

LAPORAN PRAKTIKUM ALGORITMA DAN PEMROGRAMAN

PERULANGAN FOR PEMROGRAMAN JAVA

disusun Oleh:

Nama: Amiratul Fadhilah

NIM: 2511532023

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

Asisten Praktikum: Jovantri Immanuel Gulo



**DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS
PADANG**

2025

KATA PENGANTAR

Puji syukur ke hadirat Allah Yang Maha Esa atas rahmat dan karunia-Nya, sehingga laporan praktikum dengan judul “Perulangan For” ini dapat diselesaikan dengan baik dan tepat waktu.

Laporan ini disusun sebagai bagian dari kegiatan praktikum Algoritma dan Pemrograman yang bertujuan untuk memperdalam pemahaman mengenai konsep dasar kontrol alur program di Java. Secara khusus, laporan ini akan membahas implementasi struktur perulangan for, termasuk sintaksis dasarnya, penerapan perulangan bersarang (*nested loop*), dan studi kasus terkait yang menjadi fondasi dalam menangani proses repetitif (berulang) pada sebuah program.

Penulis menyadari bahwa laporan ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk perbaikan di masa mendatang. Semoga laporan ini dapat memberikan manfaat serta pemahaman yang lebih mendalam bagi pembaca.

Padang, 30 Oktober 2025

Penulis

DAFTAR ISI

| | |
|---|----|
| KATA PENGANTAR | i |
| DAFTAR ISI | ii |
| BAB I PENDAHULUAN | |
| 1.1 Latar Belakang | 1 |
| 1.2 Tujuan Praktikum..... | 1 |
| 1.3 Manfaat Praktikum..... | 2 |
| BAB II PEMBAHASAN | |
| 2.1 Dasar Teori..... | 3 |
| 2.1.1 Struktur Kontrol Perulangan (<i>Looping</i>) | |
| 2.1.2 Perulangan For | |
| 2.1.3 Perulangan Bersarang (<i>Nested For Loop</i>) | |
| 2.1.4 Operator <i>Increment</i> dan <i>Decrement</i> | |
| 2.2 Langkah Praktikum | 4 |
| 2.3 Kode Program | 4 |
| 2.2.1 Program Perulangan Sederhana (PerulanganFor1) | |
| 2.2.2 Program Perulangan Sederhana (PerulanganFor2) | |
| 2.2.3 Program Perulangan Akumulasi (PerulanganFor3) | |
| 2.2.4 Program Perulangan Dinamis (PerulanganFor4) | |
| 2.2.5 Program Perulangan Bersarang (nestedFor0) | |
| 2.2.6 Program Perulangan Bersarang (nestedFor1) | |
| 2.2.7 Program Perulangan Bersarang (nestedFor2) | |
| BAB III KESIMPULAN | |
| 3.1 Kesimpulan | 17 |
| 3.2 Saran..... | 17 |
| DAFTAR PUSTAKA | 18 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam aktivitas pemrograman, sering ditemukan kebutuhan untuk menjalankan satu rangkaian perintah yang sama secara berulang kali. Menulis ulang kode untuk setiap pengulangan adalah hal yang tidak efisien. Cara manual seperti ini tidak hanya memakan waktu, tetapi juga meningkatkan risiko kesalahan dan membuat program menjadi sulit untuk diperbaiki atau dikembangkan.

Untuk mengatasi masalah pengulangan ini, bahasa pemrograman Java menyediakan sebuah mekanisme kontrol yang dikenal sebagai struktur perulangan atau *looping*. Mekanisme ini mengizinkan program untuk mengeksekusi blok kode yang sama berulang kali, selama sebuah kondisi tertentu terpenuhi. Salah satu struktur perulangan yang paling dasar dan penting adalah perulangan for. Struktur for umumnya digunakan pada situasi di mana jumlah pengulangan telah diketahui atau dapat ditentukan sebelum perulangan tersebut dimulai.

Oleh sebab itu, sangat penting bagi mahasiswa untuk menguasai cara kerja dan penggunaan perulangan for, termasuk juga variasinya seperti *nested loop*. Penguasaan materi ini sangat penting untuk melatih logika berpikir algoritmik dalam menyelesaikan masalah komputasi.

1.2 Tujuan Praktikum

Pelaksanaan praktikum ini memiliki beberapa tujuan khusus sebagai berikut:

1. Memahami konsep dasar, sintaksis, dan mekanisme kerja struktur kontrol perulangan for dalam bahasa Java.
2. Menerapkan perulangan for untuk penyelesaian studi kasus, baik perulangan sederhana maupun yang melibatkan proses akumulasi data.
3. Menganalisis penggunaan perulangan for yang melibatkan masukan dari pengguna
4. Memahami dan menerapkan konsep perulangan bersarang (*nested for*) untuk membuat pola atau memproses data dua dimensi.

5. Mengembangkan keterampilan analisis dan pemecahan masalah (*problem solving*) dengan pendekatan algoritmik.

1.3 Manfaat Praktikum

Manfaat yang diharapkan dari praktikum ini adalah:

1. Meningkatkan kemampuan membuat program Java yang efisien menggunakan perulangan for.
2. Mampu menerapkan logika perulangan untuk menyelesaikan masalah komputasi yang berulang.
3. Menjadi dasar yang kuat untuk belajar materi lanjutan.
4. Meningkatkan keterampilan menyusun laporan yang sistematis sebagai dokumentasi dan bahan belajar.

BAB II

PEMBAHASAN

2.1 Dasar Teori

2.2.1 Struktur Kontrol Perulangan (*Looping*)

Perulangan adalah sebuah mekanisme dalam pemrograman yang mengizinkan satu atau serangkaian instruksi untuk dieksekusi berulang kali selama suatu kondisi tertentu terpenuhi. Dalam Java, terdapat beberapa jenis struktur perulangan, antara lain *for*, *while*, dan *do-while*.

2.2.2 Perulangan For

Perulangan *for* adalah bentuk perulangan definite (terhitung) yang ideal digunakan ketika jumlah iterasi (pengulangan) sudah diketahui secara pasti. Sintaksis perulangan *for* terdiri dari tiga komponen utama yang mengontrol eksekusinya. Bagian pertama adalah inisialisasi, sebuah ekspresi yang dieksekusi satu kali saja pada awal perulangan, biasanya digunakan untuk mendeklarasikan dan memberi nilai awal pada variabel kontrol *loop* (misalnya, *int i = 0*).

Bagian kedua adalah kondisi, yaitu ekspresi *boolean* yang dievaluasi sebelum setiap iterasi. Jika kondisi ini bernilai *true*, blok kode di dalam *loop* akan dieksekusi; namun jika bernilai *false*, perulangan akan berhenti. Bagian terakhir adalah iterasi, sebuah ekspresi yang dieksekusi pada akhir setiap putaran *loop*, tepat sebelum kondisi dievaluasi kembali. Bagian iterasi ini umumnya berfungsi untuk memperbarui variabel kontrol, sering kali menggunakan operator *increment* atau *decrement*.

2.1.3 Perulangan Bersarang (*Nested For Loop*)

Perulangan bersarang adalah sebuah struktur di mana terdapat satu atau lebih perulangan di dalam perulangan lainnya. Perulangan yang berada di luar disebut *outer loop*, dan yang di dalam disebut *inner loop*. Mekanismenya, untuk setiap satu iterasi pada *outer loop*, *inner loop* akan menyelesaikan seluruh siklus iterasinya dari awal hingga akhir.

2.2.4 Operator *Increment* dan *Decrement*

Operator *increment* (++) dan *decrement* (--) adalah operator *unary* yang berfungsi untuk menambah atau mengurangi nilai variabel numerik sebanyak satu. Operator *increment* (++) menambahkan nilai variabel sebanyak 1 (setara $i = i + 1$), sedangkan *decrement* (--) menguranginya sebanyak 1 (setara $i = i - 1$). Dalam perulangan *for*, operator ini sangat penting pada bagian iterasi untuk mengontrol hitungan agar maju (i++) atau mundur (i--).

2.2 Langkah Praktikum

Langkah-langkah yang dilakukan dalam pelaksanaan praktikum ini adalah sebagai berikut:

1. Menyiapkan perangkat yang telah dilengkapi dengan *Java Development Kit* (JDK) dan sebuah *Integrated Development Environment* (IDE) seperti Eclipse.
2. Membuka IDE dan membuat sebuah *project* Java baru.
3. Di dalam *project*, membuat beberapa *class* Java baru sesuai dengan nama berkas kode program yang akan diuji (misalnya, *PerulanganFor1.java*, *nestedFor0.java*, dst.).
4. Menuliskan kode program yang telah disediakan.
5. Melakukan kompilasi (*compile*) dan eksekusi (*run*) pada setiap *class* program.
6. Mengamati dan mencatat keluaran (*output*) yang dihasilkan oleh setiap program.
7. Menganalisis alur eksekusi kode, membandingkan *output* yang dihasilkan dengan teori, dan memahami bagaimana setiap baris kode berkontribusi pada hasil akhir.
8. Menyusun laporan praktikum sesuai dengan format yang telah ditetapkan.

2.3 Kode Program

2.2.1 Program Perulangan Sederhana (PerulanganFor1)

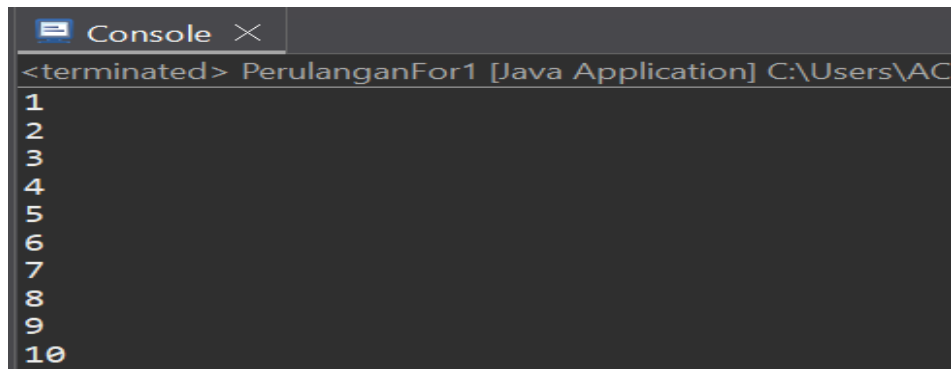
```
1    package pekan5;
2
3    public class PerulanganFor1 {
4
5        public static void main(String[] args) {
6            for (int i = 1; i <= 10; i++) {
7                System.out.println(i);
8            }
9        }
10
11    }
```

Kode Program 2.1

Struktur dan Langkah Program:

1. `package pekan5;;` Mendeklarasikan bahwa *class* ini berada dalam *package* (folder) bernama *pekan5*.
2. `public class PerulanganFor1 {...};` `public` adalah *access modifier* yang berarti *class* ini dapat diakses dari mana saja. `class` adalah kata kunci untuk mendefinisikan sebuah *class* Java dengan nama *PerulanganFor1*.
3. `public static void main(String[] args) {...};` Ini adalah *method* utama (*main method*). `public static void` adalah serangkaian kata kunci yang menetapkan *method* ini sebagai *entry point* (titik awal) eksekusi program Java.
4. `for (int i = 1; i <= 10; i++) {...};` Memulai struktur perulangan `for`.
 - `int i = 1`: Inisialisasi, variabel `i` dideklarasikan dan diberi nilai awal 1.
 - `i <= 10`: Kondisi, perulangan akan terus berjalan selama nilai `i` kurang dari atau sama dengan 10.
 - `i++`: Iterasi, nilai `i` akan ditambahkan 1 (increment) setiap kali blok kode *loop* selesai dieksekusi.
5. `System.out.println(i);` Mencetak nilai variabel `i` saat ini, kemudian `println` (print line) akan otomatis memindahkan kursor ke baris baru.

Output:

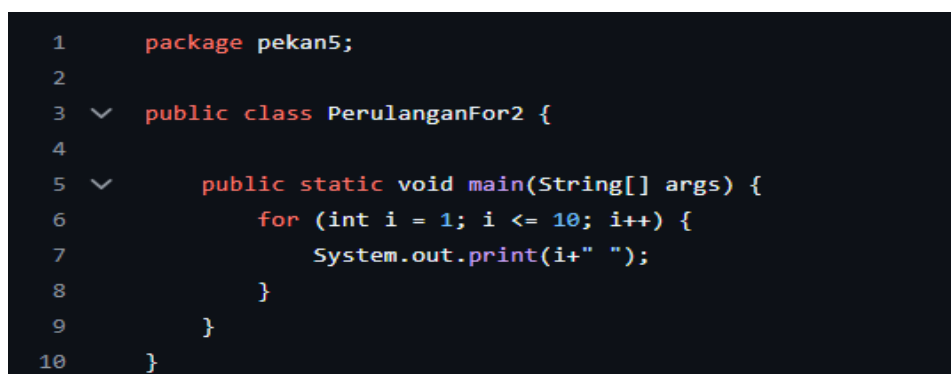


```
<terminated> PerulanganFor1 [Java Application] C:\Users\AC
1
2
3
4
5
6
7
8
9
10
```

Gambar 2.1

Program akan mengeksekusi *loop* sebanyak 10 kali. Pada setiap iterasi, program mencetak nilai *i* (dari 1, 2, 3, ... hingga 10). Karena menggunakan `println`, setiap angka dicetak pada baris yang terpisah, menghasilkan keluaran berupa daftar angka 1 sampai 10 secara vertikal.

2.2.2 Program Perulangan Sederhana (PerulanganFor2)



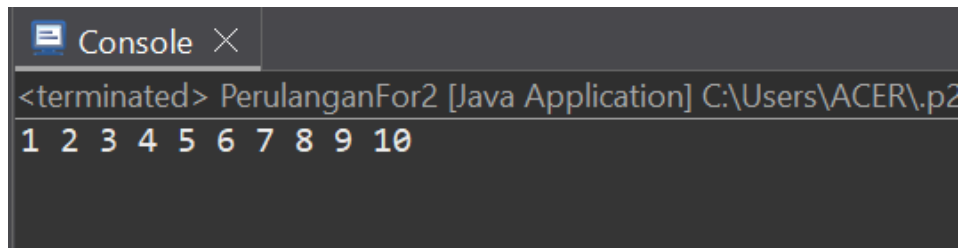
```
1 package pekan5;
2
3 public class PerulanganFor2 {
4
5     public static void main(String[] args) {
6         for (int i = 1; i <= 10; i++) {
7             System.out.print(i+" ");
8         }
9     }
10 }
```

Kode Program 2.2

Struktur dan Langkah Program:

1. `public class PerulanganFor2 {...}`: Mendefinisikan *class* `PerulanganFor2`.
2. `public static void main(String[] args) {...}`: Titik awal eksekusi program.
3. `for (int i = 1; i <= 10; i++) {...}`: Struktur perulangan yang sama dengan program sebelumnya, berulang 10 kali.
4. `System.out.print(i + " ");`: Mencetak nilai *i* saat ini, diikuti dengan sebuah karakter spasi (" "). Penggunaan *method* `print` (tanpa `ln`) membuat kursor *tetap* berada di baris yang sama setelah mencetak.

Output:



Gambar 2.2

Logika perulangan identik dengan program 2.1. Namun, karena menggunakan `System.out.print` dan menambahkan spasi, seluruh keluaran dicetak dalam satu baris horizontal: 1 2 3 4 5 6 7 8 9 10 .

2.2.3 Program Perulangan Akumulasi (PerulanganFor3)

```
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   }
17 }
18
19 }
```

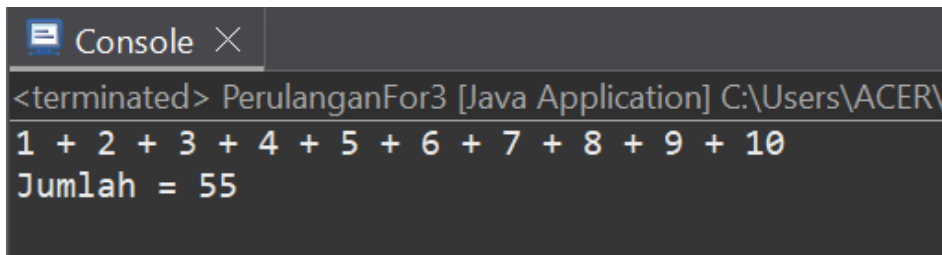
Kode Program 2.3

Struktur dan Langkah Program:

1. `public class PerulanganFor3 {...}`: Mendefinisikan *class* PerulanganFor3.
2. `public static void main(String[] args) {...}`: Titik awal eksekusi.
3. `int jumlah = 0;`: Mendeklarasikan variabel jumlah sebagai akumulator (penampung hasil) dengan nilai awal 0.
4. `for (int i = 1; i <= 10; i++) {...}`: Memulai perulangan 10 kali.

5. `System.out.print(i);`: Mencetak nilai `i` saat ini.
6. `jumlah = jumlah + i;`: Proses Akumulasi. Nilai `jumlah` yang lama diperbarui dengan menambahkan nilai `i` saat ini. (Misal: iterasi 1, `jumlah = 0 + 1`; iterasi 2, `jumlah = 1 + 2`; iterasi 3, `jumlah = 3 + 3`, dst.).
7. `if (i < 10) { ... }`: Struktur kondisional `if` untuk format pencetakan. Jika `i` belum mencapai 10, cetak " + ".
8. `System.out.println();`: Setelah *loop* selesai, pindah ke baris baru.
9. `System.out.println("Jumlah = " + jumlah);`: Mencetak hasil akhir dari variabel `jumlah`.

Output:



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

Gambar 2.3

Program ini akan mencetak deret penjumlahan. Baris pertama *output* adalah `1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10`. Selama proses ini, variabel `jumlah` mengakumulasi total nilai (`1+2+...+10 = 55`). Setelah *loop* selesai, baris kedua mencetak `Jumlah = 55`.

2.2.4 Program Perulangan Dinamis (PerulanganFor4)

```

1    package pekan5;
2
3    import java.util.Scanner;
4
5    public class PerulanganFor4 {
6
7        public static void main(String[] args) {
8            int jumlah=0;
9            int batas;
10           Scanner input= new Scanner (System.in);
11           System.out.print("Masukkan nilai batas = ");
12           batas= input.nextInt();
13           input.close();
14           for (int i=1; i<=batas; i++) {
15               System.out.print(i);
16               jumlah= jumlah+i;
17               if (i<batas) {
18                   System.out.print(" + ");
19               } else {
20                   System.out.print(" = ");
21               }
22           }
23           System.out.println(jumlah);
24       }
25   }

```

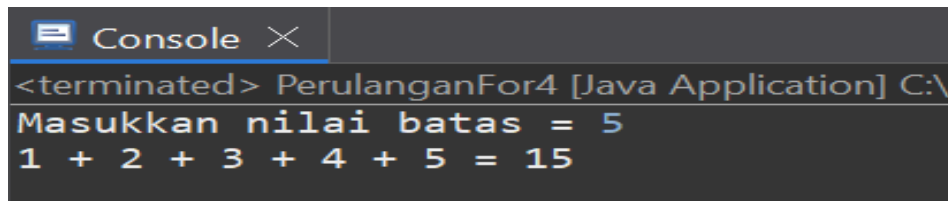
Kode Program 2.4

Struktur dan Langkah Program:

1. `import java.util.Scanner;`: Import library. Memberitahu Java untuk menggunakan *class* Scanner dari *package* java.util. Scanner dibutuhkan untuk membaca *input* dari pengguna.
2. `int jumlah = 0;` dan `int batas;`: Deklarasi variabel. *jumlah* sebagai akumulator, *batas* untuk menyimpan *input* pengguna.
3. `Scanner input = new Scanner(System.in);`: Membuat objek baru dari *class* Scanner yang akan membaca dari *standard input* (keyboard).
4. `System.out.print("Masukkan nilai batas = ");`: Menampilkan *prompt* ke pengguna.
5. `batas = input.nextInt();`: Menunggu pengguna memasukkan angka integer, lalu menyimpannya ke variabel *batas*.

6. `input.close();`: Menutup *scanner* setelah selesai digunakan (praktik yang baik untuk menghemat sumber daya).
7. `for (int i = 1; i <= batas; i++) { ... }`: Memulai perulangan. Perhatikan bagian Kondisi (`i <= batas`) yang kini bersifat dinamis, bergantung pada nilai yang dimasukkan pengguna.
8. `jumlah = jumlah + i;`: Melakukan akumulasi.
9. `if (i < batas) { ... } else { ... }`: Struktur kondisional if-else untuk format. Mencetak " + " jika belum mencapai batas, dan mencetak " = " jika i sama dengan batas.
10. `System.out.println(jumlah);`: Mencetak hasil akhir (total jumlah) di sebelah " = ".

Output:



```
<terminated> PerulanganFor4 [Java Application] C:\
Masukkan nilai batas = 5
1 + 2 + 3 + 4 + 5 = 15
```

Gambar 2.4

Program ini mirip dengan 2.3, namun jumlah perulangannya tidak tetap (tidak 10). Jika pengguna memasukkan 5 pada *prompt*, *output* akan menjadi: `1 + 2 + 3 + 4 + 5 = 15`.

2.2.5 Program Perulangan Bersarang (`nestedFor0`)

```

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             System.out.print(line);
11             System.out.println();
12         }
13     }
14
15 }

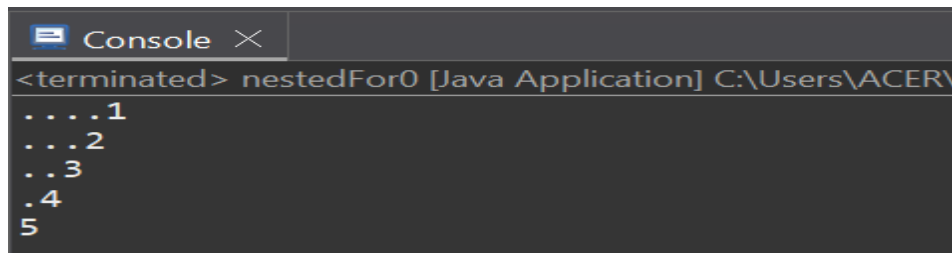
```

Kode Program 2.5

Struktur dan Langkah Program:

1. for (int line = 1; line <= 5; line++) { ... }: *Outer loop* untuk mengontrol baris, berjalan 5 kali.
2. for (int j = 1; j <= (-1 * line + 5); j++) { ... }: *Inner loop* pertama, bertugas mencetak titik (.). Kondisinya dinamis:
 - Saat line = 1, kondisi j <= 4. Mencetak 4 titik.
 - Saat line = 2, kondisi j <= 3. Mencetak 3 titik.
 - ...
 - Saat line = 5, kondisi j <= 0. *Loop* ini tidak berjalan.
3. System.out.print(line);: Dieksekusi setelah *inner loop* (loop titik) selesai. Mencetak nilai line saat ini.
4. System.out.println();: Pindah ke baris baru untuk iterasi *outer loop* berikutnya.

Output:

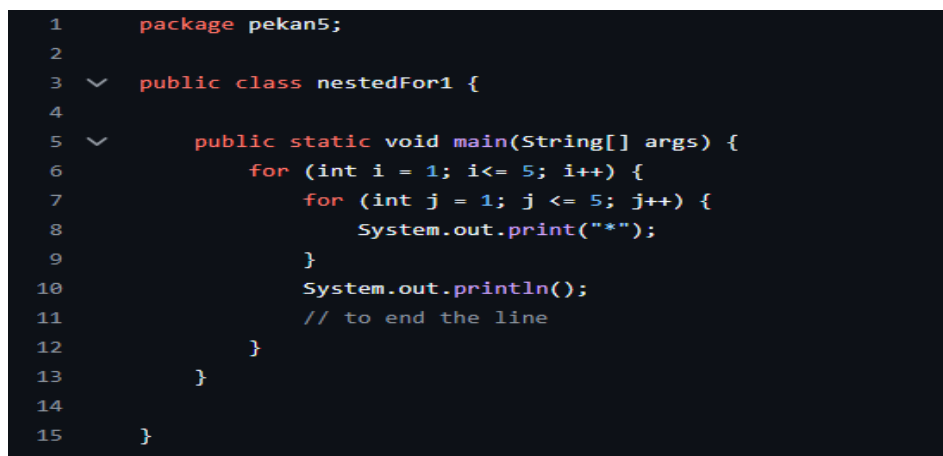


```
Console X
<terminated> nestedFor0 [Java Application] C:\Users\ACER\
....1
...2
..3
.4
5
```

Gambar 2.5

Program ini mencetak pola segitiga siku-siku rata kanan. Titik-titik digunakan sebagai *padding* di sebelah kiri, yang jumlahnya semakin berkurang, sementara angka di sebelah kanan sesuai dengan nomor baris.

2.2.6 Program Perulangan Bersarang (nestedFor1)



```
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 }
```

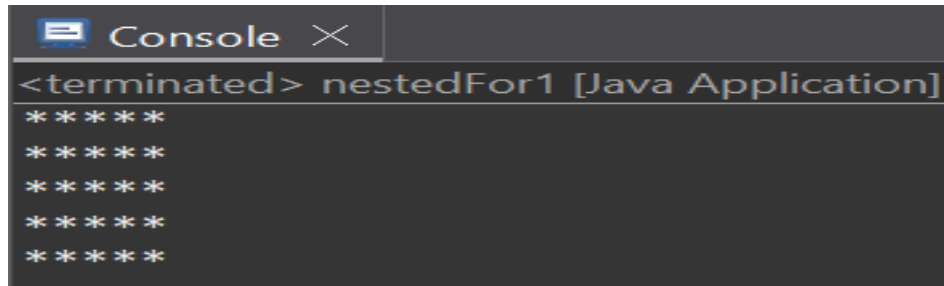
Kode Program 2.6

Struktur dan Langkah Program:

1. `public class nestedFor1 { ... }`: Mendefinisikan *class* nestedFor1.
2. `public static void main(String[] args) { ... }`: Titik awal eksekusi.
3. `for (int i = 1; i <= 5; i++) { ... }`: Ini adalah *Outer Loop* (Perulangan Luar). *Loop* ini mengontrol baris dan akan berjalan sebanyak 5 kali (i=1 sampai i=5).
4. `for (int j = 1; j <= 5; j++) { ... }`: Ini adalah *Inner Loop* (Perulangan Dalam). *Loop* ini mengontrol kolom dan akan berjalan sebanyak 5 kali (j=1 sampai j=5) untuk setiap iterasi *loop* luar.

5. `System.out.print("*");`: Mencetak satu karakter bintang tanpa pindah baris.
6. `System.out.println();`: Instruksi ini berada di dalam *outer loop* tetapi di luar *inner loop*. Ini adalah kunci dari *nested loop* untuk pola: dieksekusi setelah *inner loop* (kolom) selesai, berfungsi untuk pindah ke baris baru.

Output:



```
<terminated> nestedFor1 [Java Application]
*****
*****
*****
*****
*****
```

Gambar 2.6

Saat $i=1$ (Baris 1): *Inner loop* berjalan 5 kali, mencetak `*****`. `println()` dieksekusi, kursor pindah ke baris 2. Sedangkan saat $i=2$ (Baris 2): *Inner loop* berjalan 5 kali lagi, mencetak `*****`. `println()` dieksekusi, kursor pindah ke baris 3. Proses ini akan berulang hingga $i=5$. Hasilnya adalah sebuah persegi bintang berukuran 5×5 .

2.2.7 Program Perulangan Bersarang (nestedFor2)

```

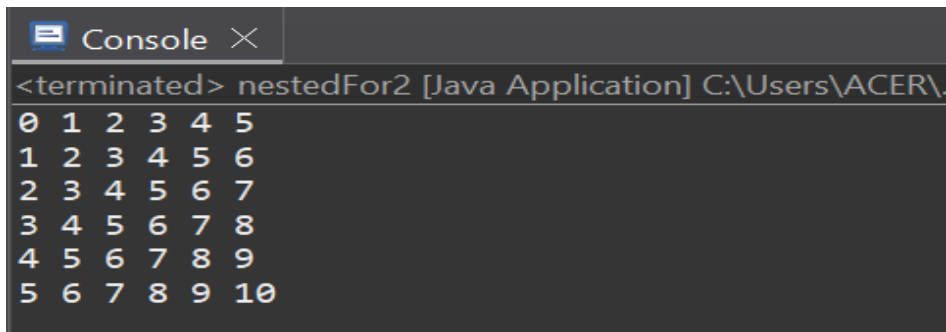
1      package pekan5;
2
3      public class nestedFor2 {
4
5          public static void main(String[] args) {
6              for (int i = 0; i <= 5; i++) {
7                  for (int j = 0; j <= 5; j++) {
8                      System.out.print(i+j+ " ");
9                  }
10                 System.out.println();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

Kode Program 2.7

Struktur dan Langkah Program:

1. `for (int i = 0; i <= 5; i++) { ... }`: *Outer loop* (baris) berjalan 6 kali (indeks 0 hingga 5).
2. `for (int j = 0; j <= 5; j++) { ... }`: *Inner loop* (kolom) berjalan 6 kali (indeks 0 hingga 5) untuk setiap iterasi *outer loop*.
3. `System.out.print(i + j + " ");`: Di dalam *inner loop*, program mencetak hasil penjumlahan antara indeks baris (i) dan indeks kolom (j), diikuti spasi.
4. `System.out.println();`: Pindah baris setelah 6 kolom selesai dicetak.

Output:



```
<terminated> nestedFor2 [Java Application] C:\Users\ACER\
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

Program ini menghasilkan sebuah tabel berukuran 6x6. Setiap sel dalam tabel berisi hasil penjumlahan dari indeks baris dan kolomnya, menciptakan pola tabel penjumlahan.

BAB III

PENUTUP

3.1 Kesimpulan

Berdasarkan analisis praktikum, dapat disimpulkan bahwa perulangan `for` adalah struktur kontrol yang penting di Java yang sangat efektif untuk eksekusi kode berulang (iterasi) saat jumlah perulangan telah diketahui. Keberhasilannya didukung oleh sintaksis tiga komponen utamanya: inisialisasi, kondisi, dan iterasi.

Implementasi `for` juga terbukti serbaguna. Perulangan ini dapat digabungkan dengan variabel akumulator (seperti pada `PerulanganFor3.java`) dan dapat beroperasi berdasarkan masukan dari pengguna melalui *class* `Scanner` (seperti pada `PerulanganFor4.java`). Selain itu, teknik perulangan bersarang (*nested loops*) terbukti sangat berguna untuk memproses data dua dimensi atau menghasilkan keluaran berbentuk pola, (seperti yang ditunjukkan dalam program `nestedFor0.java`, `nestedFor1.java`, dan `nestedFor2.java`).

3.2 Saran

Untuk pengembangan pemahaman lebih lanjut, disarankan untuk mengeksplorasi variasi lain dari perulangan `for` dan memperdalam pemahaman mengenai perulangan bersarang (*nested loops*) dengan mencoba mengimplementasikan studi kasus yang lebih kompleks, seperti membuat pola belah ketupat atau melakukan operasi matriks.

Sebagai catatan penting, dalam penulisan perulangan apa pun, sangat krusial untuk selalu memperhatikan kondisi berhenti (*termination condition*). Memastikan perulangan memiliki kondisi berhenti yang tepat adalah hal esensial untuk menghindari terjadinya *infinite loop* atau perulangan tak terhingga, yang dapat menyebabkan program berhenti bekerja (*crash*) atau tidak merespons.

DAFTAR PUSTAKA

- [1] W3Schools, "Java For Loop," [Daring]. Tersedia pada:
https://www.w3schools.com/java/java_for_loop.asp. [Diakses: 29-Okt-2025].
- [2] B. Hartono, *Pemrograman Java untuk Pemula*. Semarang: Yayasan Prima Agus Teknik, 2022. [Daring]. Tersedia pada:
<https://penerbit.stekom.ac.id/index.php/yayasanpat/article/view/351>. [Diakses: 29-Okt-2025].
- [3] Scribd, "Laporan Praktikum Algoritma Pemrograman," [Daring]. Tersedia pada: <https://www.scribd.com/document/548804796/LAPORAN-PRAKTIKUM-ALGORITMA-PEMROGRAMAN>. [Diakses: 30-Okt-2025].
- [4] CollegeSidekick, "Study Docs 24558962," [Daring]. Tersedia pada:
<https://www.collegesidekick.com/study-docs/24558962>. [Diakses: 30-Okt-2025].
- [5] Studocu, "Laporan Praktikum Pengulangan Pada Bahasa Java," [Daring]. Tersedia pada: <https://www.studocu.id/id/document/universitas-siliwangi/praktikum-pemrograman-berorientasi-objek/laporan-praktikum-pengulangan-pada-bahasa-java/43138256>. [Diakses: 30-Okt-2025].