LAPORAN PRAKTIKUM

PEMROGRAMAN ALGORITMA DAN PEMROGRAMAN

PERULANGAN WHILE DAN DO WHILE

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 5 November 2025 yang membahas tentang bahasa pemograman java spesifiknya pada bagian perulangan *while* dan *do while*, 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, 8 November 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pemrograman, perulangan (*looping*) merupakan salah satu struktur yang sangat penting untuk mengotomatisasi tugas yang bersifat berulang. Bahasa pemrograman Java menyediakan berbagai jenis perulangan, salah satunya adalah *while loop* dan *do-while loop*. Kedua jenis perulangan ini digunakan ketika jumlah pengulangan belum diketahui secara pasti di awal, melainkan bergantung pada kondisi tertentu yang ditentukan selama program berjalan.

Pemahaman terhadap *looping* sangat diperlukan karena menjadi dasar dari berbagai aplikasi, seperti pengolahan data, validasi input, simulasi, serta sistem interaktif. Dengan memahami konsep *while* dan *do-while*, mahasiswa dapat menulis kode yang lebih efisien dan sesuai dengan logika pemrograman yang terstruktur. Praktikum ini dirancang untuk melatih pemahaman mahasiswa terhadap cara kerja kedua jenis perulangan tersebut serta penerapannya dalam situasi nyata melalui contoh program sederhana seperti simulasi permainan dan pengulangan input pengguna.

1.2 Tujuan Praktikum

Tujuan dari praktikum ini adalah:

1. Memahami konsep dasar perulangan *while* dan *do-while* dalam bahasa Java.
2. Mengetahui perbedaan mendasar antara *while* dan *do-while loop*.
3. Mampu mengimplementasikan struktur perulangan untuk menyelesaikan permasalahan logika pemrograman.
4. Melatih kemampuan analisis mahasiswa dalam memahami alur eksekusi program yang bersifat interaktif atau bergantung pada kondisi pengguna.

1.3 Manfaat Praktikum

Manfaat dari praktikum ini adalah:

1. Mahasiswa dapat menguasai salah satu konsep mendasar dalam pemrograman, yaitu struktur perulangan.
2. Mahasiswa mampu menulis dan menganalisis kode Java yang menggunakan *while* dan *do-while loop.*
3. Mahasiswa memahami cara membuat program interaktif yang bergantung pada kondisi input pengguna.
4. Menumbuhkan keterampilan berpikir logis dan terstruktur dalam penyelesaian masalah melalui algoritma pengulangan.

BAB II

PEMBAHASAN

2.1 Pengertian Perulangan While dan Do-While

Dalam bahasa pemrograman Java, perulangan (*looping*) adalah suatu mekanisme yang memungkinkan blok kode dijalankan berulang kali selama kondisi tertentu terpenuhi. Struktur ini sangat penting dalam pemrograman karena banyak proses dalam komputer bersifat berulang, seperti membaca data, menghitung total nilai, atau menunggu input dari pengguna. Dengan perulangan, programmer tidak perlu menulis instruksi yang sama berkali-kali, cukup menuliskan satu blok perintah yang dijalankan berulang berdasarkan kondisi yang ditetapkan.

Dua bentuk utama dari perulangan bersyarat di Java adalah *while loop* dan *do-while loop*. Keduanya berfungsi untuk mengulang eksekusi kode hingga kondisi logika (*boolean condition*) bernilai *false*, namun keduanya memiliki perbedaan mendasar dalam cara mengevaluasi kondisi tersebut.

Perulangan *do-while* memiliki pola yang berbeda. Pada perulangan ini, blok perintah dijalankan terlebih dahulu minimal satu kali, baru setelah itu kondisi diperiksa. Jika kondisi masih bernilai *true*, maka perulangan akan berlanjut.

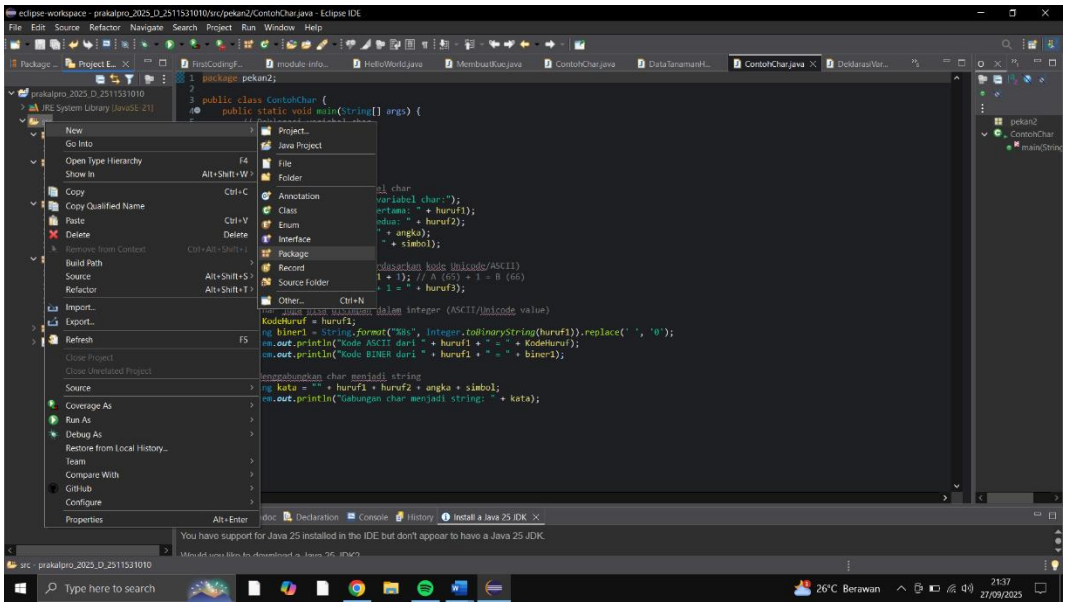
Secara umum:

1. While loop digunakan jika diperlukan pengujian kondisi di awal.
2. Do-while loop digunakan jika setidaknya ingin menjalankan perintah satu kali terlebih dahulu, baru dilakukan pengecekan kondisi.

Kedua struktur perulangan ini penting karena sering digunakan dalam berbagai situasi di dunia nyata, seperti validasi login, menu interaktif, atau proses yang memerlukan pengulangan hingga kondisi tertentu tercapai.

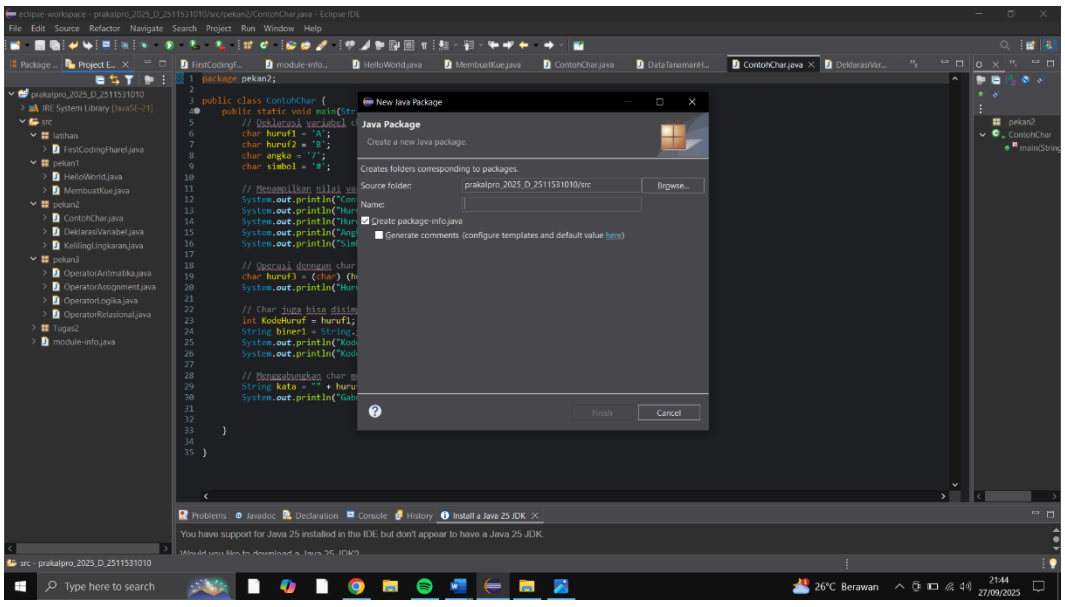
2.2 Pembuatan Package dan Class Pekan 6

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



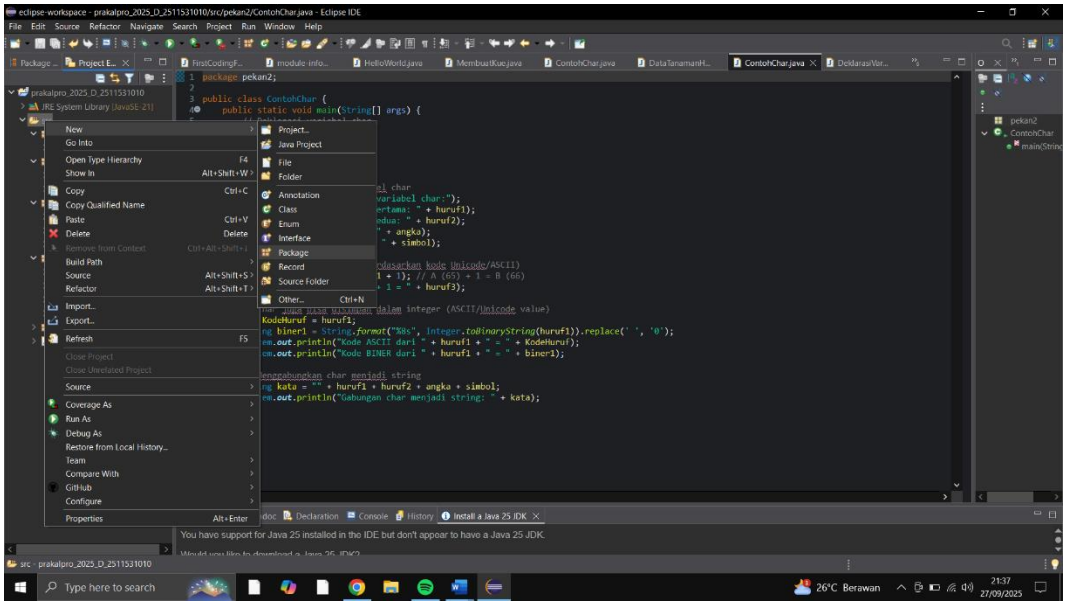
Gambar 2.1

2. Lanjut diberi nama *package* tanpa pakai spasi, huruf kapital ataupun karakter khusus. Seperti "pekan6_2511531010" dan tekan *Finish.*



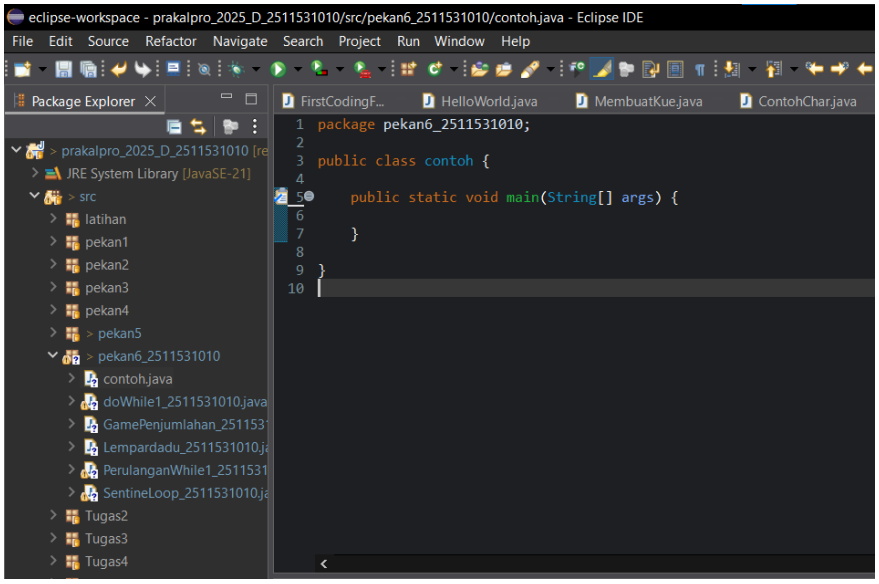
Gambar 2.2

3. Kemudian klik kanan pada package "pekan6_2511531010" tersebut dan klik "*New*" dan pilih bagian "*Class*" untuk memulai membuat program.



Gambar 2.3

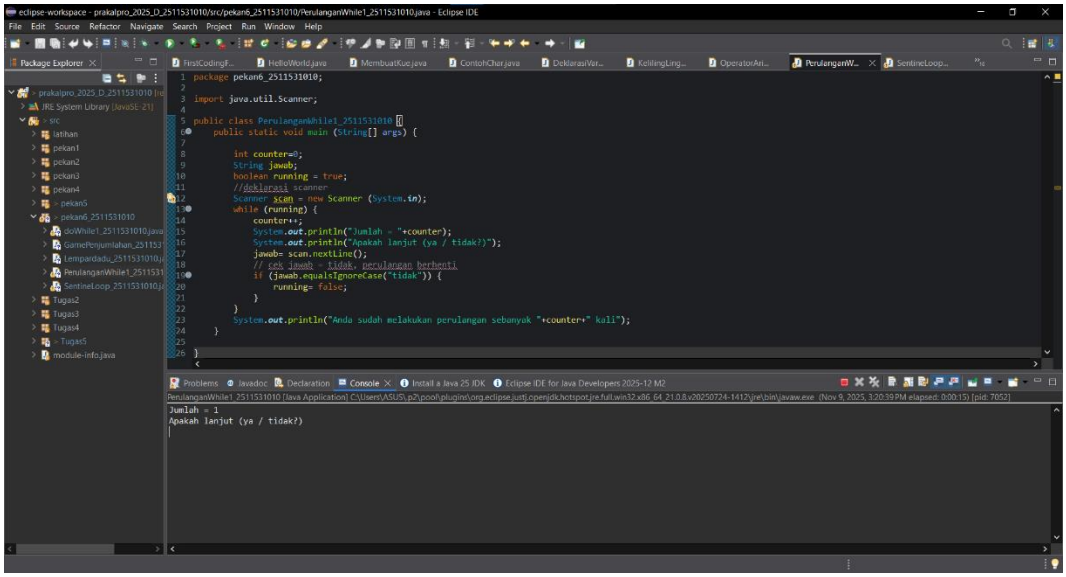
4. Buat nama *Class* yang akan dibuat pada bagian nama tanpa spasi dan menggunakan huruf kapital pada awal kata, lalu klik bagian "*public static void main(string[]args)*". Kemudian klik *finish*.



Gambar 2.4

2.3 Program Pertama (Perulangan While 1)

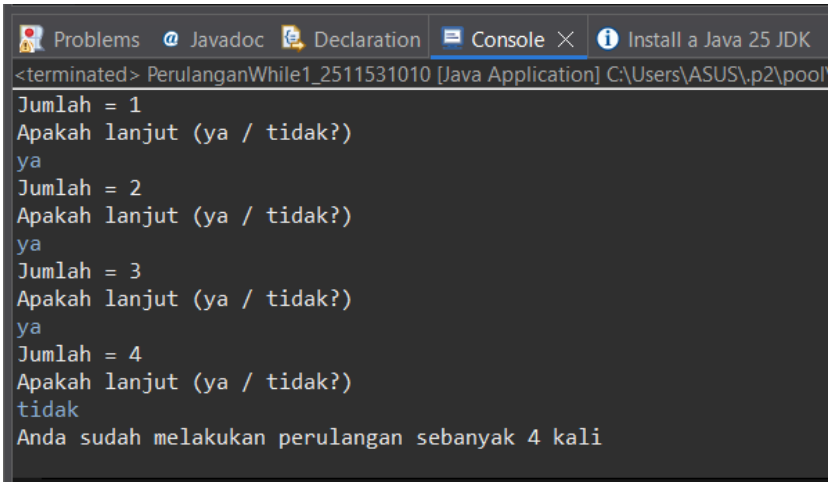
Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut "PerulanganWhile1". Dan masukkan kode pemograman berikut.



Gambar 2.5

2.3.1 Output

Program ini menjalankan perulangan yang terus meminta pengguna menjawab apakah ingin melanjutkan atau tidak. Saat pertama kali dijalankan, program akan menampilkan nilai *counter* mulai dari 1. Setiap kali pengguna menjawab “ya”, perulangan akan bertambah satu kali dan *counter* meningkat.Jika pengguna mengetik “tidak”, kondisi *running = false* membuat perulangan berhenti. Akhirnya, program menampilkan pesan berisi jumlah total pengulangan yang telah dilakukan. Sebagai contoh, jika pengguna mengetik “ya” sebanyak tiga kali dan “tidak” pada keempat kalinya, output akhirnya adalah:



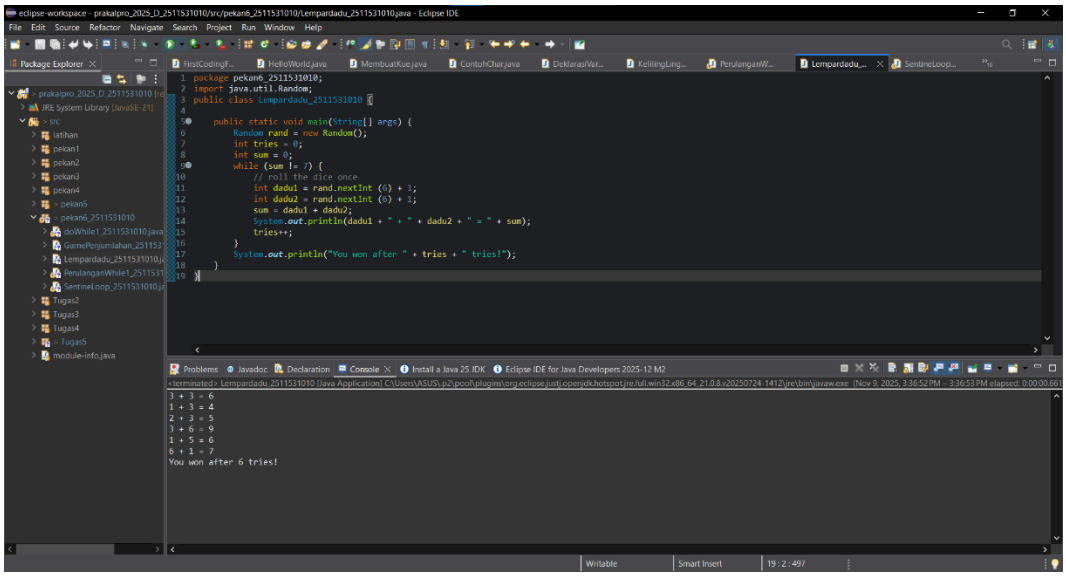
Gambar 2.6

2.3.2 Penjelasan Singkat

Program ini menunjukkan penggunaan *while loop* untuk mengontrol perulangan berdasarkan masukan pengguna. Selama kondisi *running* bernilai *true*, program terus berjalan.

2.4 Program Kedua (Lempar Dadu)

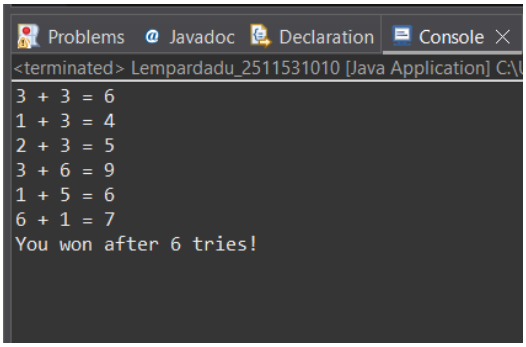
Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut "LemparDadu". Dan masukkan kode pemograman berikut.



Gambar 2.7

2.4.1 Output

Program ini mensimulasikan permainan melempar dua buah dadu. Dalam setiap iterasi *while*, dua angka acak (1–6) dijumlahkan. Perulangan terus berjalan sampai jumlah dua dadu sama dengan 7. Setiap percobaan akan menampilkan hasil lemparan dan jumlah total. Setelah mendapatkan angka 7, program berhenti dan menampilkan berapa kali percobaan dilakukan hingga menang.



Gambar 2.8

2.4.2 Penjelasan Singkat

Kode ini menunjukkan bagaimana *while loop* digunakan untuk menjalankan simulasi yang berulang hingga kondisi tertentu terpenuhi. Karena jumlah lemparan tidak diketahui di awal, *while* menjadi pilihan ideal.

2.5 Program Ketiga (Game Penjumlahan)

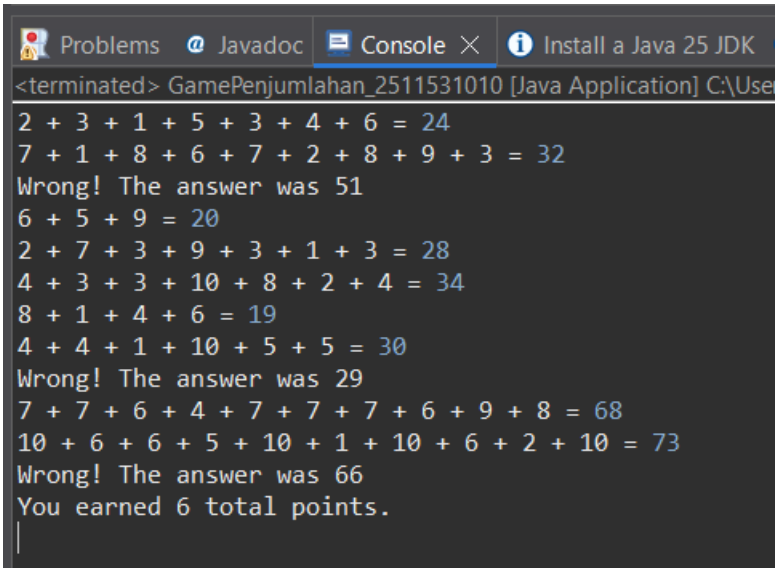
Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut "GamePenjumlahan". Dan masukkan kode pemograman berikut.

```java
package pekan6_2511531010;
import java.util.Random;

public class GamePenjumlahan_2511531010 {

    public static void main(String[] args) {
        Scanner console = new Scanner (System.in);
        Random rand = new Random();
        // play until user gets 3 wrong
        int points = 0;
        int wrong = 0;
        while (wrong < 3) {
            int result = play(console, rand);    // play one game
            if (result > 0) {
                points++;
            } else {
                wrong++;
            }
        }
        System.out.println("You earned " + points + " total points.");
    }
    // membuat soal penjumlahan dan ditampilkan ke user
    public static int play (Scanner console, Random rand) {
        // print the operands being added, and sum them
        int operands = rand.nextInt(10) + 1;
        int sum = rand.nextInt(10) + 1;
        System.out.print(sum);
        for (int i = 2; i <= operands; i++) {
            int n = rand.nextInt(10) + 1;
            sum += n;
            System.out.print(" + " + n);

        }
        System.out.print(" = ");

        // read user's guess and report whether it was correct
        int guess = console.nextInt();
        if (guess == sum) {
            return 1;
        } else {
            System.out.println("Wrong! The answer was " + sum);
            return 0;
        }
    }

}
```

Gambar 2.9

2.5.1 Output

Program ini meminta pengguna menjawab hasil dari beberapa soal penjumlahan acak. Selama pengguna belum salah tiga kali, permainan akan terus berlanjut. Jika jawaban benar, poin bertambah satu. Struktur *while* digunakan untuk mengontrol jumlah kesalahan yang diperbolehkan. Jika salah, nilai *wrong* bertambah. Setelah salah tiga kali, permainan berhenti. *Output* akhir menampilkan total poin yang diperoleh.



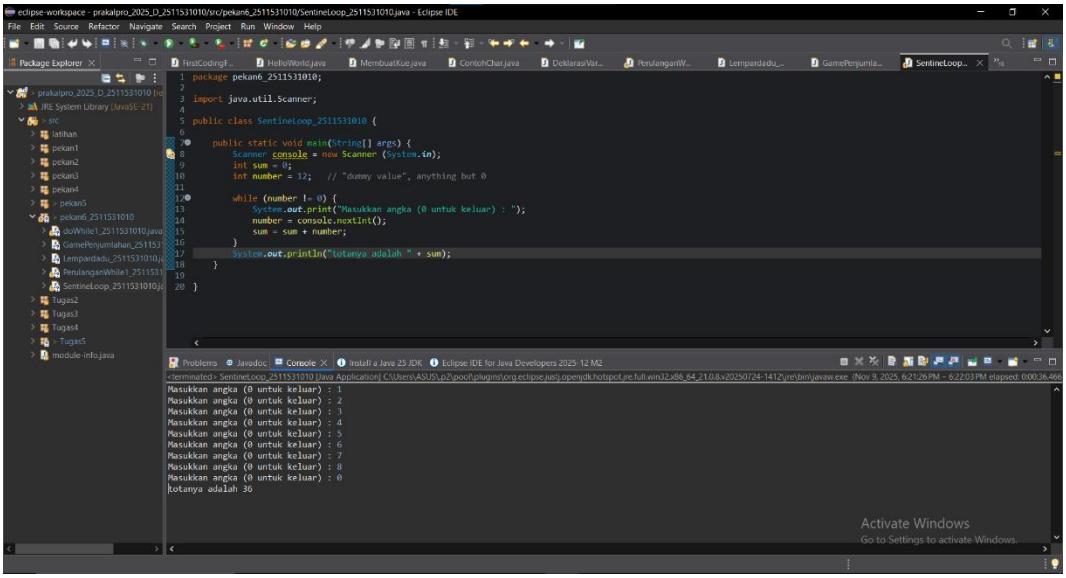
Gambar 2.10

2.5.2 Penjelasan Singkat

Program ini menggunakan struktur *while* untuk membatasi jumlah kesalahan yang diperbolehkan. Konsepnya menyerupai permainan edukatif berbasis logika, di mana program menggabungkan fungsi acak (*Random*) dan pengulangan untuk menciptakan tantangan yang dinamis.

2.6 Program Keempat (Sentinel Loop)

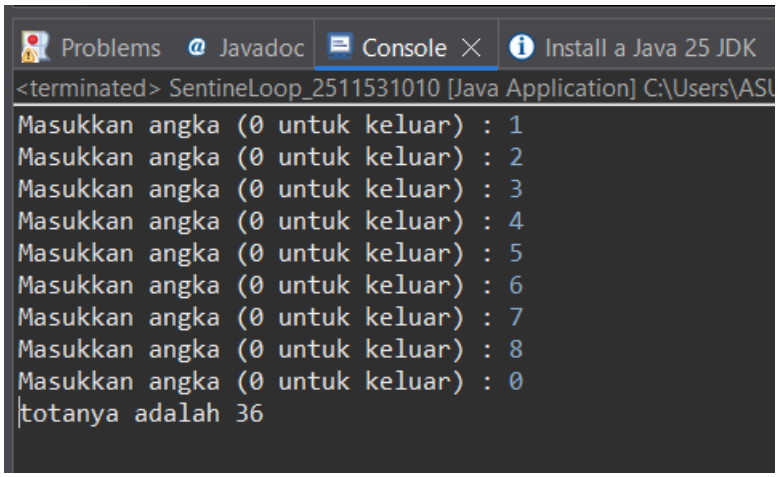
Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut “SentinelLoop”. Dan masukkan kode pemograman berikut.



Gambar 2.11

2.6.1 Output

Pada program ini, pengguna diminta memasukkan angka secara berulang. Setiap angka yang dimasukkan akan ditambahkan ke variabel *sum* yang berfungsi sebagai penampung total. Proses perulangan akan terus berlangsung hingga pengguna mengetik angka 0, yang berfungsi sebagai nilai sentinel atau penanda akhir. Setelah angka 0 dimasukkan, program akan berhenti dan menampilkan hasil penjumlahan dari seluruh angka yang telah dimasukkan sebelumnya.



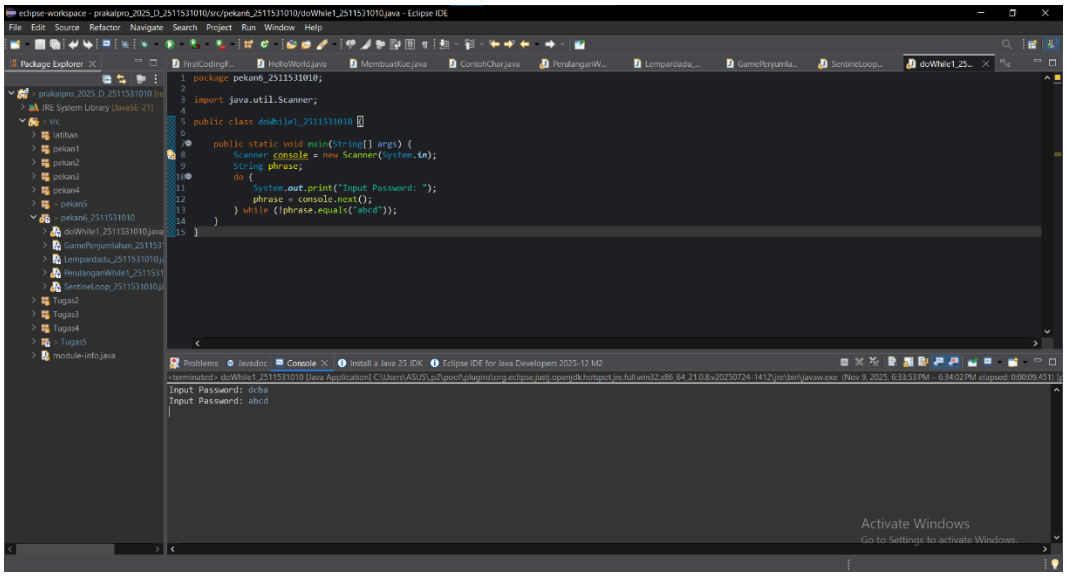
Gambar 2.12

2.6.2 Penjelasan Singkat

Program SentinelLoop menunjukkan penerapan konsep *sentinel value* dalam perulangan. Nilai sentinel digunakan untuk mengontrol kapan perulangan berhenti tanpa memerlukan batas tertentu.

2.7 Program Kelima (Do While 1)

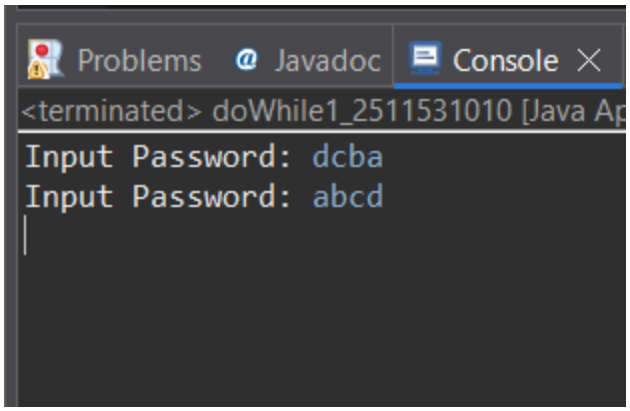
Buat *Class* seperti program sebelumnya dan beri nama *Class* tersebut "doWhile1". Dan masukkan kode pemograman berikut.



Gambar 2.13

2.7.1 Output

Program ini meminta pengguna untuk memasukkan sebuah password. Ketika dijalankan, perintah "Input Password:" akan selalu muncul minimal satu kali. Jika pengguna mengetik password yang salah, maka program akan mengulangi perintah tersebut hingga pengguna mengetik kata sandi yang benar, yaitu "abcd". Ketika kondisi sudah terpenuhi, perulangan berhenti dan program selesai tanpa menampilkan pesan tambahan.



Gambar 2.14

2.7.2 Penjelasan Singkat

Program doWhile1 menunjukkan perbedaan utama antara *do-while* dan *while*. Meskipun kondisi salah sejak awal, perintah dalam blok *do* tetap dijalankan satu kali sebelum dilakukan pemeriksaan. Struktur ini efektif digunakan dalam situasi di mana program harus menampilkan sesuatu terlebih dahulu, misalnya dalam proses *login*, validasi data, atau menu konfirmasi yang membutuhkan interaksi minimal satu kali dengan pengguna.

BAB III

PENUTUP

3.1 Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa perulangan merupakan salah satu struktur kontrol yang sangat penting dalam pemrograman. Perulangan *while* digunakan ketika kondisi harus diperiksa sebelum eksekusi blok program, sementara *do-while* lebih cocok ketika program harus dijalankan minimal satu kali sebelum pengecekan kondisi dilakukan. Melalui berbagai contoh seperti permainan dadu, *game* penjumlahan, dan penginputan data, kita dapat memahami bahwa penggunaan perulangan membuat program menjadi lebih dinamis, efisien, dan interaktif.

Selain itu, praktikum ini membantu memperkuat pemahaman mahasiswa dalam menghubungkan teori kontrol alur dengan penerapan nyata dalam bahasa Java. Dengan memahami konsep perulangan dengan baik, mahasiswa akan lebih siap untuk menyusun algoritma kompleks di tingkat yang lebih lanjut.

DAFTAR PUSTAKA

[1] Deitel, P. J., & Deitel, H. M.. 2017. *Java: How to Program (10th Edition)*. Pearson Education.

[2] Oracle. 2024. *The while and do-while Statements (Java Documentation)*. https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html. [Diakses: 8-November-2025].

[3] Wahana Komputer. 2021. *Belajar Pemrograman Java dari Nol*. Yogyakarta: Andi Publisher.

[4] Sun Microsystems. 2023. *Java Platform, Standard Edition Language and Virtual Machine Specifications*. Oracle Corporation.