

**LAPORAN PRATIKUM PEMROGRAMAN ALGORITMA DAN
PEMROGRAMAN**

PERULANGAN

disusun Oleh:

Habib Al Faruq

2511532010

Dosen Pengampu: Dr. Wahyudi S.T.M.T

Asisten Pratikum: Muhammad Zaki Al Hafiz



**DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS**

2025

KATA PENGANTAR

Puji syukur atas kehadiran Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan laporan praktikum algoritma pemrograman mengenai perulangan *for* di Java. Penulisan laporan ini bertujuan untuk membuat sebuah laporan tentang materi pemrograman mengenai penggunaan *for* agar program dapat dijalankan berulang kali sesuai dengan kondisi yang diberikan.

Penyusunan laporan praktikum ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, penulis ingin menyampaikan terima kasih kepada dosen pengampu mata kuliah praktikum pemrograman algoritma dan pemrograman, Bapak Dr. Wahyudi. S.T.M.T dan asisten praktikum, Uda Muhammad Zaki Al Hafiz yang telah memberikan arahan dan masukan dalam proses penyusunan laporan praktikum ini. Penulis berharap laporan praktikum ini dapat memberikan manfaat bagi pembaca dalam mempelajari konsep penggunaan *for*.

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	2
 BAB II PEMBAHASAN	 3
2.1 Perulangan.....	3
2.1.1 Perulangan For 1 dan 2	3
2.1.1.1 Penjelasan Kode Program	3
2.1.2 Perulangan For 3	4
2.1.2.1 Penjelasan Kode Program	4
2.1.3 Perulangan For 4	5
2.1.3.1 Penjelasan Kode Program	5
 2.2 Nested For	 6
2.2.1 Nested For 0.....	6
2.2.1.1 Penjelasan Kode Program	6
2.2.2 Nested For 1.....	7
2.2.2.1 Penjelasan Kode Program.....	7
2.2.3 Nested For 2.....	8
2.2.3.1 Penjelasan Kode Program.....	8
 BAB III KESIMPULAN	 10
3.1 Ringkasan Hasil Praktikum.....	10
DAFTAR PUSTAKA	12

BAB I

PENDAHULUAN

1.1 Latar Belakang Praktikum

Dalam bahasa pemrograman Java, struktur perulangan (*looping*) berfungsi untuk menjalankan serangkaian instruksi secara berulang tanpa perlu menulis kode yang sama berulang kali. Salah satu bentuk perulangan yang paling umum digunakan adalah *for*, karena sintaksnya sederhana dan mudah dikendalikan melalui tiga komponen utama: inisialisasi, kondisi, serta perubahan nilai variabel. Penggunaan perulangan *for* memungkinkan programmer mengotomatisasi berbagai proses seperti menampilkan deret angka, melakukan perhitungan tertentu, maupun mengakses elemen-elemen dalam array.

Selain perulangan tunggal, Java juga mendukung konsep *nested for* atau perulangan bersarang, yaitu perulangan yang ditempatkan di dalam perulangan lain. Struktur ini sering digunakan untuk menampilkan data berbentuk tabel, pola matriks, maupun pola visual seperti bentuk bintang (*pattern printing*). Pemahaman terhadap *for* dan *nested for* tidak hanya memperkuat kemampuan logika pemrograman, tetapi juga membantu mahasiswa memahami cara kerja struktur kontrol yang lebih kompleks dalam Java.

1.2 Tujuan Praktikum

Tujuan utama praktikum ini adalah membantu mahasiswa memahami konsep dasar perulangan *for* dalam bahasa Java serta mampu menerapkannya untuk menyelesaikan berbagai permasalahan yang memerlukan proses pengulangan. Melalui kegiatan ini, mahasiswa diharapkan dapat menjelaskan fungsi inisialisasi, kondisi, dan perubahan variabel dalam struktur *for*, sekaligus memahami keterkaitan antarbagian tersebut dalam mengendalikan alur eksekusi program.

Selain itu, praktikum ini juga bertujuan mengasah kemampuan mahasiswa dalam menerapkan *nested for* (perulangan bersarang) untuk membentuk pola tertentu dan menyelesaikan persoalan yang melibatkan data dua dimensi. Dengan berlatih menggunakan struktur perulangan bertingkat, mahasiswa diharapkan terbiasa berpikir secara logis dan sistematis dalam merancang program yang lebih efisien, rapi, dan terstruktur.

1.3 Manfaat Praktikum

Melalui kegiatan praktikum ini, mahasiswa diharapkan memperoleh pemahaman yang komprehensif mengenai mekanisme kerja perulangan *for* serta penerapannya dalam berbagai konteks pemrograman. Pemahaman tersebut membantu mahasiswa dalam menulis kode yang lebih efisien, ringkas, dan mudah dikelola, khususnya pada program yang memerlukan proses pengulangan dalam skala besar.

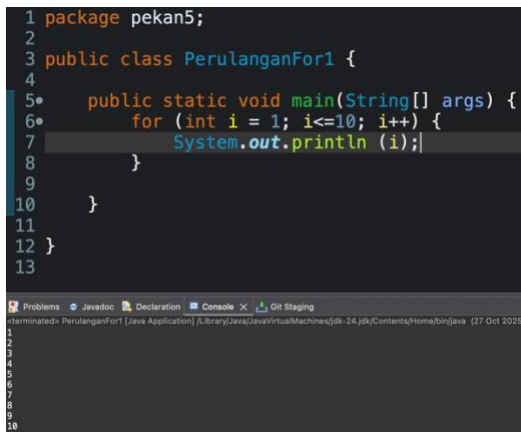
Selain itu, mahasiswa juga dilatih untuk mengembangkan kemampuan berpikir logis dan sistematis melalui penerapan *nested for* atau perulangan bersarang. Keterampilan ini tidak hanya berguna untuk menampilkan pola atau visualisasi sederhana di konsol, tetapi juga memiliki penerapan luas dalam pengolahan data dua dimensi, seperti manipulasi matriks, penerapan algoritma pencarian, serta pengelolaan struktur data berupa array 2D.

BAB II

PEMBAHASAN

2.1 Perulangan

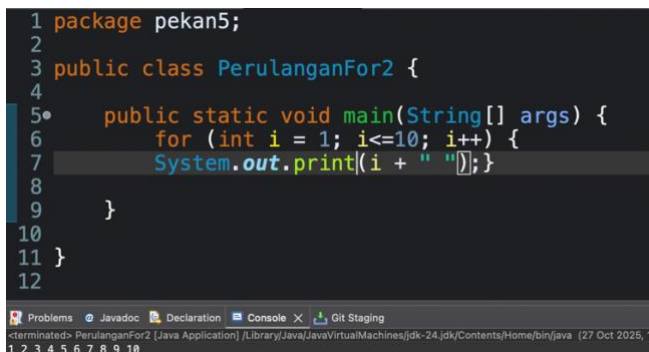
2.1.1 Perulangan For 1 dan 2



```
1 package pekan5;
2
3 public class PerulanganFor1 {
4
5     public static void main(String[] args) {
6         for (int i = 1; i<=10; i++) {
7             System.out.println(i);
8         }
9     }
10 }
11
12 }
13
```

Problems Javadoc Declaration Console X Git Staging
terminated> PerulanganFor1 [Java Application] /Library/Java/JavaVirtualMachines/jdk-24.jdk/Contents/Home/bin/java (27 Oct 2025, 10:10:10 AM)
1
2
3
4
5
6
7
8
9
10

Gambar 2.1



```
1 package pekan5;
2
3 public class PerulanganFor2 {
4
5     public static void main(String[] args) {
6         for (int i = 1; i<=10; i++) {
7             System.out.print(i + " ");
8         }
9     }
10 }
11 }
12
```

Problems Javadoc Declaration Console X Git Staging
terminated> PerulanganFor2 [Java Application] /Library/Java/JavaVirtualMachines/jdk-24.jdk/Contents/Home/bin/java (27 Oct 2025, 10:10:10 AM)
1 2 3 4 5 6 7 8 9 10

Gambar 2.2

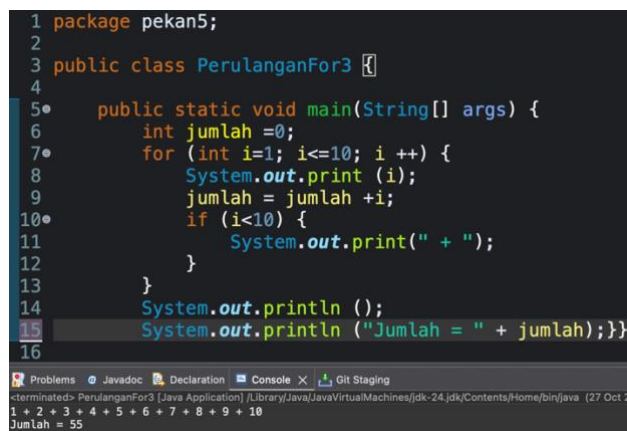
2.1.1.1 Penjelasan Kode Program

Pada gambar 2.1, kita membuat perulangan for yang mendeklarasikan *i* sebagai integer dengan nilai 1, *i* lebih kecil atau sama dengan 10, dan *i++* (setiap *i* ditambah 1). Program akan berjalan dengan mengambil nilai *i* dengan awal nilai *i* adalah 1 karena $1 \leq 10$, maka program akan mencetak nilai *i* pada *syntax print*. Setelah dicetak program akan otomatis menambah nilai *i* dengan 1 yang sekarang adalah 2. program

akan terus berjalan sampai nilai dari $i \leq 10$ sudah tidak valid atau i sudah lebih besar dari 10. Maka program akan mencetak semua nilai i yang ada. Output vertikal kebawah karena menggunakan *syntax* `System.out.println (i);`.

Pada gambar 2.2, kita membuat perulangan for yang mendeklarasikan i sebagai integer dengan nilai 1, i lebih kecil atau sama dengan 10, dan $i++$ (setiap i ditambah 1). Program akan berjalan dengan mengambil nilai i dengan awal nilai i adalah 1 karena $1 \leq 10$, maka program akan mencetak nilai i pada *syntax print*. Setelah dicetak program akan otomatis menambah nilai i dengan 1 yang sekarang adalah 2. program akan terus berjalan sampai nilai dari $i \leq 10$ sudah tidak valid atau i sudah lebih besar dari 10. Maka program akan mencetak semua nilai i yang ada. Output horizontal kesamping karena menggunakan *syntax* `System.out.print (i + " ");`.

2.1.2 Perulangan For 3



```
1 package pekan5;
2
3 public class PerulanganFor3 {
4
5     public static void main(String[] args) {
6         int jumlah = 0;
7         for (int i=1; i<=10; i++) {
8             System.out.print (i);
9             jumlah = jumlah +i;
10            if (i<10) {
11                System.out.print(" + ");
12            }
13        }
14        System.out.println ();
15        System.out.println ("Jumlah = " + jumlah);}}
16
```

Problems Javadoc Declaration Console Git Staging
terminated- PerulanganFor3 [Java Application] J:\Library\Java\JavaVirtualMachines\jdk-24.jdk\Contents\Home\bin\java (27 Oct 2024)
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
Jumlah = 55

Gambar 2.3

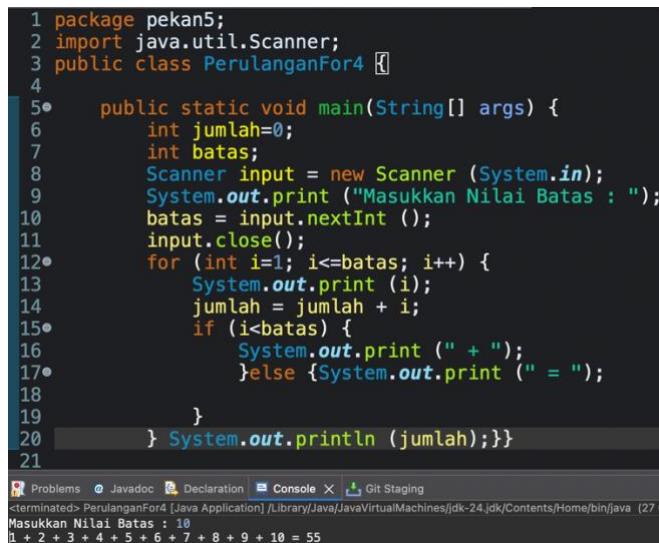
2.1.2.1 Penjelasan Kode Program

Pada gambar 2.3, kita mendeklarasikan terlebih dahulu variabel jumlah sebagai *integer* dengan nilai 0. Lalu, kita membuat perulangan for yang mendeklarasikan i sebagai integer dengan nilai 1, i lebih kecil atau sama dengan 10, dan $i++$ (setiap i ditambah 1). Program akan berjalan dengan mengambil nilai i dengan awal nilai i

adalah 1 karena $1 \leq 10$, maka program akan mencetak nilai i pada *syntax print*. Kita membuat persamaan yang menyatakan bahwa $\text{jumlah} = \text{jumlah} + i$ yang mana karena jumlah awal adalah 0 dan i adalah 1, maka jumlah baru adalah $0+1=1$.

Kemudian, kita juga menambahkan percabangan *if* ketika i kecil dari 10 maka cetak "+". Perulangan *for* akan terus berlanjut hingga i lebih dari 10. Jika i sudah lebih dari 10 maka kita akan mendapatkan nilai jumlah. Kita dapat mencetak "Jumlah = " + jumlah yang telah kita peroleh dari perulangan *for* sebelumnya. Hasil dari program persis pada output pada gambar 2.3

2.1.3 Perulangan For 4



```
1 package pekan5;
2 import java.util.Scanner;
3 public class PerulanganFor4 {
4
5     public static void main(String[] args) {
6         int jumlah=0;
7         int batas;
8         Scanner input = new Scanner (System.in);
9         System.out.print ("Masukkan Nilai Batas : ");
10        batas = input.nextInt ();
11        input.close();
12        for (int i=1; i<=batas; i++) {
13            System.out.print (i);
14            jumlah = jumlah + i;
15            if (i<batas) {
16                System.out.print (" + ");
17            }else {System.out.print (" = ");
18            }
19        }
20        System.out.println (jumlah);}}
21
```

The console output shows: "Masukkan Nilai Batas : 10" followed by "1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55".

Gambar 2.4

2.1.3.1 Penjelasan Kode Program

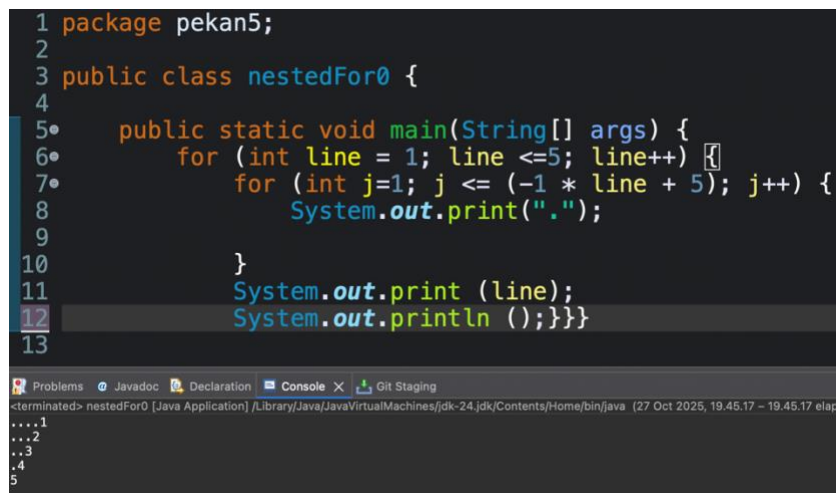
Pada gambar 2.4, kita mendeklarasikan terlebih dahulu variabel *jumlah* sebagai integer dengan nilai 0 dan variabel *batas* sebagai integer. Kita membuat *syntax input* pada java untuk menginputkan nilai *batas*. Setelah itu, kita buat perulangan *for* dengan mendeklarasikan variabel i sebagai integer dengan nilai 1, $i \leq \text{batas}$, dan $i++$. Program akan berjalan dengan mengambil nilai i dengan awal nilai i adalah 1 karena $1 \leq 10$,

maka program akan mencetak nilai i pada *syntax print*. Kita membuat persamaan yang menyatakan bahwa $\text{jumlah} = \text{jumlah} + i$ yang mana karena jumlah awal adalah 0 dan i adalah 1, maka jumlah baru adalah $0+1=1$.

Selanjutnya kita membuat percabangan, jika $i < \text{batas}$ maka kita akan mencetak " + ". Namun, jika i tidak lebih kecil dari batas yang telah diinputkan maka kita akan mencetak " = ". Perulangan for akan terus berlanjut hingga i lebih dari batas. Jika i sudah lebih dari batas maka kita akan mendapatkan nilai jumlah. Jika kita menginputkan nilai batas dengan angka 10, maka output akan persis seperti gambar 2.4.

2.2 Nested For

2.2.1 Nested For 0



```
1 package pekan5;
2
3 public class nestedFor0 {
4
5     public static void main(String[] args) {
6         for (int line = 1; line <= 5; line++) {
7             for (int j=1; j <= (-1 * line + 5); j++) {
8                 System.out.print(".");
9
10            }
11            System.out.print (line);
12            System.out.println ();}}
13
```

Problems Javadoc Declaration Console x Git Staging
<terminated> nestedFor0 [Java Application] /Library/Java/JavaVirtualMachines/jdk-24.jdk/Contents/Home/bin/java (27 Oct 2025, 19.45.17 - 19.45.17 elaps
....1
...2
..3
..4
5

Gambar 2.5

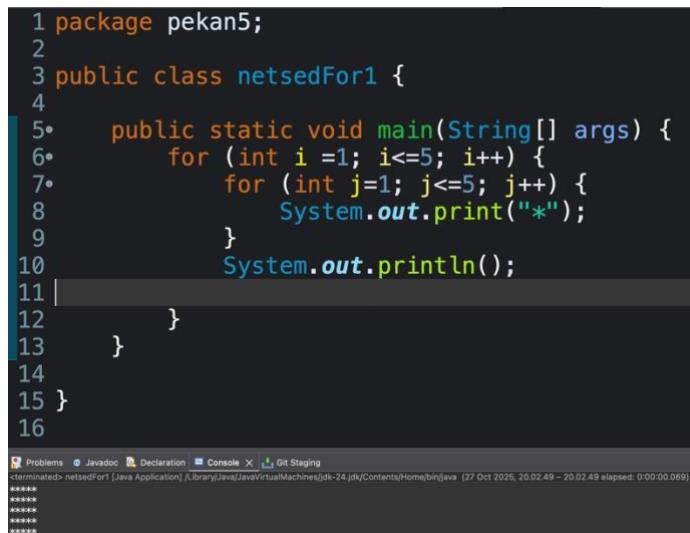
2.2.1.1 Penjelasan Kode Program

Pada gambar 2.5, kita membuat dua perulangan for yang saling bersarang (*nested for*). Perulangan pertama menggunakan variabel line yang dimulai dari 1 hingga 5 ($\text{line} \leq 5$). Artinya, program akan melakukan pengulangan sebanyak lima kali, dimulai dari baris ke-1 sampai baris ke-5. Di dalam perulangan pertama, terdapat perulangan kedua menggunakan variabel j yang dimulai dari 1 dan akan terus berjalan

selama j kurang dari atau sama dengan $(-1 * \text{line} + 5)$. Rumus ini menentukan berapa banyak titik (.) yang akan dicetak pada setiap baris. Ketika nilai line semakin besar, hasil perhitungan $(-1 * \text{line} + 5)$ akan semakin kecil, sehingga jumlah titik yang dicetak juga berkurang di setiap baris. Setelah perulangan dalam selesai mencetak titik, program mencetak nilai line dengan perintah `System.out.print(line);`.

Kemudian, perintah `System.out.println();` digunakan untuk membuat baris baru setelah mencetak angka tersebut. Dengan demikian, hasil akhir dari program ini akan menampilkan deretan titik yang semakin berkurang di tiap baris, diikuti oleh angka dari 1 sampai 5 di akhir setiap baris. Polanya akan tampak seperti segitiga menurun dengan angka di ujung kanan.

2.2.2 Nested For 1



```
1 package pekan5;
2
3 public class nestedFor1 {
4
5     public static void main(String[] args) {
6         for (int i=1; i<=5; i++) {
7             for (int j=1; j<=5; j++) {
8                 System.out.print("*");
9             }
10            System.out.println();
11        }
12    }
13 }
14
15 }
16
```


Gambar 2.6

2.2.2.1 Penjelasan Kode Program


Pada gambar 2.6 kita membuat perulangan for dengan mendeklarasikan variabel i sebagai integer dengan nilai 1, $i \leq 5$, $i++$. Di dalam perulangan for tersebut juga ada perulangan for yang mendeklarasikan variabel j sebagai integer dengan nilai

1, $j \leq 5$, dan $j++$, perulangan for yang berada di dalam juga memuat *syntax* print "*". Artinya nilai i dimulai dengan 1, selagi i belum lebih dari 5 maka program akan terus berulang. Jika i masih dibawah atau sama dengan 5 maka nilai j dimulai dari 1, selagi j belum lebih dari 5 maka program akan terus berjalan.

Jika j masih lebih kecil atau sama dengan 5 maka program akan mencetak "*". Namun, jika j sudah lebih dari 5 maka program akan langsung menambah i dengan 1, begitu seterusnya sampai i lebih dari 5. Jika i sudah lebih dari 5, maka output yang didapatkan akan persis seperti gambar 2.6.

2.2.3 Nested For 2

```
1 package pekan5;
2
3 public class nestedFor2 {
4
5     public static void main(String[] args) {
6         for (int i=0; i<=5; i++) {
7             for (int j = 0; j<=5; j++) {
8                 System.out.print (i+j+ " ");
9             }
10            System.out.println();
11        }
12    }
13 }
14
15 }
16
```



Gambar 2.7

2.2.3.1 Penjelasan Kode Program

Pada gambar 2.7, kita membuat dua perulangan for yang saling bersarang. Perulangan pertama menggunakan variabel i yang dideklarasikan sebagai integer dengan nilai awal 0, memiliki kondisi $i \leq 5$, dan setiap perulangan akan menambah nilai i sebanyak 1 ($i++$). Artinya, nilai i akan dimulai dari 0 hingga 5. Selama kondisi $i \leq 5$ masih terpenuhi, maka perulangan akan terus dijalankan.

Di dalam perulangan i, terdapat perulangan for kedua yang menggunakan variabel j, juga dimulai dari 0, dengan kondisi $j \leq 5$, dan setiap iterasi menambah nilai j sebanyak 1 ($j++$). Pada perulangan bagian dalam ini, program akan mencetak hasil penjumlahan antara i dan j menggunakan perintah `System.out.print(i + j + " ");`. Artinya, setiap kali nilai j bertambah, program akan menampilkan hasil dari $i + j$ yang dipisahkan dengan spasi.

Setelah perulangan j selesai (ketika j sudah lebih dari 5), program menjalankan perintah `System.out.println();` untuk membuat baris baru, sehingga hasil dari setiap nilai i dicetak di baris yang berbeda. Dengan demikian, output yang dihasilkan akan membentuk pola tabel angka, di mana setiap baris memperlihatkan hasil penjumlahan antara i dan j dari 0 hingga 5 seperti pada gambar 2.7.

BAB III

KESIMPULAN

3.1 Ringkasan Hasil Praktikum

Praktikum mengenai penggunaan *for* dan *nested for* dalam bahasa pemrograman Java memberikan pemahaman mendalam mengenai cara kerja struktur pengulangan sebagai salah satu kontrol alur program. Melalui percobaan yang dilakukan, mahasiswa belajar bahwa perulangan *for* berfungsi mengeksekusi blok perintah secara berulang berdasarkan kondisi tertentu tanpa perlu menulis kode yang sama berulang kali. Struktur ini membantu meningkatkan efisiensi penulisan program dan mempermudah pelaksanaan proses berulang secara sistematis.

Selama kegiatan praktikum, mahasiswa diperkenalkan pada tiga komponen utama dalam perulangan *for*, yaitu inisialisasi, kondisi, dan perubahan nilai variabel. Ketiga komponen tersebut bekerja secara berurutan dan saling berkaitan dalam mengatur jalannya eksekusi perulangan. Pemahaman terhadap mekanisme ini memungkinkan mahasiswa untuk mengendalikan alur program dengan lebih efektif, baik untuk menampilkan deret angka maupun melakukan perhitungan berulang yang bersifat numerik.

Selain mempelajari perulangan tunggal, mahasiswa juga dikenalkan dengan konsep *nested for* atau perulangan bersarang. Struktur ini memungkinkan adanya satu perulangan di dalam perulangan lain, yang sangat berguna untuk menampilkan pola dua dimensi seperti tabel, matriks, atau bentuk karakter tertentu. Melalui penerapan *nested for*, mahasiswa dilatih berpikir secara logis dan sistematis karena harus memahami hubungan bertingkat antarperulangan yang saling memengaruhi hasil akhir program.

Dari hasil praktikum dapat disimpulkan bahwa penerapan *for* dan *nested for* tidak hanya terbatas pada pembuatan pola visual, tetapi juga memiliki peran penting

dalam pengolahan data yang lebih kompleks. Struktur ini digunakan dalam berbagai penerapan pemrograman seperti manipulasi array dua dimensi, algoritma pencarian, serta pemrosesan data numerik. Oleh karena itu, pemahaman terhadap konsep perulangan menjadi dasar penting sebelum mempelajari algoritma dan struktur data yang lebih lanjut.

Secara keseluruhan, praktikum ini melatih mahasiswa untuk berpikir logis, analitis, dan terstruktur dalam menyusun program. Pemahaman terhadap konsep *looping* tidak hanya memperkuat kemampuan teknis dalam menulis kode Java, tetapi juga menumbuhkan cara berpikir komputasional yang efisien dan terorganisir. Dengan bekal ini, mahasiswa diharapkan mampu mengembangkan solusi pemrograman yang efektif, kreatif, dan dapat diterapkan pada berbagai permasalahan nyata di bidang teknologi informasi.

DAFTAR PUSTAKA

- [1] Oracle, *The Java® Tutorials — Primitive Data Types*. [Online]. Available: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>
- [2] GeeksforGeeks, *Java Data Types – GeeksforGeeks*. [Online]. Available: <https://www.geeksforgeeks.org/data-types-in-java>