

LAPORAN PRATIKUM
ALGORITMA PEMROGRAMAN DAN PEMROGRAMAN

PERULANGAN *FOR*

DISUSUN OLEH:

Rayya Syaquinah Putri Hasibuan

2511532007

DOSEN PENGAMPU:

Dr. Wahyudi, S.T. M.T

ASISTEN PRATIKUM:

Aufan Taufiqurrahman



PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji syukur penulis panjatkan kehadiran Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga laporan praktikum ini dapat diselesaikan dengan baik. Laporan ini disusun sebagai salah satu bentuk pertanggungjawaban pelaksanaan praktikum Algoritma Pemrograman dengan judul Perulangan *For*. Laporan ini bertujuan untuk mendokumentasikan prosedur, hasil, serta pembahasan dari kegiatan praktikum, sekaligus sebagai sarana pembelajaran bagi penulis untuk lebih memahami materi terkait.

Penulis menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan demi penyempurnaan laporan di masa mendatang. Akhir kata, penulis mengucapkan terima kasih kepada dosen pengampu, asisten pratikum, serta semua pihak yang telah membantu sehingga laporan ini dapat terselesaikan.

Padang, 29 Oktober 2025

Rayya Syaqqinah Putri Hasibuan

DAFTAR ISI

Contents

KATA PENGANTAR.....	ii
DAFTAR ISI	iii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Tujuan Pratikum.....	1
1.3 Manfaat Pratikum.....	1
BAB II PEMBAHASAN	2
2.1 Deskripsi Materi Pratikum	2
2.2 Perulangan For 1	3
2.3 Perulangan For 2	4
2.4 Perulangan For 3	5
2.5 Perulangan For 4	6
2.6 Nested For 0	7
2.7 Nested For 1	8
2.8 Nested For 2	9
BAB III KESIMPULAN DAN SARAN	11
3.1 KESIMPULAN	11
3.2 SARAN	11
DAFTARPUSTAKA	12

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam pemrograman, proses pengulangan merupakan salah satu konsep dasar yang sangat penting untuk menyelesaikan berbagai permasalahan yang bersifat repetitif. Tanpa pengulangan, programmer harus menulis kode yang sama berulang kali, sehingga tidak efisien dan rawan kesalahan.

Pada bahasa Java, salah satu struktur pengulangan yang paling umum digunakan adalah *perulangan for*. Struktur ini memungkinkan program menjalankan satu atau beberapa perintah secara berulang dengan jumlah iterasi yang telah ditentukan. Melalui pemahaman terhadap *looping for*, mahasiswa dapat mengontrol alur eksekusi program dengan lebih efektif dan efisien.

1.2 Tujuan Pratikum

Tujuan dari praktikum ini adalah agar siswa memahami cara kerja pengulangan *for* pada bahasa pemrograman Java. Melalui praktikum ini, mahasiswa belajar bagaimana membuat program yang bisa menjalankan perintah secara berulang tanpa harus menulis kode berkali-kali. Selain itu, praktikum ini juga bertujuan melatih kemampuan dalam membuat logika program yang teratur dan efisien.

1.3 Manfaat Pratikum

Manfaat dari praktikum pengulangan *for* pada Java adalah membantu siswa memahami cara berpikir logis dan teratur dalam membuat program. Melalui praktikum ini, mahasiswa menjadi lebih teliti dalam menulis kode karena kesalahan kecil dapat memengaruhi hasil program. Selain itu, mahasiswa juga belajar menyelesaikan masalah yang memerlukan pengulangan dengan lebih mudah dan efisien, seperti menghitung deret angka atau mencetak pola tertentu.

BAB II PEMBAHASAN

Pada praktikum ini dilakukan percobaan untuk memahami cara kerja perulangan *for* pada Java. Perulangan *for* digunakan untuk menjalankan perintah secara berulang dengan jumlah yang sudah ditentukan. Melalui percobaan ini, siswa belajar melihat bagaimana program melakukan pengulangan langkah demi langkah hingga menghasilkan output yang diinginkan.

2.1 Deskripsi Materi Pratikum

Proses mengulangi satu atau lebih (perulangan *if*) pernyataan tanpa henti selama kondisi yang telah ditentukan terpenuhi disebut sebagai *looping*. Secara umum, sebuah variabel disiapkan untuk iterasi atau penanda variabel untuk menandakan akhir dari loop. Persyaratan awal akan menetapkan: 1. Penghitungan akan dimulai dari 1 ($i = 1$); 2. Berapa banyak penghitungan? 3. Saya akan menambah +1 ($i++$) pada setiap iterasi hingga $i \leq 10$. Nilai penghitungan disimpan dalam variabel i pada loop *for*. Akibatnya, nilai i akan selalu bertambah satu pada setiap pengulangan. karena bagian $i++$ adalah tempat kita menentukan hal tersebut.

Salah satu jenis *nested for* menggunakan perintah *for*. Oleh karena itu, *nested for* adalah bentuk di mana pernyataan *for* mengandung pernyataan *for* lainnya. Pernyataan *if* yang berada di dalam pernyataan *if* lainnya disebut sebagai *if* bersarang. Dalam Java, kondisi *if* yang ditulis di dalam kondisi *if* luar disebut sebagai pernyataan *nested for*.

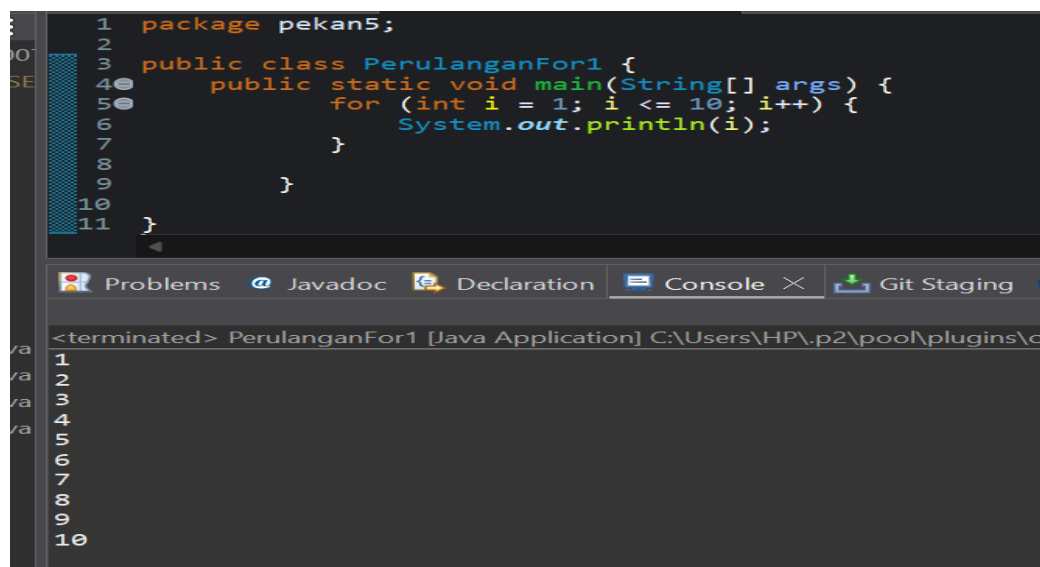
Pernyataan *nested for* berfungsi serupa dengan pernyataan pengambilan keputusan lainnya seperti *if*, *else*, *if..else*, dan sebagainya. Jika kondisi yang dinyatakan dalam pernyataan *if* benar, pernyataan ini akan menjalankan blok kode.

Di sisi lain, blok kode ditempatkan di dalam blok *if* lainnya dalam pernyataan *nested if*. Hanya ketika kondisi luar benar, blok kode di dalamnya akan dijalankan. Untuk

memeriksa banyak kondisi dalam program dan mengembalikan *respons* yang berbeda berdasarkan kondisi pengujian, kita memerlukan pernyataan *nested if*.

2.2 Perulangan For 1

Perulangan *for* digunakan untuk mengetahui jumlah iterasi yang perlu dieksekusi. Perulangan ini digunakan untuk mengulang data dalam array atau untuk menjalankan tugas yang dilakukan berulang kali.



```

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

```

The screenshot shows the output of the program in the console window, displaying the numbers 1 through 10, each on a new line.

Gambar 2.1 kode program dan output "PerulanganFor1"

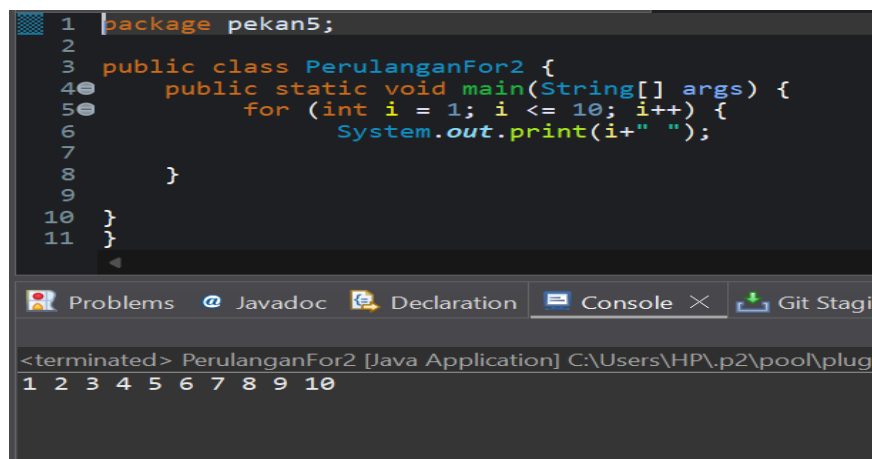
Langkah pengerjaan:

1. Membuat package pekan5 dan *new class* untuk pemrograman "PerulanganFor1".
2. Eksekusi program dimulai dari method *public static void main(String[] args)*
3. Menginisialisasi `int i = 1` → memberi nilai awal *i* sebesar 1.
4. Menginput kondisi `i <= 10` untuk mengecek apakah *i* masih kurang dari atau sama dengan 10.
5. Menginput `i++` untuk menambah nilai *i* satu setiap kali perulangan.
6. Menginput `System.out.println` agar hasil output berderet secara vertikal.
7. Dari hasil kode program ini ketika di *run* akan menghasilkan output seperti pada gambar 2.1

Program Java ini menggunakan struktur pengulangan *for* yang berjalan dari nilai awal 1 hingga 10, dengan setiap iterasi mencetak nilai variabel *i* ke konsol. Program ini mendemonstrasikan penggunaan perulangan *for* yang sederhana dan efektif untuk menampilkan angka berurutan dalam rentang tertentu. Output yang dihasilkan adalah angka 1 sampai 10 yang ditampilkan baris per baris secara horizontal karena menggunakan *System.out.println* pada program. Kode ini menunjukkan pemahaman dasar tentang kontrol alur dalam pemrograman Java.

2.3 Perulangan For 2

Masih sama dengan perulangan *for* 2, perulangan *for* digunakan untuk mengetahui jumlah iterasi yang perlu dieksekusi. Perulangan ini digunakan untuk mengulang data dalam array atau untuk menjalankan tugas yang dilakukan berulang kali



```

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

```

Problems Javadoc Declaration Console X Git Stagi

<terminated> PerulanganFor2 [Java Application] C:\Users\HP\p2\pool\plug

1 2 3 4 5 6 7 8 9 10

Gambar 2.2 kode program dan output "PerulanganFor2"

Langkah pengerjaan:

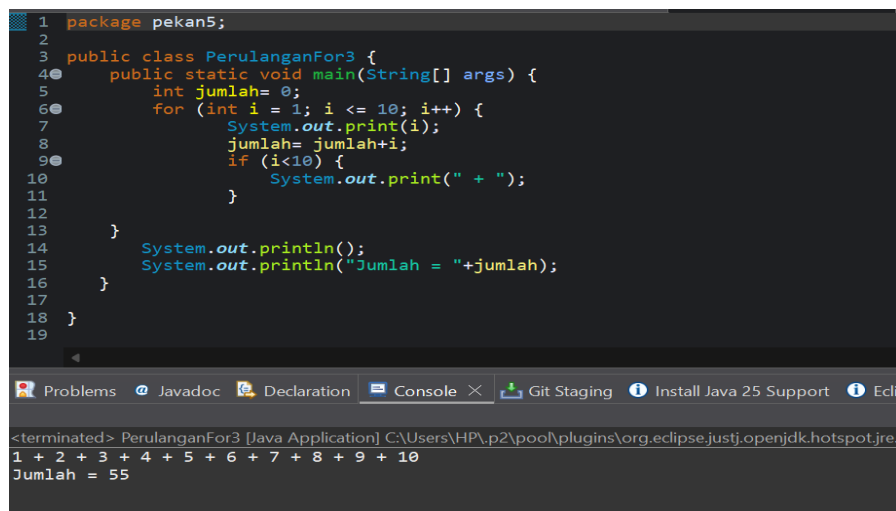
1. Membuat package pekan5 dan *new class* untuk pemrograman "PerulanganFor1".
2. Eksekusi program dimulai dari method *public static void main(String[] args)*
3. Menginisialisasi *int i = 1* → memberi nilai awal *i* sebesar 1.
4. Menginput kondisi *i <= 10* untuk mengecek apakah *i* masih kurang dari atau sama dengan 10.
5. Menginput *i++* untuk menambah nilai *i* satu setiap kali perulangan.

6. Menginput `System.out.print(i + " ");` dan dari perintah ini akan menghasilkan output (nilai variabel `i`) berderet secara horizontal diikuti oleh satu spasi.
7. Dari hasil kode program ini ketika di *run* akan menghasilkan output seperti pada gambar 2.2

Untuk analisis hasil, kode program perulangan *for* 1 dan perulangan *for* 2 tidak jauh berbeda, yang membedakan kedua program ini adalah pada kode program `System.out.println` dan `System.out.print` yang menghasilkan bentuk output berbeda (baris secara horizontal dan baris secara vertikal dengan spasi).

2.4 Perulangan For 3

Di dalam perulangan *for* 3 ini menggunakan aritmetika penjumlahan.



```

1 package pekan5;
2
3 public class PerulanganFor3 {
4     public static void main(String[] args) {
5         int jumlah= 0;
6         for (int i = 1; i <= 10; i++) {
7             System.out.print(i);
8             jumlah= jumlah+i;
9             if (i<10) {
10                 System.out.print(" + ");
11             }
12         }
13         System.out.println();
14         System.out.println("Jumlah = "+jumlah);
15     }
16 }
17
18
19

```

Problems Javadoc Declaration Console × Git Staging Install Java 25 Support Eclip

<terminated> PerulanganFor3 [Java Application] C:\Users\HP\p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre...

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

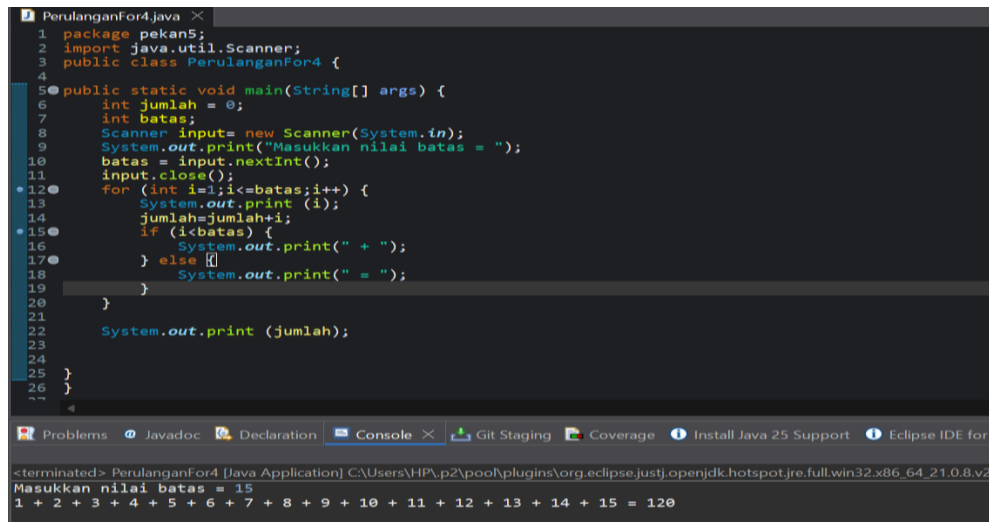
Gambar 2.4 kode program dan output "PerulanganFor3"

Langkah pengerjaan:

1. Inisialisasi `int jumlah = 0` → `int jumlah` adalah tempat menyimpan total.
2. Input perulangan *for* dimulai dari `i=1` hingga `i=10` (`i<= 10`)
3. Input iterasi nilai `i`, tambahkan `i` ke variabel `jumlah`, lalu input `i++`
4. Lalu run kode dan hasil program akan menampilkan output 1 hingga 10 yang dijumlahkan dengan hasil 55.

Program dengan kode yang telah disusun berhasil menampilkan angka 1 sampai 10 dengan tanda tambah diantaranya, lalu menghitung jumlah semuanya dengan hasil penjumlahannya adalah 55.

2.5 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 {
18                System.out.print(" = ");
19            }
20        }
21        System.out.print(jumlah);
22    }
23
24 }
25
26 }

```

Problems Javadoc Declaration Console Git Staging Coverage Install Java 25 Support Eclipse IDE for

<terminated> PerulanganFor4 [Java Application] C:\Users\HP\p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v2
 Masukkan nilai batas = 15
 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 = 120

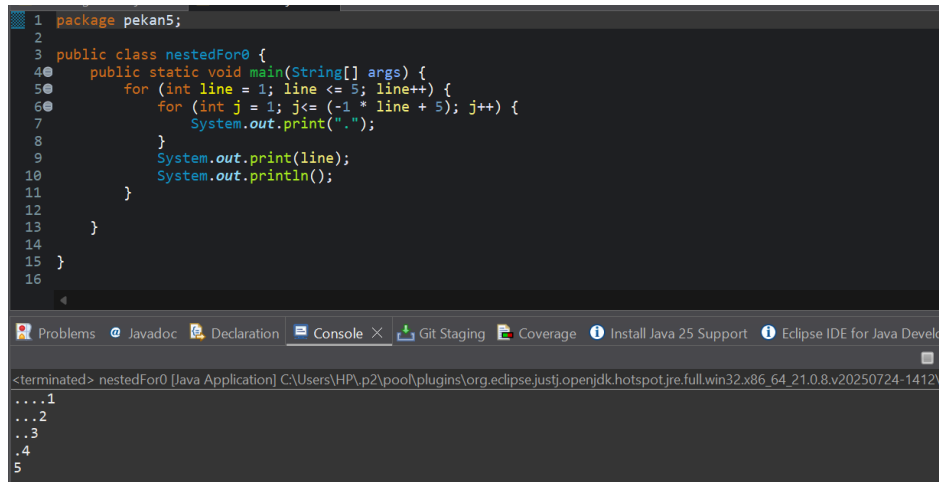
Gambar 2.5 kode program dan output "PerulanganFor4"

Langkah pengerjaan:

1. Pada program ini menggunakan *Scanner* `input = new Scanner`.
2. Inisialisasi `int jumlah = 0`, dan `int batas`.
3. Pada kode perulangan, `for (int i = 1; i <= batas; i++)`.
4. Pengguna menginput nilai `batas=15`, maka perulangan dimulai dari 1 dan berlanjut hingga 15.
5. Pada setiap iterasi, nilai `i` akan ditambahkan ke variabel `jumlah = jumlah+i`.
6. Pada output menampilkan hasil penjumlahan 1 hingga 15 yaitu 120.

Sesuai dengan kode program pada gambar, program ini berhasil menjumlahkan bilangan bulat dari 1 hingga 15.

2.6 Nested For 0



```

1 package pekan5;
2
3 public class nestedFor0 {
4     public static void main(String[] args) {
5         for (int line = 1; line <= 5; line++) {
6             for (int j = 1; j <= (-1 * line + 5); j++) {
7                 System.out.print(".");
8             }
9             System.out.print(line);
10            System.out.println();
11        }
12    }
13 }
14
15 }
16

```

Problems Javadoc Declaration Console X Git Staging Coverage Install Java 25 Support Eclipse IDE for Java Develo

<terminated> nestedFor0 [Java Application] C:\Users\HP\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\j

```

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

```

Gambar 2.5 adalah kode program dan hasil output dari program "NestedFor0"

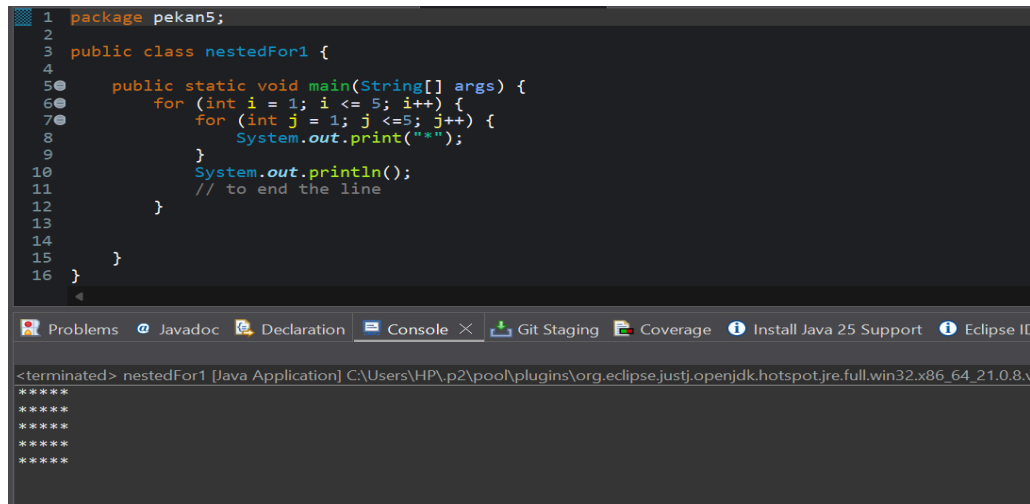
Langkah pengerjaan:

1. Deklarasi kelas dan metode utama (program dimulai dengan `public class nestedFor0` dan `public static void main (String[] args)` sebagai pintu masuk eksekusi.
2. Membuat perulangan luar `for (int line = 1; line <= 5; line++)`.
3. Membuat perulangan dalam `for (int j = 1; j <= (-1 * line + 5); j++)` yang berfungsi mencetak titik(.) beberapa kali pada setiap baris.
4. Mencetak titik karakter `System.out.print(".");` dijalankan berulang kali sesuai jumlah hasil perulangan j.
5. Mencetak angka baris setelah perulangan dalam selesai, `System.out.print(line);` menampilkan angka baris di akhir deretan titik.
6. Pindah ke baris berikutnya `System.out.println();` menambahkan baris baru agar output berikutnya muncul di baris selanjutnya.
7. Perulangan berlanjut hingga baris ke-5 setiap iterasi *line* bertambah satu, sehingga jumlah titik berkurang satu per baris, menghasilkan bentuk pola menurun.

Program ini menggunakan konsep *nested for* untuk mencetak pola titik dan angka dengan jumlah titik yang semakin berkurang setiap barisnya. Kode program `(-1 * line + 5)` membuat jumlah titik menurun seiring naiknya nilai *line*, sehingga

menghasilkan output berupa segitiga miring ke kiri dengan bentuk gambar 2.5. Secara logika, kode ini menunjukkan hubungan terbalik antara nilai *line* dan jumlah titik, serta mendemonstrasikan cara mengontrol tampilan pola menggunakan dua tingkat perulangan *for*.

2.7 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            // to end the line
12        }
13    }
14 }
15
16 }

```

Problems Javadoc Declaration Console Git Staging Coverage Install Java 25 Support Eclipse IDE

<terminated> nestedFor1 [Java Application] C:\Users\HP\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v

```

*****
*****
*****
*****
*****

```

Gambar 2.7 adalah kode program dan output dari program "NestedFor1"

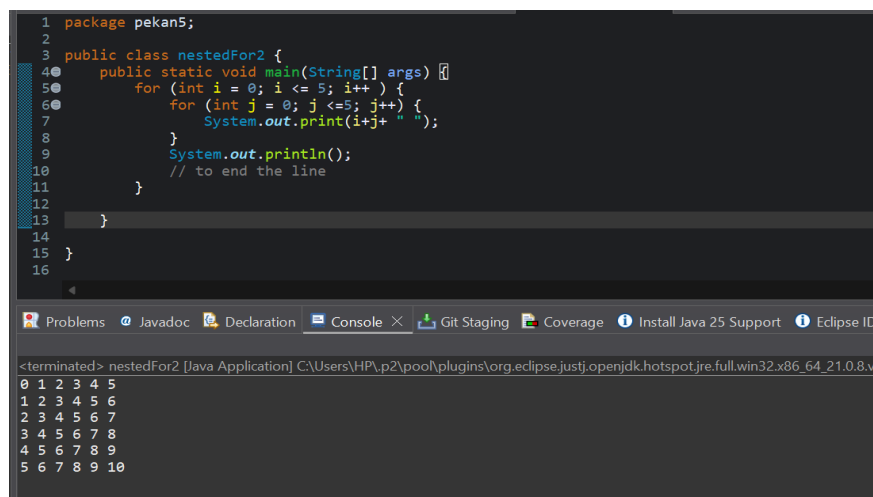
Langkah pengerjaannya:

1. Deklarasi kelas dan metode utama program dimulai dengan *public class nestedFor1* dan *public static void main(String[] args)* sebagai struktur dasar Java.
2. Membuat perulangan luar (baris) *for (int i = 1; i <= 5; i++)* menentukan ada 5 baris yang akan dicetak. Variabel *i* berfungsi sebagai penghitung baris.
3. Membuat perulangan dalam (kolom) *for (int j = 1; j <= 5; j++)* menentukan bahwa pada setiap baris, akan dicetak 5 bintang. Variabel *j* menghitung posisi kolom dalam satu baris.
4. Mencetak simbol bintang *System.out.print("*");* mencetak tanda * tanpa membuat baris baru, sehingga bintang muncul berdampingan secara horizontal.
5. Pindah ke baris berikutnya Setelah satu baris penuh bintang selesai, *System.out.println();* digunakan agar kursor berpindah ke baris berikutnya untuk mencetak deretan bintang baru.

6. Perulangan berlanjut hingga baris ke-5 karena *loop* luar (i) berjalan dari 1 sampai 5, proses ini mengulang sebanyak 5 kali dan menghasilkan pola persegi bintang.

Program ini menggunakan konsep *nested loop* untuk mencetak pola persegi bintang berukuran 5x5. Perulangan luar mengatur jumlah baris, sementara perulangan dalam mengatur jumlah kolom dalam tiap baris. Setiap baris berisi lima tanda *, dan *System.out.println()* memastikan pola tercetak dalam bentuk kotak. Hasil akhirnya berupa bintang persegi pada output gambar 2.7 yang menunjukkan koordinasi sempurna antara dua loop bersarang dalam menghasilkan pola dua dimensi.

2.8 Nested For 2



```

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

```

Problems Javadoc Declaration Console X Git Staging Coverage Install Java 25 Support Eclipse ID

<terminated> nestedFor2 [Java Application] C:\Users\HP\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v

```

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 2.8 adalah kode program dan hasil output dari program ”NestedFor2”

Langkah pengerjaan:

1. Membuat package, package pekan5;
2. Membuat *class nestedFor2*, *public class nestedFor2* {
3. Membuat method main *public static void main(String[] args) {*
4. Membuat perulangan pertama (*outer loop*) *for (int i = 0; i <= 5; i++) {*, i dimulai dari 0 akan berjalan sampai i = 5 (total 6 baris)

5. Membuat perulangan kedua (*inner loop*) `for (int j = 0; j <= 5; j++)` { j dimulai dari 0 akan berjalan sampai j = 5 (total 6 kolom)

6. Menampilkan nilai penjumlahan `System.out.print(i + j + " ");` Setiap perulangan mencetak hasil penjumlahan i + j dalam satu baris. Pindah baris setelah *inner loop* selesai `System.out.println();`

Program ini menggunakan nested loop untuk menampilkan tabel penjumlahan. Setiap baris meningkat 1 dari baris sebelumnya, menunjukkan pola aritmatika sederhana.

BAB III

KESIMPULAN DAN SARAN

3.1 KESIMPULAN

Dari hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa struktur *perulangan for* pada Java berfungsi untuk menjalankan perintah secara berulang dengan jumlah iterasi yang sudah ditentukan. Melalui berbagai percobaan, siswa dapat memahami cara kerja *inisialisasi*, *kondisi*, dan *pembaruan* dalam perulangan serta bagaimana ketiganya saling berhubungan untuk mengontrol jalannya program.

Selain itu, praktikum ini membuktikan bahwa penggunaan *for loop* dapat membuat program lebih efisien, terstruktur, dan mudah dibaca. Konsep *nested loop* juga memperluas pemahaman mahasiswa dalam membuat pola dan logika dua dimensi yang lebih kompleks. Dengan memahami perulangan, mahasiswa dapat menulis kode yang lebih rapi dan efektif dalam menyelesaikan berbagai permasalahan pemrograman.

3.2 SARAN

Dalam melakukan praktikum, disarankan agar mahasiswa lebih teliti saat menentukan batas perulangan dan kondisi *loop*, karena kesalahan kecil dapat menyebabkan program berjalan tidak sesuai atau bahkan tidak berhenti. Pemahaman teori sebelum praktik juga penting agar proses penulisan kode berjalan lebih lancar dan hasilnya sesuai dengan yang diharapkan.

DAFTAR PUSTAKA

- [1] F. A. Rahman, Bab 2: Perulangan dalam JavaScript, Universitas Komputer Indonesia (UNIKOM), Bandung, 2021. [Online]. Available: <https://repository.unikom.ac.id/69035/1/Bab%202%20Perulangan%20dalam%20JavaScript.pdf>
- [2] Scaler Topics, "Nested if statement in Java," Scaler, 2023. [Online]. Available: <https://www.scaler.com/topics/nested-if-statement-in-java/>
- [3] Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi, Perulangan Bersarang, LMS SPADA Kemdikbud, 2022. [Online]. Available: https://lmsspada.kemdiktisaintek.go.id/pluginfile.php/685472/mod_resource/content/1/perulangan%20bersarang.pdf