LAPORAN PRAKTIKUM PEMOGRAMAN ALGORITMA DAN PEMOGRAMAN

"PERULANGAN FOR DALAM BAHASA PEMROGRAMAN JAVA"

Disusun Oleh :

Dzhillan Dzhalila

2511531001

Dosen Pengampu:

DR. Wahyudi, S.T, M.T

Asisten Praktikum:

Aufan Taufiqurrahman

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji syukur penulis panjatkan atas kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmat dan berkatnya, sehingga penulis dapat menyelesaikan laporan praktikum Algoritma Pemrograman dan Pemrograman dengan baik. Laporan ini disusun sebagai dokumentasi dan refleksi atas praktikum yang telah dilaksanakan pada pertemuan minggu ke-5 untuk mata kuliah Praktikum Algoritma Pemrograman. Praktikum ini berfokus pada pemahaman konsep Perulangan For (*For Loop)* yang ada pada bahasa pemrograman Java.

Melalui praktikum ini, output yang diharapkan adalah penulis dapat memahami konsep konsep dasar penggunaan Pengulanagn For dalam bahasa Java. Dalam Praktikum ini juga, penulis dapat meimplementasikan langsung Perulangan For untuk projek-projek sederhana, sehingga dapat membantu penulis memahami penggunaan Perulangan For yang benar dalam penulisan kode program.

Penulis menyadari bahawa laporan praktikum ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan dan pengembangan di masa mendatang.

Semoga laporan ini dapat memberi manfaat, baik sebagai bahan pembelajaran bagi penulis maupun referensi bagi pembaca dalam memahami dasar-dasar pemrograman komputer.

Padang, 2025

Dzhillan Dzhalila

251531001

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam pemrograman dengan bahasa apapun, kita mungkin sering kali ingin menduplikasi suatu blok kode agar output yang dihasilkan adalah kode berulang dari kode blok kode tersebut. Kita termasuk penulis sendiri mungkin bisa saja mengambil langkah untuk menulis blok kode tersebut satu persatu. Namun, seperti yang kita ketahui, langkah tersebut akan memakan waktu yang cukup lama dan melelahkan. Apalagi batas perulangan yang kita inginkan bukan dalam jumlah yang sedikit, seperti kita menginginkan seratus kali perulangan bahkan seribu kali perulangan. Untuk itu, kita dapat menggunakan pernyataan alur kontrol yang tersedia dalam bahasa pemrograman yaitu ada *For Loop* dan atau *nested for*.

Perulangan For dapat mencegah ketidakefektifan yang ditimbulkan saat kita menulis manual syntak satu persatu. Penggunaan *For Loop* memungkinkan kode dieksekusi berulang kali berdasarkan kode boolean tertentu. *For loop* digunakan ketika jumlah literasi sudah diketahui sebelup *loop* dijalankan. Adapun *nested loop for* juga digunakan dalam bahasa pemrograman Java. *Nested loop for* memungkinkan perulangan di dalam perulangan. Pemahaman dari Perulangan For inilah yang akan menjadi harapan output dari terlaksananya praktikum ini.

1.2 Tujuan

1. Memahami konsep dasar perulangan For/ *for looping* dalam bahasa pemrograman Java.
2. Mampu meingmplementasikan perulangan For dalam pemrograman mulai dari projek sederhana hingga pada projek dengan level yang lebih tinggi.
3. Membuat suatu projek pemrograman dengan lebih ekfektif dan efisien dengan penggunaan perulangan For.

4. Mampu mengembangkan pemahaman terhadap alur perulangan For dalam kondisi tertentu.

1.3 Manfaat

1. Memberikan pengalaman kepada mahasiswa dalam menulis program dengan menggunakan *for looping.*
2. Memotivasi para mahasiswa untuk menambah pengetahuan dan pemahaman terhadap jenis-jenis perulangan lainnya.

BAB II

PEMBAHASAN

2.1 Perulangan For

Perulangan For atau biasanya juga disebut *For loop* adalah pernyataan alur kontrol dalam kode program. Perulangan For merupakan salah satu contoh jenis perulangan yang memiliki jumlah pengulangan yang tentu dan terhitung (*counted loop).* Dalam struktur perulangan ini, akan ada aksi yang dieksekusi berulang kali selama suatu kondisi.

Adapun struktur dasar dari perulangan ini adalah sebagai berikut;

```
for (Inisialisasi; kondisi; decreament/increament) {
// blok kode yang akan diulang }
```

a. Inisialisasi adalah pernyataan yang dijalankan hanya satu kali di awal perulangan. Biasanya, inisialisasi digunakan untuk memberikan nilai awal pada variabel penghitung. Contohnya: i=0
b. Kondisi adalah pernyataan yang akan terus dievaluasi setiap literasi. Selama kondisi ini bernilai benar/*true,* perulangan akan terus berlanjut. Namun, jika kondisi sudah bernilai salah/*false,* maka perulangan akan berhenti. Contoh adalah i < 5.
c. Perubahan adalah pernyataan yang akan diekseskusi setiap literasi telah selesai. Biasanya untuk mengubah nilai variabel penghitung. Perubahan pada perulangan For terbagi atas 2, yaitu;
    - Increment (++) : menambah nilai variabel sebesar 1.
    - Decrement (--) : mengurangi nilai variabel sebesar 1.

2.1.1 Perulangan For 1

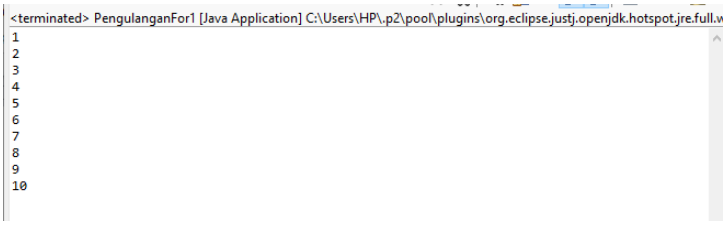
2.1.1.1 Kode Program

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

Kode Program 2.1: Kode Program For Sederhana

2.1.1.2 Penjelasan

Dalam program Perulangan For 1 dapat dilihat bahwa variabel **i** dideklarasikan dan dinisialisasikan dengan nilai 1. Ini adalah nilai awal perhitungan. Kondisi *loop* pada program ini adalah i kurang dari atau sama dengan 10. Selama kondisi tersebut dipenuhi maka blok dalam loop akan terus dijalankan. Jika kondisi tidak terpenuhi atau bernilai salah/*false* maka *loop* akan terhenti. Misal nilai i = 12, itu berarti kondisi sudah tidak terpenuhi, maka *loop* akan berhenti dan lanjut pada kepada perubahan. Dalam pemrograman ini, perubahan yang digunakan adalah *incresment* yaitu i = i + 1. Lalu output yang dihasilkan oleh program ini adalah sebagai berikut;



Gambar 2.1 : Hasil Ouput dari Program Perulangan For 1

2.1.2 Perulangan For 2

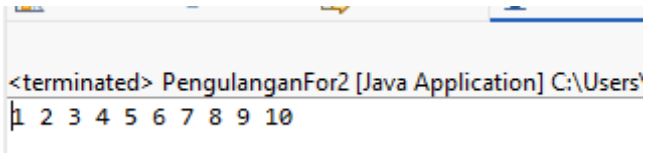
2.1.2.1 Kode Program

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

Kode Program 2.2 : Kode Program Perulangan For Tanpa "Println"

2.1.2.2 Penjelasan

Dalam program ini, kita tidak menggunakan System.out.println. Hal ini menyebabkan adanya perubahan pada output yang akan ditampilkan secara horizontal karena tidak ada perintah "println" yang membuat outputnya berada pada *line* yang berbeda. Selain itu, inisialisasi, kondisi, serta perubahan program ini masih sama dengan program sebelumnya. Adapun output yang dihasilkan ada sebagai berikut;



Gambar 2.1 : Hasil Output dari Program Perulangan For 2

2.1.3 Perulangan For 3

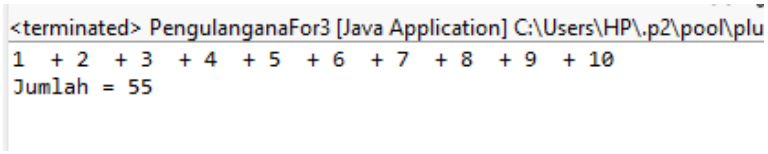
2.1.3.1 Kode Program

```
      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 : Kode Program Perulangan For dengan Operasi Aritmatika

2.1.3.2 Penjelasan

Dalam program ini terdapat operasi arimatika yaitu penjumlahan dari seluruh bilangan dari perulangan. Dari pemrograman tersebut kita dapat melihat variabel yang digunakan ada variabel “jumlah” dan juga “i”. Variabel “jumlah” di set pada nilai “0”, sementara variabel “i” memiliki nilai awal 1. Dan kondisi yang digunakan masih tetap sama dengan program sebelumnya dan masih menggunakan increament. Program ini juga tidak menggunakan perintah “Println” dan menggunakan tambahan ouput tanda operasi penambahan yaitu “+”. Adapun output yang dihasilkan oleh program ini adalah;



Gambar 2.3 : Hasil Output dari Program Perulangan For 3

2.1.4 Perulangan For 4

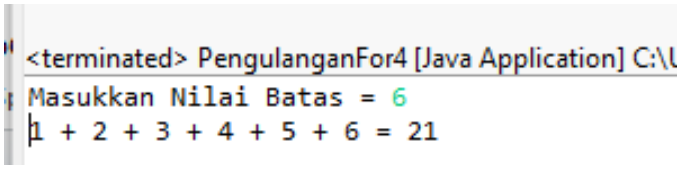
2.1.4.1 Kode Program

```
        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 : Kode Program For dengan Inputan User

2.1.4.2 Penjelasan

Tujuan dari program ini adalah menjumlahkan semua bilangan sampai batas bilangan yang diinputkan oleh *user* itu sendiri. Ada tambahan variabel yaitu variabel batas yang diinputkan olehpengguna. Kondisi yang digunakan adalah i<= batas, yang berarti kondisi perulangan akan terus dilakukan selama nilai variabel "i" masih lebih kecil atau sama dengan nilai batas inputan user. Lalu, jika kondisi bernilai salah maka perulangan akan berhenti secara otomatis. Dalam program ini juga menggunakan kondisi *if* di dalam perulangan for itu sendiri. Hal ini memungkinkan program dapat berhenti jika nilai dari variabel "i" sudah tidak sesuai dengan kondisi perulangan for. Setelah itu, akan ada output nilai dari penjumlahan semua bilangan dengan bilangan batas yang dimasukkan oleh *user* tadi. Adapun hasil output dari program ini adalah;



Gambar 2.4 : Hasil Ouput dari Program Perulangan For 4

Pada gambar 2.4 , batas yang dimasukkan adalah 6 dan ouput yang dihasilkan adalah penjumlahan dari angka 1 hingga angka 6.

2.2 Nested For

Nested For adalah perulangan For yang berada di dalam perulangan For lainnya. Perulangan luar akan mengulang beberapa kali, dan di setiap literasi dari perulangan luar, perulangan dalam juga akan diulang.

2.2.1 Nested For 1

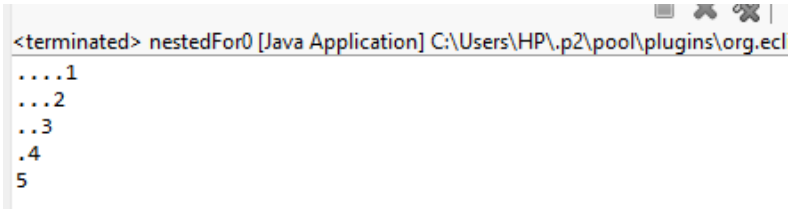
2.2.1.1 Kode Program

```
        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 Nested For Decreasment

2.2.1.2 Penjelasan

Program di atas mengharapkan output yang memiliki komponen angka (1-5) dan juga titik (.). Dalam program ini kita menggunakan 2 sistem perulangan For. Pada pengulangan luar kita menggunakan dan mendeklrasikan variabel bernama “line”. Nilai awal variabel “line” adalah 1 dan kondisi yang diberikan untuk pengulangan luar ini adalah line <= , yang berarti line yang ada hanya ada 5 line. Dan pengulangan dalam menggunakan j dan mendeklarasikan nilai awalnya yaitu 1. Dan untuk kondisi dari perulangan dalam yaitu j <= (-1* line +5) dan akan dihasilkan output berupa (.). Hasil output dari program di atas adalah sebagai berikut;



Gambar 2.5 : Hasil Output Program Nested For 1

Dari gambar 2.5 , kita dapat melihat ada 2 kompenen iutout yaitu berupa angka dan juga titik. Angka dihasilkan dari perulangan for luar, sementara titik dihasilkan dari perulangan

2.2.2 Nested For 2

2.2.2.1 Kode Program

```
for (int i = 0; i <=100; i++) {
for ( int j = 1; j<=10; j++) {
       System.out.print("*");
}
System.out.println(); }}
```

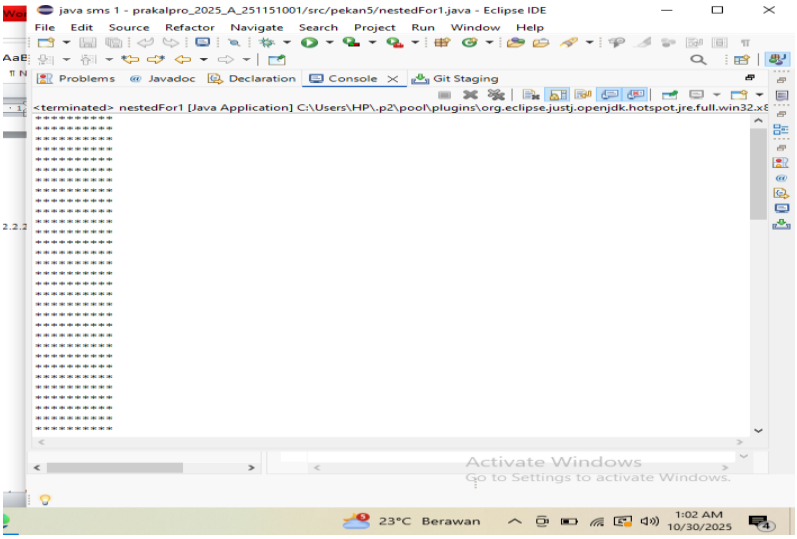
Kode Program 2.6 : Kode Program Nested For 2

2.2.2.2 Penjelasan

Program di atas dirancang untuk menghasilkan pola berbentuk kotak yang terdiri dari karakter asterisk (*). Pola ini dibentuk menggunakan dua perulangan for bersarang (nested loop). Pada perulangan luar, dideklarasikan variabel i dengan nilai awal 0, dan perulangan akan terus berjalan selama kondisi i <= 100 terpenuhi. Artinya, perulangan luar akan dijalankan sebanyak 101 kali (dari i = 0 hingga i = 100), sehingga program akan menghasilkan 101 baris output. Di dalam setiap iterasi perulangan luar, terdapat perulangan dalam yang menggunakan variabel j. Variabel j diinisialisasi dengan nilai 1, dan perulangan dalam akan berlangsung selama j <= 10. Dengan demikian, pada setiap baris, program akan mencetak karakter * sebanyak 10 kali secara berurutan tanpa pindah baris (karena menggunakan System.out.print).

Setelah 10 karakter * selesai dicetak pada suatu baris, perintah System.out.println(); dieksekusi untuk memindahkan kursor ke baris berikutnya, sehingga baris berikutnya dapat dicetak di bawahnya.

Adapun output dari program ini adalah;



Gambar 2.6 : Hasil Output Program Nested For 2

2.2.3 Netsed For 3

2.2.3.1 Kode Program

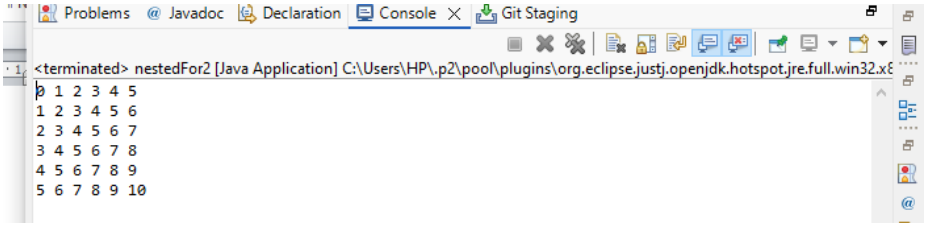
```
public static void main(String[] args) {
for (int i = 0; i<= 5; i++) {
      for (int j = 0; j<= 5; j++) {
            System.out.print(i+j +" ");
      }
      System.out.println();
}
```

Kode Program 2.7 : Kode Program Nested For 3

2.2.3.2 Penjelasan

Program di atas dirancang untuk menghasilkan pola angka berbentuk matriks berukuran 6 × 6, di mana setiap elemen pada matriks merupakan jumlah dari indeks baris (i) dan indeks kolom (j). Pola ini dibentuk menggunakan dua perulangan for bersarang (nested loop).Pada perulangan luar, dideklarasikan variabel i dengan nilai awal 0, dan perulangan akan berlangsung selama kondisi i <= 5 terpenuhi. Artinya, program akan menghasilkan 6 baris output (karena mencakup nilai i = 0, 1, 2, 3, 4, 5).

Di dalam setiap iterasi perulangan luar, terdapat perulangan dalam yang menggunakan variabel j, yang juga dimulai dari 0 dan berakhir pada 5 (j <= 5). Dengan demikian, pada setiap baris, program akan melakukan 6 kali pencetakan, masing-masing berisi hasil penjumlahan i + j. Setelah setiap baris selesai dicetak, perintah System.out.println(); digunakan untuk memindahkan kursor ke baris berikutnya, sehingga pola tetap terbentuk dalam susunan baris dan kolom yang rapi. Hasil output dari program ini adalah sebagai berikut;



Gambar 2.7 : Hasil Ouput Program Nested For 3

BAB III

KESIMPULAN

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa perulangan for merupakan struktur kendali yang sangat efektif untuk mengeksekusi blok kode secara berulang dengan jumlah iterasi yang telah ditentukan. Penggunaan for memungkinkan penulisan kode yang lebih ringkas, terstruktur, dan mudah dipahami, terutama ketika jumlah perulangan bersifat tetap atau dapat diprediksi.

Selain itu, konsep nested for (perulangan bersarang) terbukti sangat berguna dalam menangani masalah yang melibatkan dimensi ganda, seperti pembentukan pola teks (misalnya segitiga, kotak, atau susunan rata kanan), operasi matriks, serta simulasi struktur data dua dimensi. Dalam nested for, perulangan luar umumnya mengatur baris, sedangkan perulangan dalam mengatur kolom, sehingga kombinasi keduanya mampu menghasilkan output yang terorganisasi secara sistematis.

Melalui berbagai contoh program—mulai dari pencetakan angka berurutan, penjumlahan deret, hingga pembentukan pola dengan titik dan bintang—terlihat jelas bahwa pemahaman terhadap inisialisasi, kondisi, dan inkremen/dekremen dalam perulangan sangat menentukan keberhasilan dalam menghasilkan output yang diinginkan.

Dengan demikian, penguasaan terhadap struktur perulangan for dan nested for merupakan Pondasi penting dalam pemrograman, karena tidak hanya meningkatkan efisiensi penulisan kode, tetapi juga membangun kemampuan logika algoritmA yang diperlukan untuk menyelesaikan berbagai permasalahan komputasi yang lebih kompleks di masa depan.

DAFTAR PUSTAKA

[1] Dicoding, “Apa itu Looping? Kenali Jenis-Jenisnya,” 2025. [Daring]. Tersedia pada : *https://www.dicoding.com/blog/apa-itu-looping-kenali-jenis-jenisnya/#:~:text=Dengan%20looping%2C%20programmer%20dapat%20mengeksekusi,tetap%20rapi%20dan%20mudah%20dipelihara*. [Diakses : 29-Okt-2025].

[2] TheKnowledgeacademy, “Java For Loop : Syntax, Examples, and Types,” 2025. [Daring]. Tersedia pada : https://www.theknowledgeacademy.com/blog/java-for-loop/#:~:text=Perulangan%20for%20dalam%20Java%20adalah,jika%20suatu%20kondisi%20bernilai%20benar. https://www.petanikode.com/java-perulangan/ [Diakses : 29-Okt-2025].

[3] Itbox, “ Belajar Java Looping :Menguasai Jenis Perulangan dalam Java,”. 2024. [Daring]. Tersedia pada : https://itbox.id/blog/belajar-java-looping . [Diakses : 29-Okt-2025].