# LAPORAN PRATIKUM

# ALGORITMA DAN PEMROGARAMAN

# PERULANGAN WHILE DAN DO-WHILE


Disusun Oleh:

Khairun Nisa

2511532015

Dosen Pengampu:

DR. WAHYUDI, S.T, M.T

Asisten Pratikum:

Aufan Taufiqurrahman




FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS


2025

# KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas terselesaikannya laporan praktikum Algoritma dan Pemrograman ini. Laporan ini disusun sebagai dokumentasi hasil kegiatan praktikum yang telah dilaksanakan pada 4 November 2025, dengan pembahasan mengenai perulangan while dan do-while.

Penulis mengucapkan terima kasih kepada Bapak Dosen serta asisten laboratorium yang telah membimbing selama praktikum berlangsung. Semoga laporan ini dapat bermanfaat bagi pembaca dan menjadi referensi untuk perkembangan ilmu pengetahuan di bidang pemrograman.

Padang, 5 September 2025

Penulis

# DAFTAR ISI

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Dalam pengembangan program computer, sering kali diperlukan suatu mekanisme untuk mengeksekusi sekolompok instruksi secara berulang hingga kondisi tertentu terpenuhi. Proses pengulangan ini dikenal sebagai *looping* atau perulangan, dan merupakan salah satu konsep dasar dalam struktur kendali alur program. Dua bentuk perulangan yang paling umum digunakan dalam bahasa pemrograman prosedural, seperti bahasa java adalah perulangan while dan do-while.

Perulangan while digunakan untuk mengulang suatu proses perulangan yang belum diketahui jumlahnya. Pada perulangan while pengecekan kondisi akan dilakukan terlebih dahulu. Jika kondisi masih bernilai benar (true) maka perulangan akan terus dilakukan. Sebaliknya jika bernilai salah maka akan dihentikan [1]. Sedangkan perulangan do-while merupakan perulangan yang hamper sama dengan perulangan while, namun pernyataan dilakukan terlebih dahulu kemudian dilakukan pengecekan.

Do-while adalah salah satu pernyataan pengulanagn yang memungkinkan kita untuk membuat program berjalan secara fleksibel berdasarkan keinginan kita. Do-while berfungsi untuk mengullangi pengeksekusian beberapa substatement berdasarkan conditional expression yang ada. Untuk menentukan waktu yang tepat untuk menggunakan while dan do-while adalh tergantung dari kasusnya. Syarat perulangannya tidak berkaitan dengan hasil hitung pada blok kode yang diulang, maka pakailah while. Tetapi, bila syarat perulangannya berkaitan dengan hasil perhitungan di blok kode yang diulang, maka pakailah do-while [2].

Pemahaman terhadap perbedaan dan penerapan yang tepat dari while dan do-while sangat penting bagi mahasiswa dalam merancang algoritma yang efesien dan logis. Selain itu, penggunaan perulangan yang tidak tepat dapat menyebabkan *infinte loop* (perulangan tak hingga), yang merupakan kesalaha umum dalam pemrograman.

## 1.2 Tujuan Praktikum

1. Memahami konsep dasar perulangan (*looping*) dalam pemrograman, khusunya struktur kendali while dan do-while.
2. Mampu membedakan karakteristik dan mekanisme kerja antara perulangan while dan do-while.
3. Melatih kemampuan dalam menulis, mengimplementasikan, dan mengeksekusi program yang menggunakan perulangan untuk menyelesaikan masalah tertentu.
4. Meningkatkan kemampuan logika pemrograman melalui penggunaan struktur perulangan yang efisien dan tepat.
5. Memahami kondisi berhenti (*loop termination condition*) dan menghindari perulangan tak terbatas (*infinite loop*).

## 1.3 Manfaat Praktikum

1. Memberikan pemahaman praktis tentang bagaimana perulangan digunakan untuk mengeksekusi blok kode berulang kali selama kondisi tertentu terpenuhi.
2. Membantu mahasiswa dalam mengembangkan algoritma yang lebih efisien dengan memamnfaatkan struktur perulangan.
3. Meningkatkan keterampilan pemecahan masalah melalui penerapan perulangan dalam berbagai studi kasus, seperti validasi input, penghitungan, atau iterasi data.
4. Menjadi dasar penting untuk mempelajari struktru data dan algoritma yang lebih kompleks di Tingkat lanjut.
5. Memberikan pengalaman langsung dalam debugging dan menangani kesalahan umum dalam penggunaan perulangan.

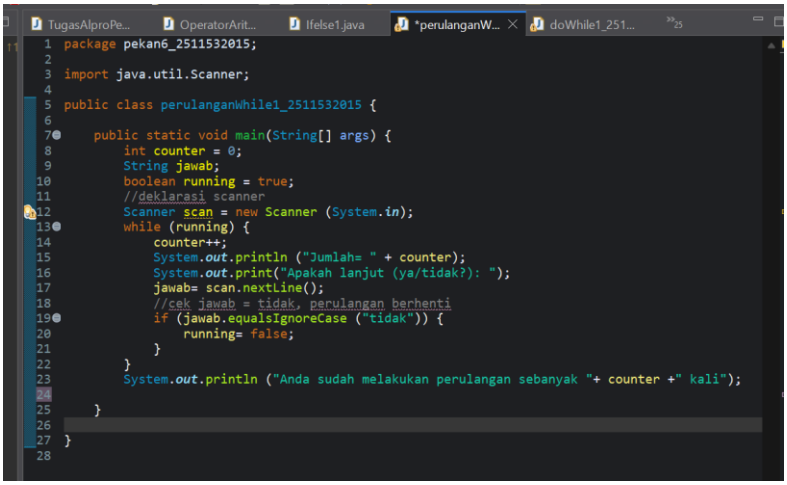
# BAB II
# PEMBAHASAN

## 2.1 Perulangan While

Pernyataan while loop adalah pernyataan atau blok pernyataan yang diulang-ulang sampai mencapai kondisi yang cocok. Perulangan dengan menggunakan teknik while ini sebenarnya adalah suatu bentuk perulangan yang memodifikasi teknik pencabangan (*branching*) secara kasar. Pernyataan yang ada di dalam blok perulangan akan dieksekusi dengan cara memeriksa ekspresi yang ada, sepanjang ekspresi berniali true maka statement akan terus di eksekusi. Variabel sebagai kontrol perulangan bentuk ini diinisialisai di luar blok perulangan ini. Dan penambahan atau increment nilai variabel berada di dalam blok perulangan ini. Kelebihan perulangan dengan bentuk ini adalah variabel yang dideklarasikan tidak hanya bertipe integer atau float saja namun bisa juga bertipe Boolean atau string [3].

Pada praktikum ini, dibuat sebuah program sederhana yang memnfaatkan struktur perulangan while untuk menghitung berapa kali suatu proses dijalankan berdasarkan keputusan pengguna.

1. Sebelum mengetikkan kode program, terlebih dahulu membuat package baru dengan cara yang sama seperti parktikum minggu sebelumnya, dan diberi nama ‘pekan 6’. Kemudian buat class baru di dalam package pekan 6, dan diberi nama ‘perulanganWhile’.
2. Kemudian ketikkan kode program seperti berikut:

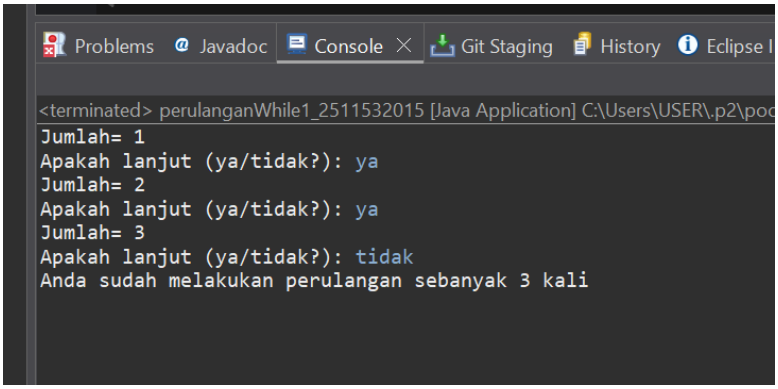


Gambar 2.1: Kode program perulanganWhile

Program ini dirancang menggunakan struktur perulangan while yang akan terus mengeksekusi blok kode selama kondisi tertentu bernilai true.

- Variabel counter diinisialisasikan dengan nilai 0 untuk mencatat jumlah perulangan.
- Variabel running diatur sebagai true agar perulanagn while dapat dimulai.
- Di dalam loop, program akan:
  a) Menambah nilai counter sebanyak 1 (counter++)
  b) Menampilkan pesan bahwa proses ke-counter sedang berjalan
  c) Meminta input dari pengguna dengan menggunakan class scanner (Apakah ingin melanjutkan (ya/tidak))
- Jika pengguna memasukkan 'ya' perulangan akan terus berjalan, jika pengguna memsukkan 'tidak, maka nilai running diubah menjadi false, sehingga perulangan akan berhenti pada iterasi berikutnya.

3. Setelah dijalankan program akan menghasilkan:



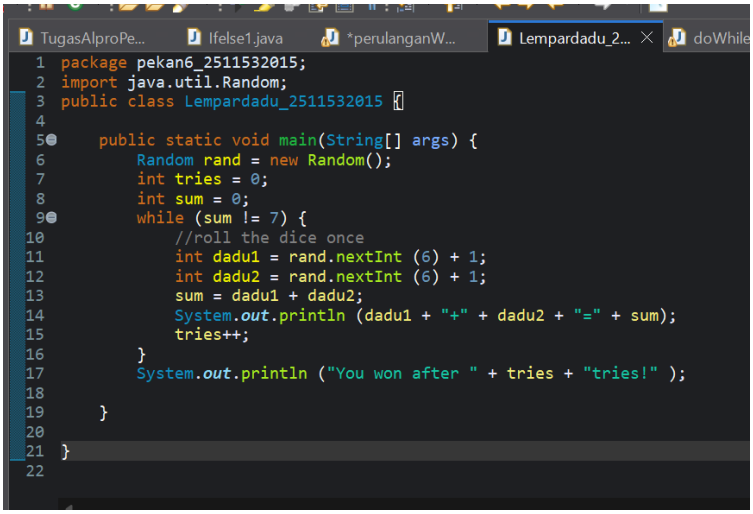

Gambar 2.2: Hasil Output program perulanganWhile

Setelah program dijalankan, dapat diamati bahwa perulangan while berhasil mengeksekusi blok kode berulang kali sesuai dengan keinginan pengguna. Misalnya, jika pengguna memilih 'ya' sebanyak 2 kali dan 'tidak' pada percobaa ke-3, maka program akan mencetak prose ke-1 dan ke-2. Setelah input 'tidak' loop berhenti dan program menampilkan "Anda sudah melakukan perulangan sebanyak 3 kali".

## 2.2 Lempar Dadu

Program ini mensimulasikan permainan lempar dadu. Program ini menggunakan program bilangan random atau acak dengan kelas java.util.Random. Kelas java.util.Random digunakan untuk menghasilkan angak pseudorandom. Metode yang diimplementasikan oleh kelas sini digunakan untuk menghasilkan tipe data acak yang berbeda seperti int, double, dan float [4]. Dalam program ini komputer akan terus melempar dua dadu (masing-masing bernilai 1-6) secara acak, dan akan berhenti hanya ketika jumlah kedua dadu tersebut sama dengan 7. Program ini juga menghitung berapa kali percobaan (tries) yang diperlukan hingga mendapatkan hasil 7. Berikut langkah-langkah pembuatan programnya:

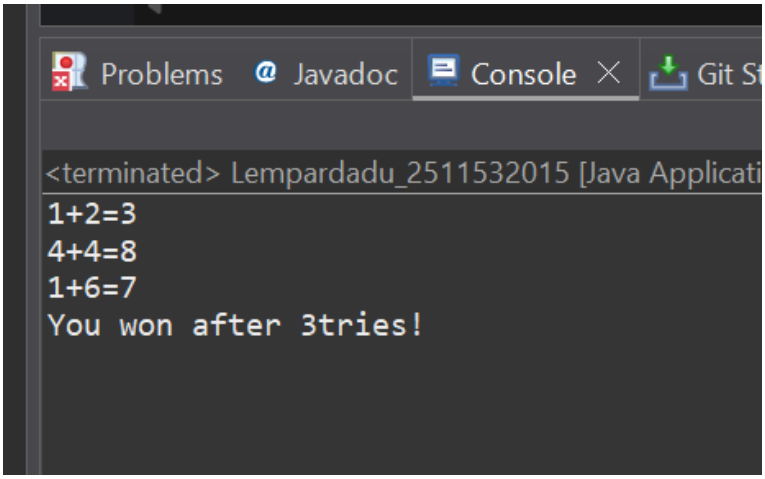
1. Sebelum menegtikkan kode program, terlebih dahulu membuat class baru di dalam package pekan 6 dan diberi nama ‘lemparDadu’.
2. Kemudian ketikkan kode program seperti berikut:




Gambar 2.3: Kode program lemparDadu

Program ini dimulai dengan membuat objek dari kelas Random untuk menghasilkan angka acak, lalu menginisialisasi dua variabel: yaitu tries yang bernilai 0 untuk menghitung jumlah percobaan, dan sum yang juga bernilai 0 untuk menyimpan jumlah nilai dari dua dadu.

3. Ketika dijalankan, program akan menghasilkan output:




Gambar 2.4: Hasil ouput program lemparDadu

Perulangan akan terus berjalan selama jumlah kedua dadu belum sama dengan 7 (while (sum != 7)). Di dalam perulangan, program menghasilkan dua bilangan acak antara 1-6 (masing-masing mewakili nilai satu dadu), menjumlahkannya, lalu mencetak hasil lemparan tersebut. Setiap kali program melempar dadu, variabel tries ditambah 1 (tries++) untuk mencatat bahwa satu percobaan telah dilakukan. Setelah mencetak hasil, program Kembali memeriksa kondisi perulangan, jika jumlah kedua dadu masih belum 7, perulangan diulang, namun setelah jumlah kedua dadu sama dengan 7 perulangan dihentikan dan menampilkan pesan bahwa permainan selesai dan menyebutkan berapa kali percobaan yang dilakukan untuk mendapatkan hasil 7.

## 2.3 Game Penjumlahan

Program ini adalah permainan interaktif sederhana yang dirancang untuk melatih kemampuan penjumlahan dasar. Game ini menggunakan struktru perulangan while untuk membatasi jumlah percobaan hingga 3 kali kesalahan, dan memanggil kode terpisah Bernama play() untuk membuat satu putaran soal penjumlahan. Setiap kali pengguna menjawab benar, ia mendapat poin, jika salah ia kehilangan satu kesempatan. Permainan berhenti ketika pengguna sudah salah sebanyak 3 kali atau ketika ia menyelesaikan semua putaran tanpa batas, tapi karena kondisi while (wrong

< 3) digunakan, maka permainan akan berhenti setelah 3 kali kesalahan. Berikut langkah-langkah pembuatan kode program GamePenjumlahan:

1. Sebelum mengetikkan kode program terlebih dahulu membuat class baru di package pekan 6 dengan nama 'GamePenjumlahan'
2. Kemudian ketikkan kode program seperti di bawah ini:

```
package pekan6_2511532015;
import java.util.Scanner;
import java.util.Random;
public class GamePenjumlahan_2511532015 {

    public static void main(String[] args) {
        Scanner console = new Scanner (System.in);
        Random rand = new Random ();
        //play util user gets 3 wrong
        int points = 0;
        int wrong = 0;
        while (wrong < 3) {
            int result = play(console, rand); //play once 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 (4) + 2;
        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 (" = ");

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

}
```

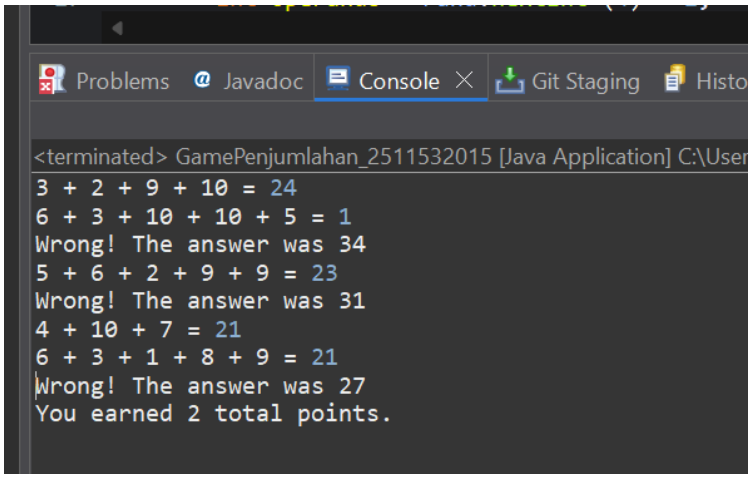
Gambar 2.5: Kode program GamePenjumlahan

Pada awal program, di dalam metode main (), dibuat objek Scanner untuk menerima input dari pengguna, dan objek Random untuk menghasilkan angka acak. Dua variabel penting juga diinisialisasikan, points untuk menyimpan skor, dan wrong untuk menyimpan jumlah jawaban salah. Kemudian, program memasuki perulangan while (wrong< 3) yang artinya selama jumlah kesalahan belum mencapai 3, program akan terus berjalan.

Di dalam loop, program memanggil metode play (console, rand) yang merupakan fungsi terpisah yang bertugas membuat satu soal penjumlahan acak, menampilkan soalnya, lalu meminta jawaban dari pengguna. Jika jawaban benar, metode play() mengembalikan nilai 1, dan jika salah mengembalikan 0. Nilai kembalian ini kemudia digunakan di main() untuk memperbarui skor (points++) atau menambah jumlah kesalahan (wrong++).

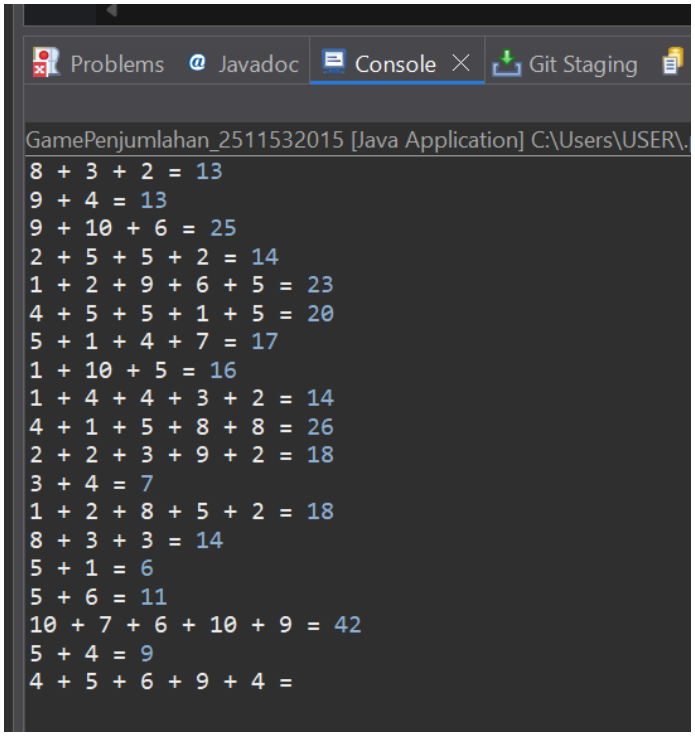
3. Setelah program dijalankan, akan menghasilkan output:

Setelah program dijalankan, alur eksekusinya bergantung pada proses pengguna terhdap soal-soal yang diberikan. Jika pengguna melakukan tiga kesalahan, maka program akan berhenti secara otomatis. Setiap kali jawaban salah, jumlah kesalahan akan bertambah satu dan program menampilkan jawaban yang benar. Begitu kesalahan mencapai angka tiga, perulangan utama berakhir dan program menampilkan pesan akhir berupa total skor yang berhasil dikumpulkan selama bermain.




Gambar 2.6: Hasil output GamePenjumlahan 3 kesalahan

Sebaliknya, jika pengguna tidak pernah melakukan kesalahan, maka program tidak akan pernah berhenti dengan sendirinya. Ia akan terus menghasilkan soal penjumlahan yang baru, meminta jawaban, dan menambah skor setiap kali jawaban benar. Dengan kata lain, selama jawaban user selalu benar, permainan akan berlangsung terus-menerus tanpa akhir.

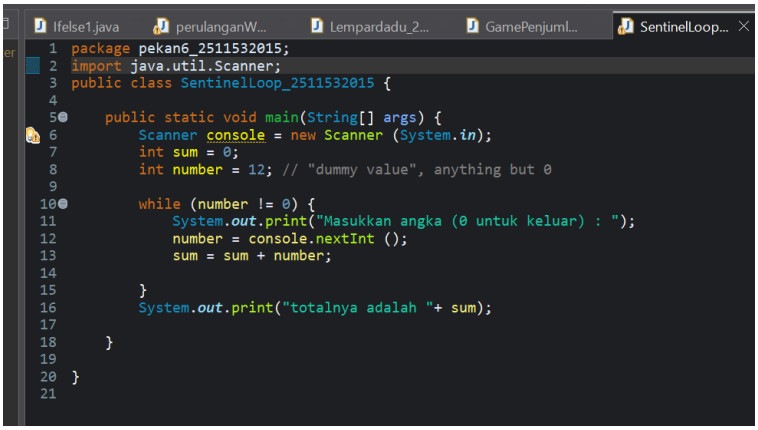



Gambar 2.7: Hasil output GamePenjumlahan tanpa kesalahan

## 2.4 Sentinel Loop

Sifat perulangan Sentinel Controlled Loop yang dimana proses perulangan akan berhenti / berkahir (terminate) apabila ada input Nilai khusu dimasukkan. Jadi, sifat perulangan ini yang tidak terbatas karena diketahui sebelumnya beberapa kali loop / perulangan yang akan dieksekusi. Sentinel Loop biasanya memiliki struktur seperti: nilai dari user dibaca untuk inisialisasi dan selama nilai sentine sama dengan nilai yang dimasukkan oleh user pada pengecekan kondisi maka statement akan dilaksanakan pada loop body dan nilai berikutnya adalah nilai yang diinputkan oleh user. Dan apabila sebuah perulangan dengan kondisinya terus bernilai true maka perulangan akan dilakukan terus menerus dan tidak akan berhenti. Istilah dari hal tersebut merupakan loop tak terbatas (*Infinity Loop*) [5]. Berikut langkah-langkah dalam pembuatan program Sentinel Loop:

1. Sebelum mengetikkan kode program, terlebih dahulu membuat class baru di package pekan 6 dengan nama ‘SentinelLoop_2511532015’.
2. Kemudian ketikkan kode program:



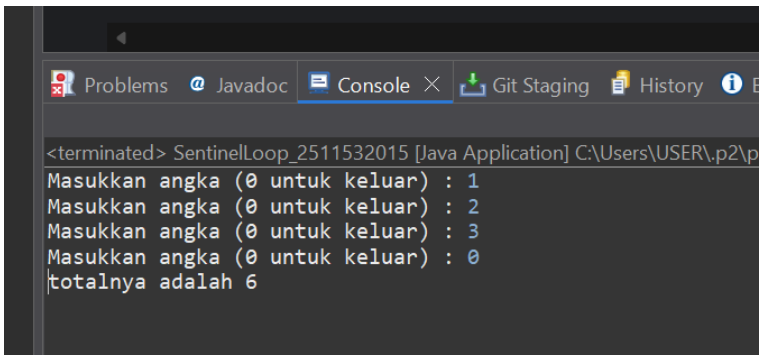
```java
package pekan6_2511532015;
import java.util.Scanner;
public class SentinelLoop_2511532015 {

    public static void main(String[] args) {
        Scanner console = new Scanner (System.in);
        int sum = 0;
        int number = 12; // "dummy value", anything but 0

        while (number != 0) {
            System.out.print("Masukkan angka (0 untuk keluar) : ");
            number = console.nextInt ();
            sum = sum + number;

        }
        System.out.print("totalnya adalah "+ sum);

    }

}
```

Gambar 2.8: Kode program SentinelLoop

Variabel sum diinisialisasikan dengan nilai 0 untuk menyimpan total penjumlahan dan variabel number diberi nilai sembarang (bukan 0) agar loop bisa dimulai. Ini disebut *dummy value* karena hanya memastikan kondisi while pertama kali bernilai true.

Kondisi while (number !=0) memastika bahwa selama angka yang dimasukkan bukan 0, program akan terus meminta input baru. Setiap kali pengguna memasukkan angka, nilai tersebut langsung ditambha ke sum termasuk saat pengguna akhirnya memasukkan 0.

3. Setelah program dijalankan akan menghasilkan output:



```
Problems  Javadoc  Console  Git Staging  History
<terminated> SentinelLoop_2511532015 [Java Application] C:\Users\USER\.p2\p
Masukkan angka (0 untuk keluar) : 1
Masukkan angka (0 untuk keluar) : 2
Masukkan angka (0 untuk keluar) : 3
Masukkan angka (0 untuk keluar) : 0
totalnya adalah 6
```

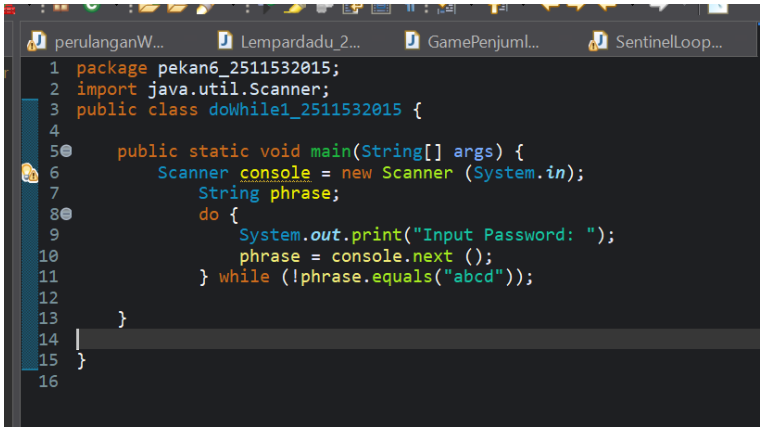
Gambar 2.9: Hasil output SentinelLoop

Setelah program dijalankan, user langsung disambut dengan permintaan untuk memasukkan angka. Setiap angka yang dimasukkan akan segera ditambah ke total, dan proses ini berulang tanpa batas hingga pengguna memasukkan nilai 0. Setelah angka 0 dimasukkan, program melanjutkan ke baris berikutnya, yaitu mencetak total jumlah yang telah dikumpulkan.

## 2.5 Do-While

Perulangan do-while digunakan bila jumlah perulangan belum diketahui. Pada badan perulangan do-while, dapat berisi statement Tunggal atau banyak statement. Pada pernyataan do, mula-mula statement dijalankan, selanjutnya kondisi diuji pada pernyataan while. Pada pernyataan do-while, statement dilaksanakan terlebih dahulu, kemudian setelahnya kondisi diuji. Jika kondisi bernilai benar, maka perulangan dilanjutkan, jika bernilai salah perulangan dihentikan [6]. Berikut langkah-langkah pembuatan program Do-While:

1. Sebelum mengetikkan kode program, terlebih dahulu membuat class baru pada package pekan 6 dengan nama doWhile_2511532015.
2. Kemudian ketikkan kode program:

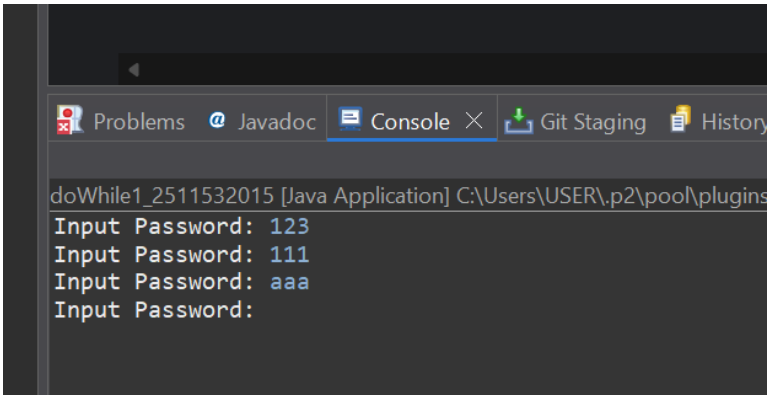



Gambar 2.10: Kode program doWhile

Program ini adalah contoh sederhana penggunaan perulangan do-while yang dirancang untuk terus meminta pengguna memasukkan kata sandi hingga kata sandi bernilai benar yaitu ‘abcd’.

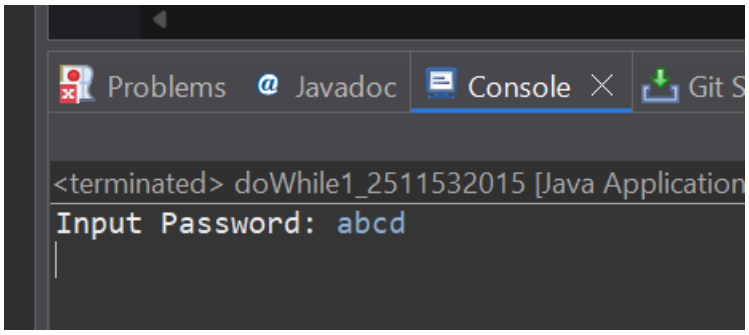
Program dimulai dari class scanner untuk membaca input pengguna melalui konsol. Variabel phrase diinisialisasikan sebagai tipe string tanpa nilai awal, karena akan diisi pengguna melalui input pertama. Perulangan do-while dijalankan, dimana blok do akan dieksekusi minimal satu kali, lalu kondisi while diperiksa. Kondisi tersebut adalah !phrase.equals(“abcd”), yang berarti selama phrase tidak sama dengan ‘abcd’, maka ulang lagi.

3. Ketika dijalankan program akan menghasilkan output:




Gambar 2.11: Hasil output doWhile jika password salah

Ketika program dijalankan akan menampilkan ‘input password:’ dan menunggu pengguna memasukkan password. Jika password yang dimasukkan bukan ‘abcd’, maka do-while akan Kembali ke awal dan menampilkan ‘input password:’ sekali lagi, begitu saja sampai password yang dimasukkan bernilai benar.




Gambar 2.12: Hasil output jika password benar

Namun, ketika pengguna memasukkan password yang tepat, yaitu ‘abcd’ kondisi perulangan while(!phrase.equals(“abcd”)) menjasi false, sehingga perulangan berhenti. Program tidak akan menampilkan apapun lagi.

# KESIMPULAN

## 3.1 Kesimpulan

Praktikum ini memberikan pemahaman mendalam mengenai konsep perulangan (looping) dalam pemrograman java, khususnya struktur kendali while, do-while, dan pola sentinel loop. Melalui serangkaian program praktis, seperti simulasi lempar dadu, game penjumlahan, validasi password, hingga akumulasi input pengguna, dimana setiap jenis perulangan memiliki karakteristik dan konteks pengguna yang berbeda.

Perulangan while cocok digunakan ketika pengecekan kondisi perlu dilakukan sebelum blok kode dieksekusi, sehingga memungkinkan suatu blok tidak pernah dijalankan jika kondisi awal sudah salah. Sebaliknya do-while menjamin bahwa blok kode dieksekusi minimal satu kali, karena pengecekan kondisi dilakukan setelah dieksekusi sangat berguna dalam kasus seperti validasi input atau permintaan password, di mana pengguna setidaknya harus memberikan input sekali. Sementara itu, sentinel loop menunjukkan cara efektif untuk mengelola input berulang tanpa batas jumlah, dengan menggunakan nilai khusus (seperti 0) sebagai penanda akhir.

## DAFTAR PUSTAKA


[1] "Strutur Algoritma Pemrograman: Perulangan," in *Strutur Algoritma Pemrograman: Perulangan*, Indonesia, Kemendiktisanitek, p. 12.

[2] Bellshade, "Perulangan While - Java - Ballshade," Vercel, [Online]. Available: https://bellshade-website.vercel.app/learn/java/basic/perulangan. [Accessed 4 November 2025].

[3] M. R. Muhammad Munawir, "Peurlangan For While Do While," 22 Maret 2021. [Online]. Available: https://id.scribd.com/document/511960226/Makalah-Perulangan-For-While-Do-While. [Accessed 5 November 2025].

[4] J. Squirrels, "Kelas Java.util.Random," CodeGym, 21 July 2023. [Online]. Available: https://codegym.cc/id/groups/posts/id.825.kelas-java-util-random. [Accessed 5 November 2025].

[5] "Java - Algoritma Perulangan," Daisama Bali, 13 February 2019. [Online]. Available: https://daismabali.com/artikel_detail/8/1/Java---Algoritma-Perulangan.html. [Accessed 7 November 2025].

[6] R. Yunanto, "Perulangan "While"," [Online]. Available: https://repository.unikom.ac.id/35915/1/Modul.Bahasa.C.2010.v3%20Bab08.pdf. [Accessed 7 November 2025].

[7] "Java - Algoritma Perulangan," Daisama Bali, 2019. [Online].