loop-in-programming/> [Diakses: 29-Okt-2025]
- [4] Oracle, “The for Statement,” *The Java™ Tutorials: Language Basics* [Daring]. Tersedia pada: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html> [Diakses: 29-Okt-2025]
- [5] “Java Nested Loops,” *W3Schools* [Daring]. Tersedia pada: https://www.w3schools.com/java/java_for_loop_nested.asp [Diakses: 29-Okt-2025]