

LAPORAN PRAKTIKUM
PEMROGRAMAN ALGORITMA DAN PEMROGRAMAN
PERULANGAN DAN NESTED FOR

Disusun Oleh:

Muhammad Fharel

2511531010

Dosen Pengampu:

Dr. Wahyudi S.T.M.T

Asisten Pratikum:

Muhammad Zaki Al Hafiz



DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Segala puji penulis panjatkan ke hadirat Allah SWT atas rahmat dan karunia-Nya sehingga laporan praktikum Algoritma dan Pemograman pada tanggal 29 Oktober 2025 yang membahas tentang bahasa pemograman java spesifiknya pada bagian perulangan *for* dan *nested for*, dan bagaimana penerapannya untuk membuat system yang sesuai dengan kebutuhan. Materi ini penting karena menjadi fondasi dalam memahami pemograman.

Ucapan terima kasih ditujukan kepada dosen pengampu, asisten praktikum, serta rekan-rekan yang telah membantu dalam proses pelaksanaan praktikum. Penulis menyadari bahwa penulisan laporan masih jauh dari kata sempurna, oleh karena itu kritik dan saran sangat diharapkan untuk penyempurnaan di kemudian hari. Semoga laporan ini memberikan manfaat dan menambah wawasan pembaca.

Padang, 1 November 2025

Penulis

DAFTAR ISI

| | |
|--|-----------|
| KATA PENGANTAR | i |
| DAFTAR ISI | ii |
| BAB I PENDAHULUAN | 1 |
| 1.1 Latar Belakang..... | 1 |
| 1.2 Tujuan Praktikum | 1 |
| 1.3 Manfaat Praktikum | 2 |
| BAB II PEMBAHASAN | 3 |
| 2.1 Pengertian dan Penjelasan Perulangan For dan Nested For | 3 |
| 2.2 Pembuatan Package dan Class Pekan 5 | 4 |
| 2.3 Program Pertama (Perulangan For 1) | 6 |
| 2.3.1 Output..... | 7 |
| 2.3.2 Penjelasan Singkat | 7 |
| 2.4 Program Kedua (Perulangan For 2)..... | 7 |
| 2.4.1 Output..... | 8 |
| 2.4.2 Penjelasan Singkat | 8 |
| 2.5 Program Ketiga (Perulangan For 3) | 9 |
| 2.5.1 Output..... | 9 |
| 2.5.2 Penjelasan Singkat | 10 |
| 2.6 Program Keempat (Perulangan For 4)..... | 10 |
| 2.6.1 Output..... | 11 |
| 2.6.2 Penjelasan Singkat | 11 |
| 2.7 Program Kelima (Nested For 0) | 12 |
| 2.7.1 Output..... | 12 |
| 2.7.2 Penjelasan Singkat | 13 |

| | | |
|------------------------------|--------------------------------------|-----------|
| 2.8 | Program Kelima (Nested For 0) | 13 |
| 2.8.1 | Output..... | 13 |
| 2.8.2 | Penjelasan Singkat | 14 |
| 2.9 | Program Ketujuh (Nested For 2) | 14 |
| 2.9.1 | Output..... | 14 |
| 2.9.2 | Penjelasan Singkat | 15 |
| BAB III PENUTUP | | 16 |
| 3.1 | Kesimpulan..... | 16 |
| DAFTAR PUSTAKA | | 17 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pemrograman, perulangan (*looping*) merupakan konsep dasar yang sangat penting karena memungkinkan suatu program menjalankan instruksi yang sama berulang kali tanpa harus menuliskannya secara berulang. Salah satu bentuk perulangan yang umum digunakan adalah perulangan *for*. Struktur ini membantu programmer mengeksekusi blok kode dalam jumlah yang telah ditentukan, sehingga meningkatkan efisiensi dan keterbacaan program.

Selain itu, dalam beberapa kasus diperlukan perulangan bersarang (*nested loop*), yaitu perulangan di dalam perulangan lain. Teknik ini sering digunakan dalam pengolahan data berbentuk tabel, matriks, atau pola tertentu seperti pembuatan bentuk bintang di layar. Pemahaman terhadap kedua jenis perulangan ini menjadi hal mendasar bagi mahasiswa yang mempelajari bahasa pemrograman, terutama Java, karena hampir semua algoritma kompleks melibatkan konsep looping.

Praktikum ini bertujuan untuk melatih kemampuan mahasiswa dalam menggunakan struktur *for* secara tepat dan efektif. Melalui praktikum ini, mahasiswa diharapkan mampu memahami sintaks dasar, logika pengulangan, serta penerapan *for* dalam berbagai kasus pemrograman. Pemahaman terhadap konsep ini menjadi fondasi penting dalam pengembangan aplikasi yang lebih kompleks di masa mendatang.

1.2 Tujuan Praktikum

Tujuan dari praktikum ini adalah:

1. Memahami konsep dasar perulangan *for* dalam bahasa pemrograman Java.
2. Mengetahui cara kerja dan penerapan *nested for* untuk berbagai kasus.
3. Melatih kemampuan mahasiswa dalam menulis kode yang efisien dan logis.

1.3 Manfaat Praktikum

Manfaat dari praktikum ini adalah:

1. Meningkatkan kemampuan logika pemrograman dan analisis algoritma.
2. Memberikan pemahaman praktis tentang struktur perulangan dalam Java.
3. Membantu mahasiswa dalam menyusun program yang efisien dan terstruktur.

BAB II

PEMBAHASAN

2.1 Pengertian dan Penjelasan Perulangan For dan Nested For

a. Perulangan For

Perulangan *for* adalah salah satu bentuk perulangan yang digunakan ketika jumlah iterasi sudah diketahui. Struktur umumnya seperti berikut:

```
for (inisialisasi; kondisi; increment/decrement) {  
    // blok kode yang diulang  
}
```

Contohnya:

```
for (int i = 1; i <= 5; i++) {  
    System.out.println("Perulangan ke-" + i);  
}
```

Kode di atas akan menampilkan teks “Perulangan ke-1” hingga “Perulangan ke-5”.

Komponen utamanya terdiri dari:

- 1) *Inisialisasi*: Menentukan nilai awal variabel kontrol (misalnya `int i = 1`).
- 2) *Kondisi*: Mengevaluasi apakah perulangan akan terus berjalan (`i <= 5`).
- 3) *Increment/Decrement*: Mengubah nilai variabel kontrol setiap iterasi (`i++`).

Struktur ini banyak digunakan karena sintaksnya sederhana dan mudah dipahami.

b. Nested For

Nested for berarti satu perulangan ditempatkan di dalam perulangan lain. Biasanya digunakan untuk membuat pola dua dimensi atau mengakses data dalam struktur matriks. Contohnya seperti berikut:

```

for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        System.out.print("* ");
    }
    System.out.println();
}

```

Dari contoh tersebut, maka akan menghasilkan output seperti berikut:

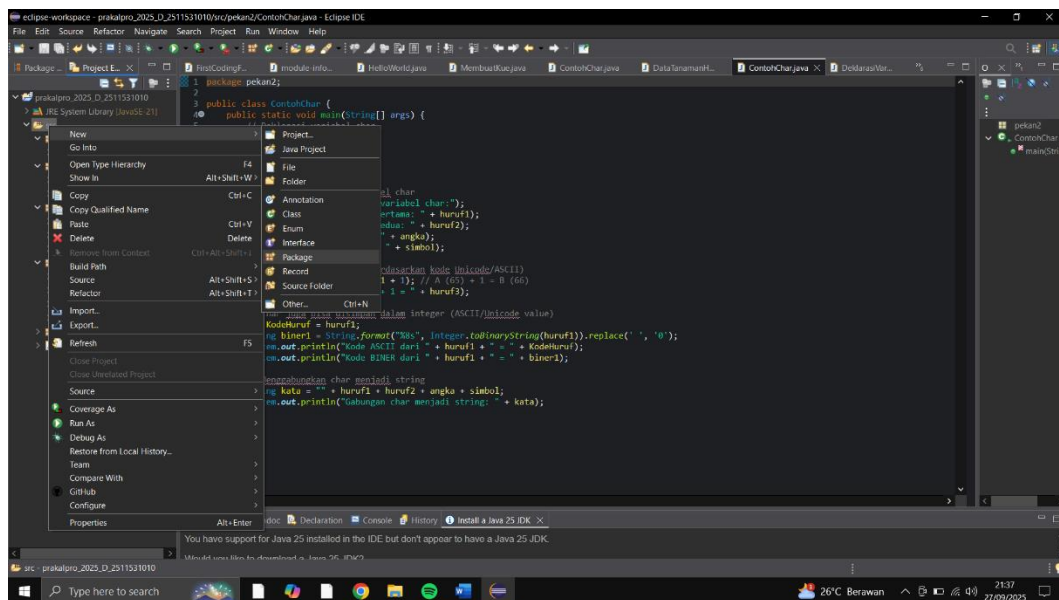
```

* * *
* * *
* * *

```

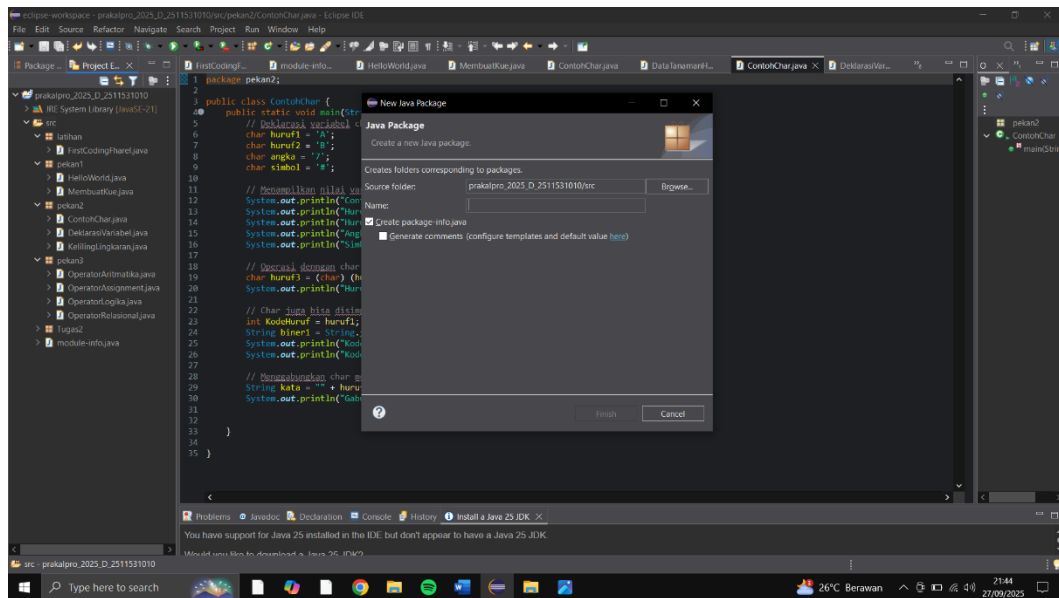
2.2 Pembuatan Package dan Class Pekan 5

1. Untuk membuat *package* baru, klik kanan pada src dan tekan “New” setelah itu pilih *package*.



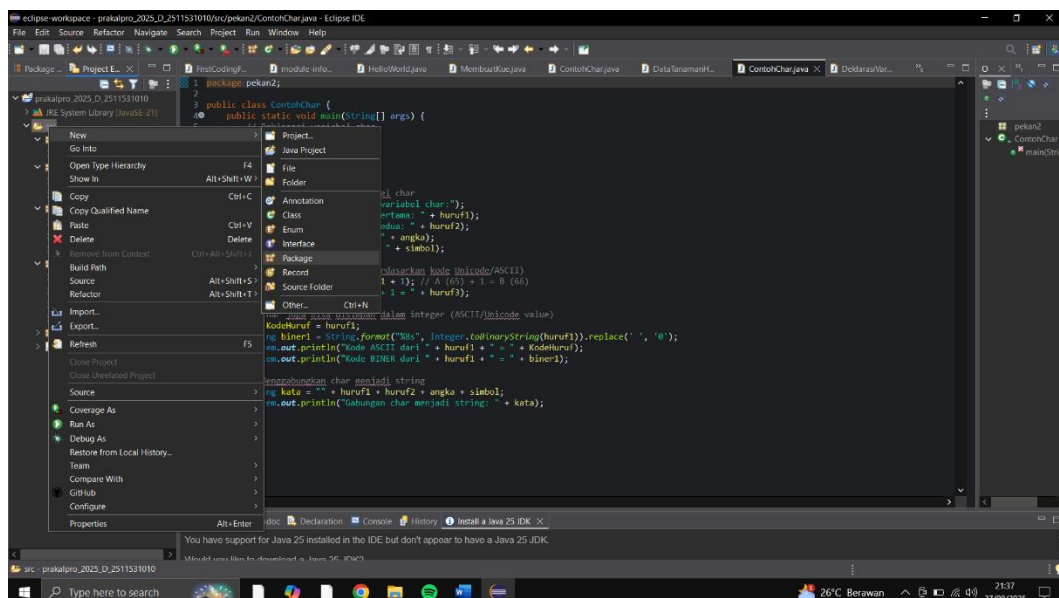
Gambar 2.1

2. Lanjut diberi nama *package* tanpa pakai spasi, huruf kapital ataupun karakter khusus. Seperti “pekan5” dan tekan *Finish*.



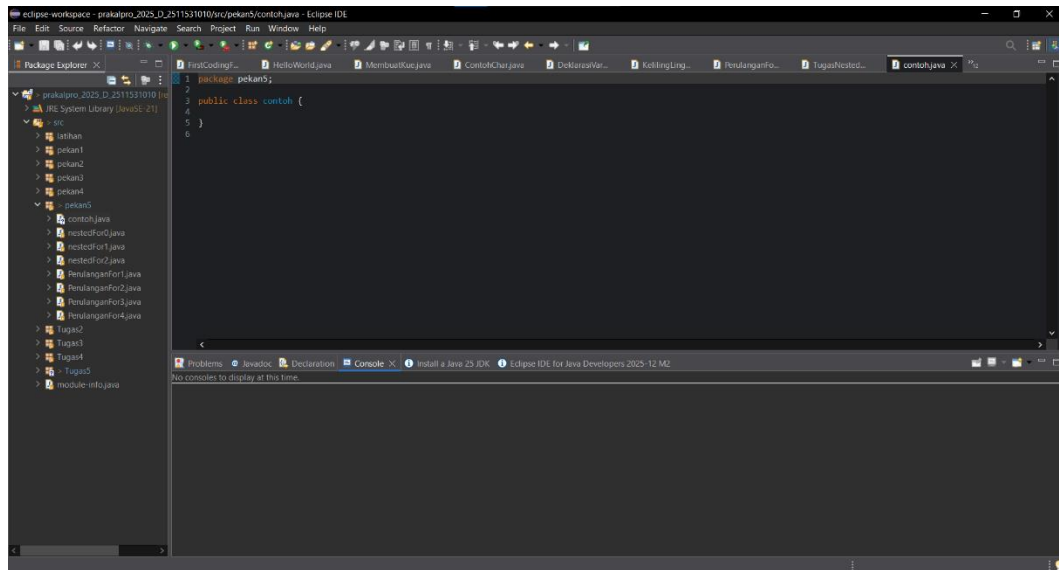
Gambar 2.2

3. Kemudian klik kanan pada package “pekan5” tersebut dan klik “New” dan pilih bagian “Class” untuk memulai membuat program.



Gambar 2.3

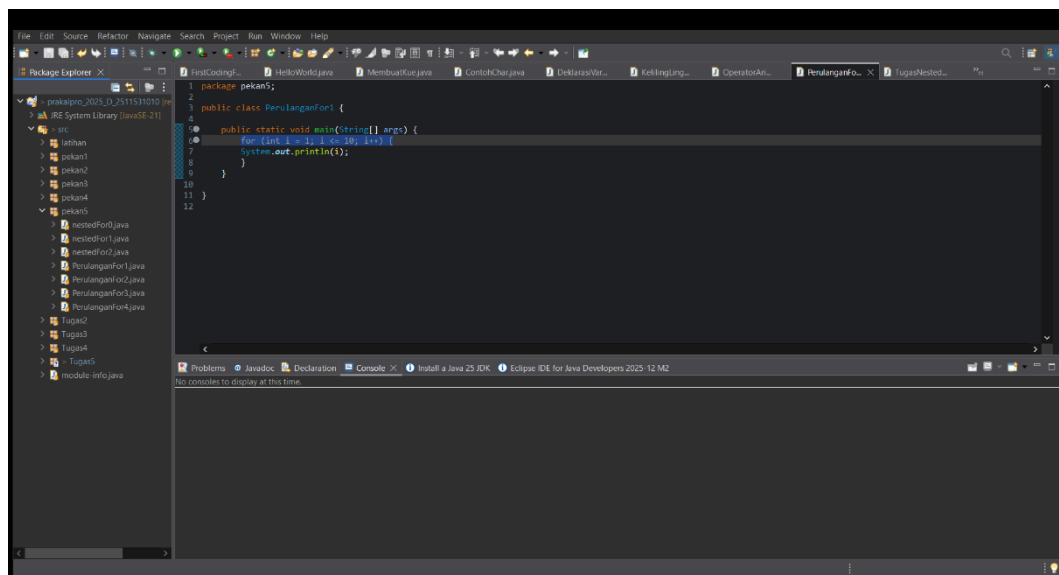
4. Buat nama *Class* yang akan dibuat pada bagian nama tanpa spasi dan menggunakan huruf kapital pada awal kata, lalu klik bagian “*public static void main(string[] args)*”. Kemudian klik *finish*.



Gambar 2.4

2.3 Program Pertama (Perulangan For 1)

Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut “PerulanganFor1”. Dan masukkan kode pemrograman berikut.

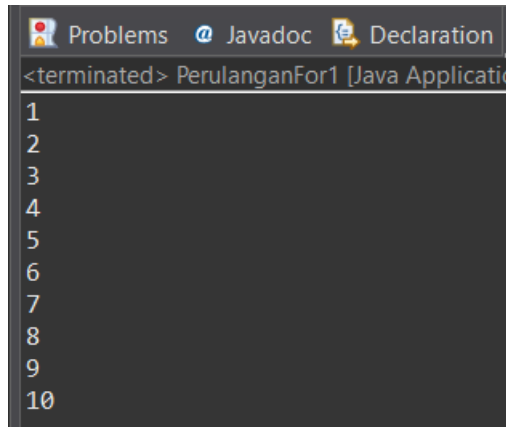


Gambar 2.5

2.3.1 Output

Program di atas menggunakan perulangan *for* yang dimulai dari $i = 1$ hingga $i = 10$. Pada setiap iterasi, nilai i akan dicetak menggunakan `System.out.println(i);` yang otomatis membuat baris baru setelah setiap angka.

Maka, *output* yang dihasilkan adalah:



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

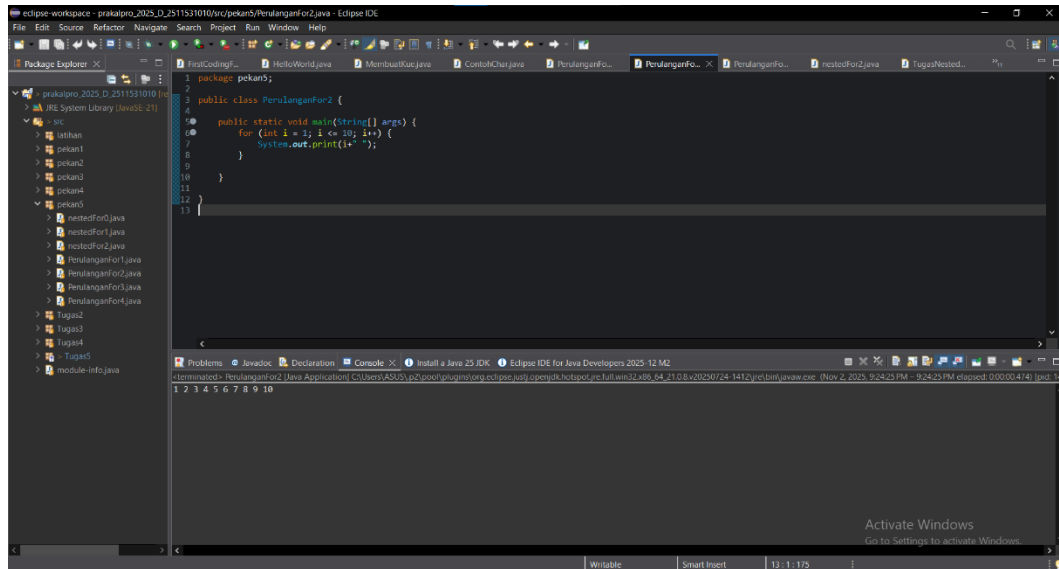
Gambar 2.6

2.3.2 Penjelasan Singkat

Perulangan *for* jenis ini berguna untuk menampilkan urutan angka secara sederhana dan sistematis. Kode ini juga sering digunakan dalam latihan awal pemrograman untuk memahami pola iterasi. Prinsip dasarnya adalah menentukan titik awal, kondisi batas, dan perubahan nilai tiap langkah. Program ini juga menunjukkan bagaimana komputer mengeksekusi instruksi berulang secara efisien tanpa penulisan kode berulang.

2.4 Program Kedua (Perulangan For 2)

Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut “PerulanganFor2”. Dan masukkan kode pemrograman berikut.



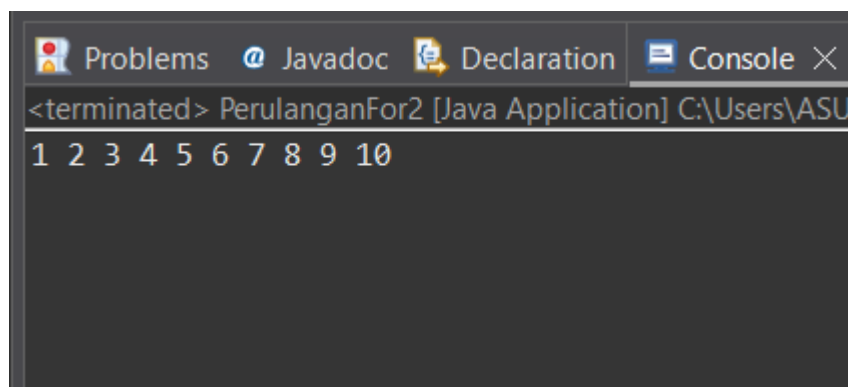
Gambar 2.7

2.4.1 Output

Perbedaannya dengan program sebelumnya terletak pada penggunaan *System.out.print* alih-alih *println*. Akibatnya, setiap nilai *i* yang dicetak tidak berpindah ke baris baru, melainkan disambung di baris yang sama dengan spasi di antaranya.

Program tetap melakukan iterasi sebanyak 10 kali (dari *i* = 1 hingga *i* = 10), tetapi hasil tampilannya berbentuk horizontal.

Maka, *output* yang dihasilkan adalah:



Gambar 2.8

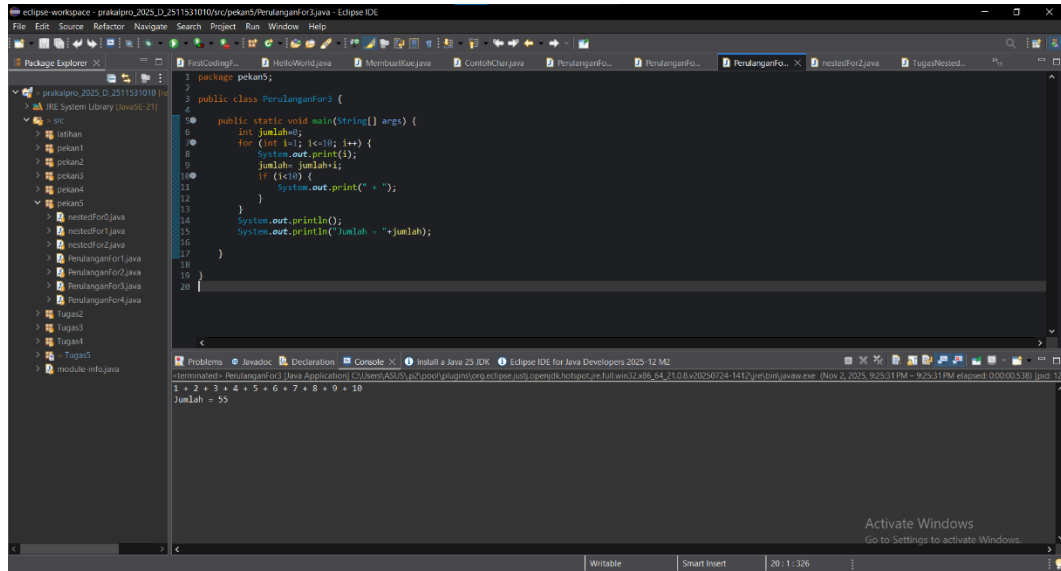
2.4.2 Penjelasan Singkat

Program ini menunjukkan bagaimana perbedaan kecil dalam fungsi *output* dapat menghasilkan tampilan yang berbeda. *print* menjaga kursor di baris yang sama, sedangkan *println* memindahkan ke baris baru. Struktur ini sering digunakan saat programmer ingin

menampilkan data dalam satu baris, seperti daftar angka atau hasil perhitungan singkat.

2.5 Program Ketiga (Perulangan For 3)

Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut “PerulanganFor3”. Dan masukkan kode pemrograman berikut.



```
1 package pekan3;
2
3 public class PerulanganFor3 {
4
5     public static void main(String[] args) {
6         int jumlah0;
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
20 }
```

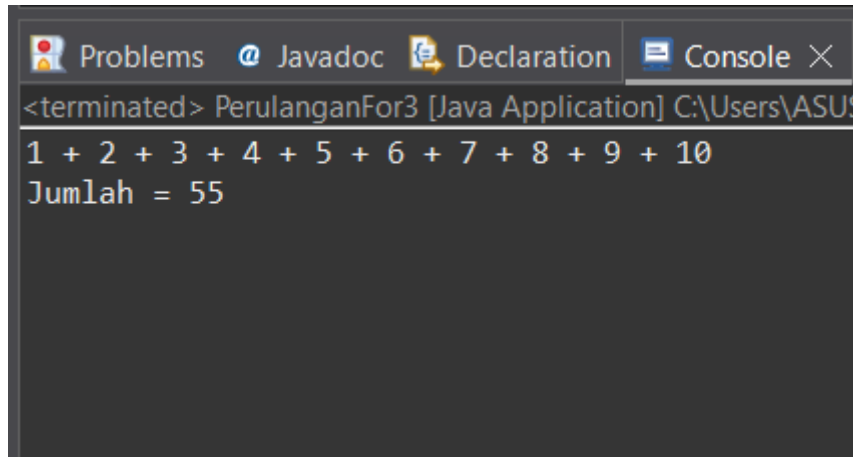
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
Jumlah = 55

Gambar 2.9

2.5.1 Output

Program ini menggunakan perulangan untuk menghitung total penjumlahan dari angka 1 sampai 10. Variabel jumlah bertugas menyimpan hasil akumulasi dari setiap nilai *i*. Selama perulangan berlangsung, angka akan ditampilkan di layar disertai tanda “+” di antaranya, kecuali pada angka terakhir. Setelah perulangan selesai, hasil penjumlahan seluruh angka akan ditampilkan di baris baru.

Maka, *output* yang dihasilkan adalah:



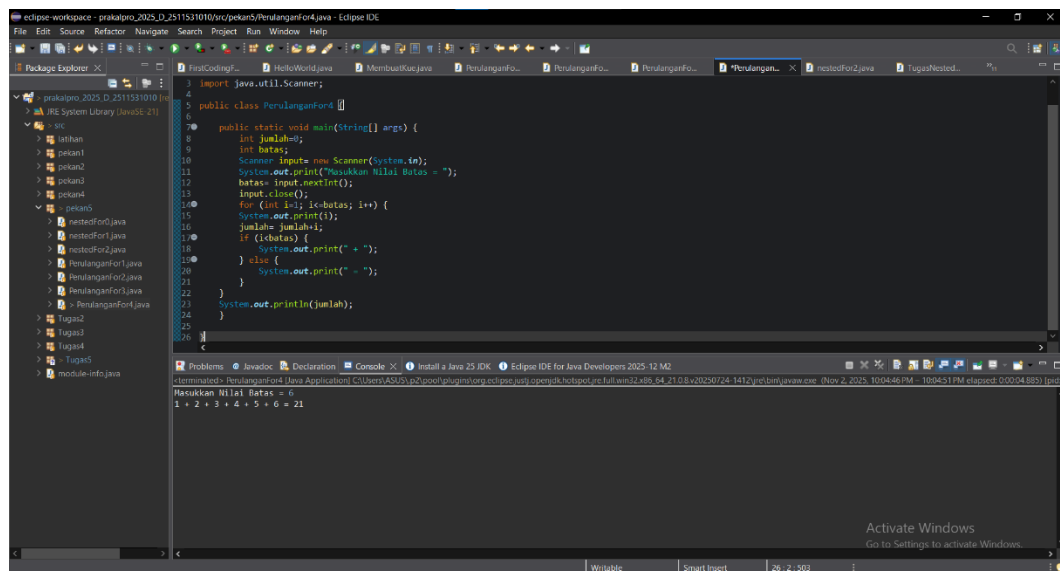
Gambar 2.10

2.5.2 Penjelasan Singkat

Program ini menjadi contoh penggunaan perulangan untuk perhitungan otomatis. Nilai jumlah berperan sebagai akumulator yang menampung total dari seluruh nilai i . Konsep ini banyak digunakan dalam logika pemrograman seperti perhitungan rata-rata, total harga, atau hasil evaluasi data. Logika kondisional *if* juga menunjukkan kontrol tambahan dalam menyesuaikan tampilan *output*.

2.6 Program Keempat (Perulangan For 4)

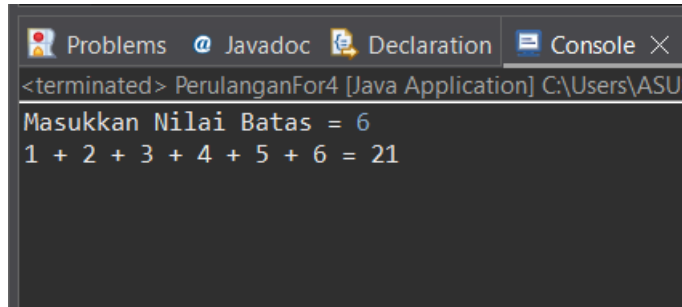
Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut “PerulanganFor4”. Dan masukkan kode pemrograman berikut.



Gambar 2.11

2.6.1 Output

Program ini menambahkan unsur interaksi dengan pengguna. Nilai batas (batas atas perulangan) tidak ditentukan di awal, tetapi dimasukkan oleh pengguna melalui *Scanner*. Misalnya, jika pengguna memasukkan angka 6. Maka, *output* yang dihasilkan adalah:



```
<terminated> PerulanganFor4 [Java Application] C:\Users\ASUS
Masukkan Nilai Batas = 6
1 + 2 + 3 + 4 + 5 + 6 = 21
```

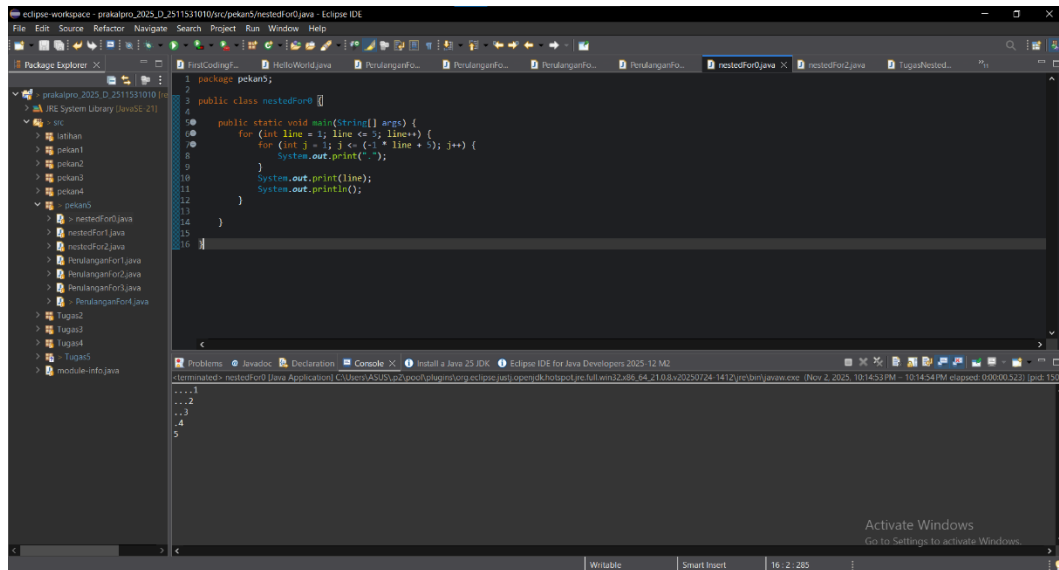
Gambar 2.12

2.6.2 Penjelasan Singkat

Perulangan ini memperlihatkan bagaimana input dinamis dapat memengaruhi jalannya program. Dengan menentukan batas perulangan melalui pengguna, program menjadi fleksibel dan interaktif. Logika *if-else* di dalam *loop* juga memperlihatkan bagaimana kondisi tertentu dapat mengubah tampilan *output*, menunjukkan kombinasi antara kontrol alur dan pengolahan data.

2.7 Program Kelima (Nested For 0)

Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut “nestedFor0”. Dan masukkan kode pemrograman berikut.



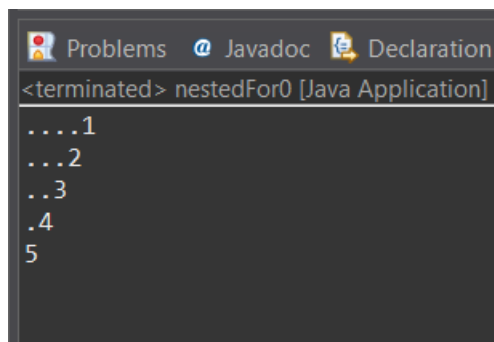
Gambar 2.13

2.7.1 Output

Program ini menggunakan *nested loop* (perulangan bersarang) untuk mencetak pola berupa titik (.) dan angka.

Pada setiap baris, jumlah titik akan berkurang satu, sedangkan angka yang dicetak meningkat. Perhitungan $(-1 * \text{line} + 5)$ membuat jumlah titik berkurang sesuai baris seberapa perulangan berada. Tampak bahwa baris pertama memiliki empat titik diikuti angka 1, lalu titik berkurang hingga hanya tersisa angka 5 di baris terakhir.

Maka, *output* yang dihasilkan adalah:



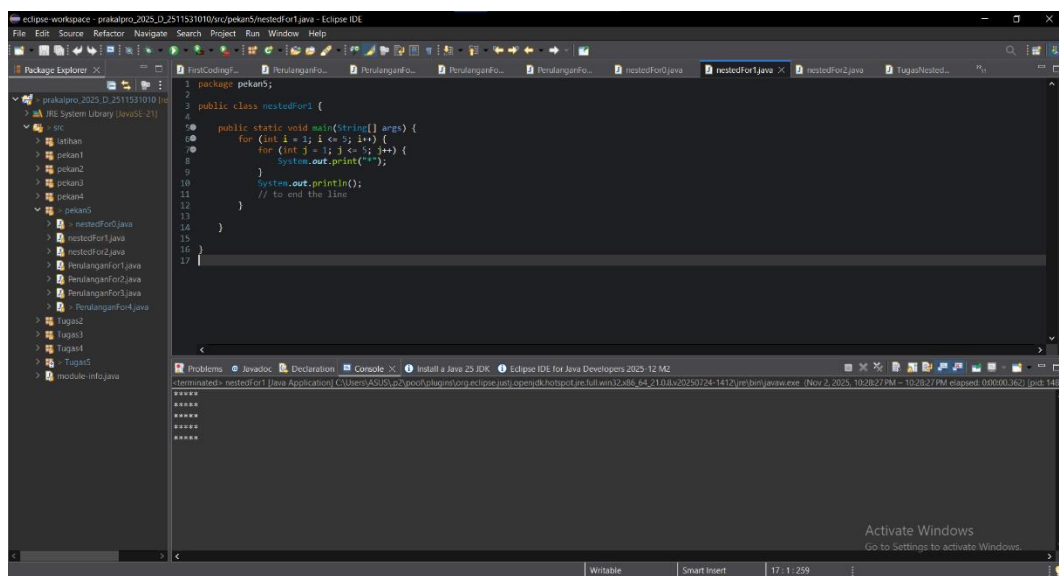
Gambar 2.14

2.7.2 Penjelasan Singkat

Program mengajarkan cara berpikir dua dimensi: satu perulangan mengatur baris (vertikal), sementara yang lain mengatur isi baris (horizontal). Konsep tersebut penting untuk memahami struktur pola, grafik, maupun manipulasi data berbentuk tabel.

2.8 Program Kelima (Nested For 0)

Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut “nestedFor1”. Dan masukkan kode pemrograman berikut.

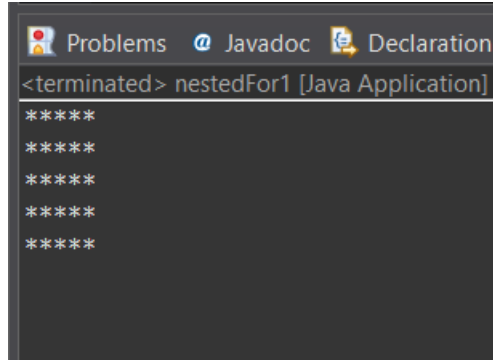


Gambar 2.15

2.8.1 Output

Program mencetak pola bintang berbentuk persegi 5x5. Perulangan luar (i) menentukan jumlah baris, sedangkan perulangan dalam (j) menentukan jumlah kolom.

Maka, *output* yang dihasilkan adalah:



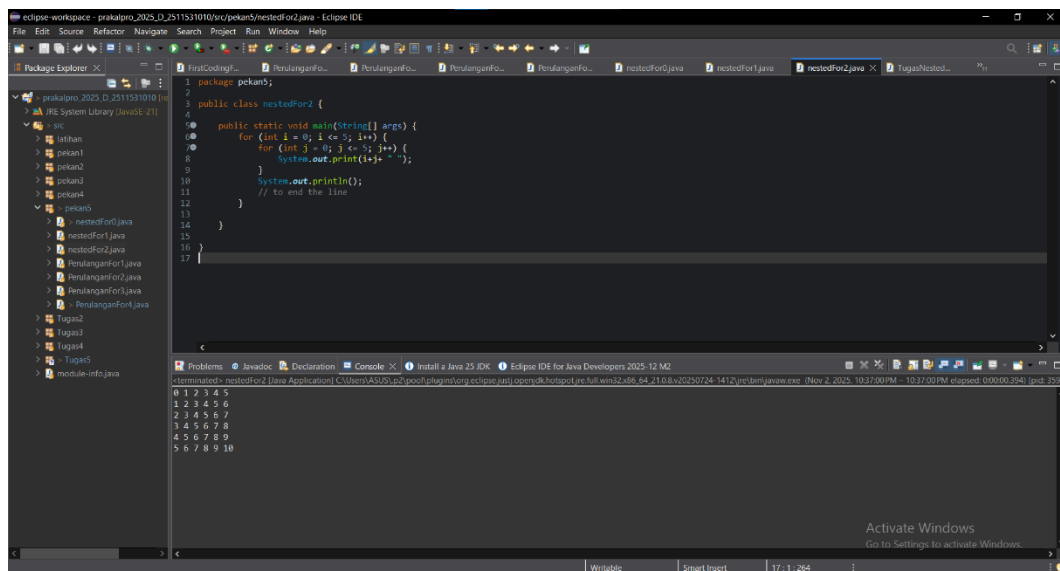
Gambar 2.16

2.8.2 Penjelasan Singkat

Nested loop ini digunakan untuk mencetak bentuk sederhana seperti persegi atau persegi panjang. Struktur ini menjadi dasar logika dalam pembuatan pola visual, misalnya segitiga, piramida, atau pola simetris. Melalui contoh ini, mahasiswa memahami bagaimana dua *loop* dapat bekerja sama menghasilkan keluaran berbentuk *grid*.

2.9 Program Ketujuh (Nested For 2)

Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut “nestedFor2”. Dan masukkan kode pemrograman berikut.



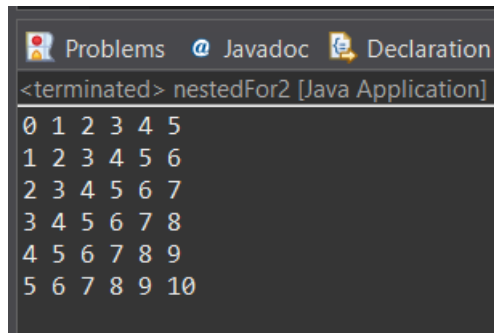
Gambar 2.17

2.9.1 Output

Program menampilkan hasil penjumlahan antara dua variabel perulangan *i* dan *j* dalam bentuk tabel. Variabel *i* berperan sebagai

baris, sedangkan *j* sebagai kolom. Setiap nilai yang dicetak merupakan hasil penjumlahan keduanya.

Maka, *output* yang dihasilkan adalah:



```
Problems Javadoc Declaration
<terminated> nestedFor2 [Java Application]
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 2.18

2.9.2 Penjelasan Singkat

Program ini memperlihatkan penerapan *nested loop* dalam operasi aritmetika dua dimensi. Konsep ini sangat berguna dalam berbagai aplikasi, seperti pembuatan tabel, matriks, atau logika. Melalui pola ini, kita bisa memahami bagaimana dua perulangan saling berinteraksi untuk menghasilkan kombinasi nilai yang kompleks dan terstruktur.

BAB III

PENUTUP

3.1 Kesimpulan

Dari hasil praktikum mengenai perulangan *for* dan *nested for* pada bahasa pemrograman *Java*, dapat disimpulkan bahwa struktur perulangan merupakan salah satu konsep dasar yang sangat penting dalam pemrograman. Perulangan *for* digunakan ketika jumlah pengulangan sudah diketahui secara pasti, sehingga mempermudah pengulangan instruksi tanpa perlu menulis kode secara berulang. Melalui beberapa contoh program yang diuji, dapat dipahami bahwa perulangan *for* mampu menghasilkan pola keluaran, seperti menampilkan deretan angka, menghitung jumlah total, atau membentuk pola tertentu.

Sementara itu, *nested for* atau perulangan bersarang menunjukkan bagaimana sebuah perulangan dapat berada di dalam perulangan lainnya untuk membentuk pola, seperti tabel angka atau bentuk bintang. Struktur ini banyak digunakan dalam kasus pemrograman yang melibatkan data dua dimensi atau tampilan visual pola.

Secara keseluruhan, praktikum ini membantu memahami logika dasar pengulangan, pengendalian variabel, serta cara kerja program dalam mengeksekusi blok kode secara berurutan berdasarkan kondisi tertentu. Pemahaman terhadap konsep ini menjadi dasar yang penting sebelum mempelajari struktur kontrol program yang lebih lanjut seperti *while*, *do-while*, dan penggunaan *loop control statements*.

DAFTAR PUSTAKA

- [1] Kadir, Abdul. (2013). *Dasar Pemrograman Java*. Yogyakarta: Andi Offset.
- [2] Wahana Komputer. (2020). *Belajar Java untuk Pemula*. Yogyakarta: Andi Publisher.
- [3] Purnama, Eko. (2021). *Logika dan Algoritma Pemrograman*. Bandung: Informatika.
- [4] Kementerian Komunikasi dan Informatika. (2021). *Modul Dasar Pemrograman Java*. Jakarta: Kominfo Digital Talent Scholarship.