LAPORAN PRAKTIKUM

PEMROGRAMAN ALGORITMA DAN PEMROGRAMAN

“PERULANGAN WHILE & DO WHILE DAN

SENTINEL LOOP”

Disusun Oleh :

Dzhillan Dzhalila

2511531001

Dosen Pengampu:

Dr. Wahyudi, S.T, M.T

Asisten Praktikum:

Aufan Taufiqurrahman

FAKULTAS TEKNOLOGI IN*FOR*MASI

DEPARTEMEN IN*FOR*MATIKA

UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji syukur penulis panjatkan atas kehadirat Allah SWT 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-6 untuk mata kuliah Praktikum Algoritma Pemrograman. Praktikum ini berfokus pada pembahasan konsep Perulangan *While* (*While Loop*) dan Perulangan *Do While* (*Do While loop*) serta juga membahas *Sentinel Loop* yang ada pada bahasa pemrogragraman Java.

Melalui praktikum ini, hasil yang diharapkan adalah penulis dapat memahami dan mengimplementasikan konsep dasar Pengulangan *While*, Pengulangan *Do While* dan *Sentinel Loop* dalam bahasa Java. Melalui projek projek sederhana yang dilakukan pada praktikum ini membantu penulis untuk memahami bagaimana pemilihan dan penggunaan masing masing perulangan sehingga projek yang dihasilkan dapat mencapai keefektifan yang diinginkan.

Penulis menyadari bahwa, laporan praktikum ini masih memiliki banyak kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan dan pengembangan di masa mendatang. Semoga laporan ini dapat memberikan manfaat, baik sebagai bahan pembelajaran penulis maupun referensi pembaca dalam memahami dasar-dasar pemrograman komputer.

Padang, 2025

Dzhillan Dzhalila

251531001

DAFTAR ISI

Contents

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam pemrograman terkhusus pemograman dengan bahasa Java, kemampuan mengontrol alur eksekusi program merupakan salah satu fondasi utama dalam membangun solusi yang efektif dan efisien. Salah satu aspek penting dalam pengendalian alur tersebut adalah penggunanan struktur perulangan. Setelah mempelajari perulangan *For*, kita akan lanjut memperlajari struktur pengulangan lain yaitu perulangan *while, Do While* dan untuk menunjang pemahaman kita juga akan membahas konsep dari *sentinel loop*. Konsep konsep ini tidak hanya menghindarkan programmer dari penulisan kode yang berulang secara manual, tetapi juga memungkinkan penanganan data atau input yang bersifat dinamin dan tidak pasti jumlahnya.

Perulangan *While* akan digunakan saat kita tidak tahu berapa kali perlu perulangan perlu dijalankan. Hal ini dikarenakan eksekusi blok kode perluangan *while* akan terus berjalan selagi suatu kondisi masih terpenuhi. Sedangkan untuk perulangan *Do While*, menjamin bahwa blok kode akan dijalankan minimal satu kali sebelum pengecekan kondisi. Sementara itu, *sentinel loop* merupakan penerapan perulangan yang dikendalikan oleh nilai penanda yang umum digunakan dalam skenario interaktif seperti penerimaan input dari pengguna hingga suatu niali khusus dimasukkan sebagai sinyal pengehentian.

Dengan mempraktikkan konsep-konsep ini, kita diharapkan mampu memilih dan menerapkan struktur perulangan yang paling sesuai dengan permasalahan yang dihadapi, sehingga dapat mengahasilkan program yang lebih efektif, terstruktur dan adaptif terhadap berbagai kondisi input.

1.2 Tujuan

1. Memahami konsep dasar perulangan *while, Do While,* serta memahami konsep *sentinel loop* dalam bahasa pemrograman Java.
2. Mampu mengimplementasikan dan mengidentifikasi penggunaan konsep perulangan *while* dan *Do While* dalam kode program sederhana hingga pada projek dengan level yang lebih rumit.
3. Mampu menganalisis suatu program yang menggunakan konsep *sentinel loop* dan menerapkan konsep tersebut pada projek sederhana.
4. Mampu mengembangkan pemahaman terhadap kontrol alur perulangan *while*, *Do While* dan *sentinel loop* pada kode program.

1.3 Manfaat

1. Memberikan pemahaman melalui pengalaman langsung bagaimana jenis jenis perulangan bekerja pada program.
2. Meningkatkan kemampuan dalam memulis kode program yang lebih efektif dan efisien.

BAB II
PEMBAHASAN

2.1 Program Perulangan *While* 1

```
perulanganWhil...  Lempardadu_25...  GamePenjumlah...  SentinelLoop_...  doWhile1_2511...

1  package pekan6_2511531001;
2
3  import java.util.Scanner;
4
5  public class perulanganWhile1_2511531001 {
6      public static void main (String [] args) {
7          int counter = 0;
8          String jawab ;
9          boolean running = true;
10         //deklarasi scanner
11         Scanner scan = new Scanner (System.in);
12         while (running) {
13             counter++;
14             System.out.println("Jumlah = "+counter);
15             System.out.print("Apakah lanjut? (Iya/Tidak)=");
16             jawab = scan.nextLine();
17             //cek jawab = tidak, perulangan berhenti
18             if (jawab.equalsIgnoreCase("tidak")){
19                 running = false;
20             }
21             }
22         System.out.println("Anda Sudah melakukan perulangan sebanyak "+counter+" kali");
23
```

Gambar 2.1 : Kode Program perulangan*While*1

2.1.1 Langkah-Langkah

Adapun Langkah-Langkah untuk membuat program ini adalah;

1. *Import Scanner methods* untuk membaca input jawaban dari pengguna dan akan disimpan dalam variabel “jawab”.
2. Menginisialisasi variabel
   - “counter” dengan tipe data integer (bilangan bulat) dan set nilai variabel tersebut dengan 0.
   - “jawab” dengan tipe data String
   - “running” dengan tipe data boolean dan menyetel nilai awalnya adalah *true*
3. Menggunakan perulangan *while* dengan kondisi yang digunakan adalah *running* dan memiliki blok kode yang akan diulang jika nilai kondisi bernilai benar seperti yang ada pada gambar 2.1.
4. Menggunakan percabangan *single if* dengan *kondisinya* menggunakan *method equalseIgnoreCase*(“tidak”) terhadap

variabel jawab. Dan *statement* dari blok kode percabangan ini adalah nilai *running = false.*

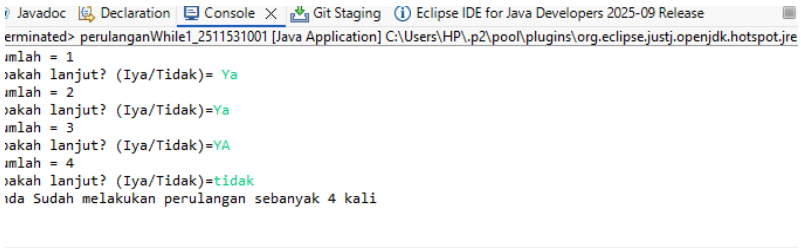
5. Mencetak kalimat sesuai dengan *line* ke-22 di gambar 2.1, jika nilai percabangan bernilai *true.*

2.1.2 Penjelasan Program

Alur kerja dari program ini adalah saat pengguna terus menjawab “ya” pada pertanyaan yang diujikan, jumlah bilangan pada variabel “counter” akan terus bertambah. Namun, saat pengguna menjawab “Tidak” maka perulangan akan berhenti dan mencetak jumlah pengulangan yang telah dilakukan sebelum akhirnya pengguna menjawab “Tidak”. Inilah peran dari konsep perulanagn *while* yang kita gunakan pada program ini. Saat kondisi perulangan terpenuhi maka akan terjadi penambahan pada variabel “counter” itu disebabkan karena blok kode “counter++”. Setelah itu, jalannya program tergantung dari *inputan* jawaban dari pengguna. Jika pengguna memasukkan jawaban “Ya”, maka perulangan akan terus terjadi, dan variabel “counter” nilainya akan terus bertambah.

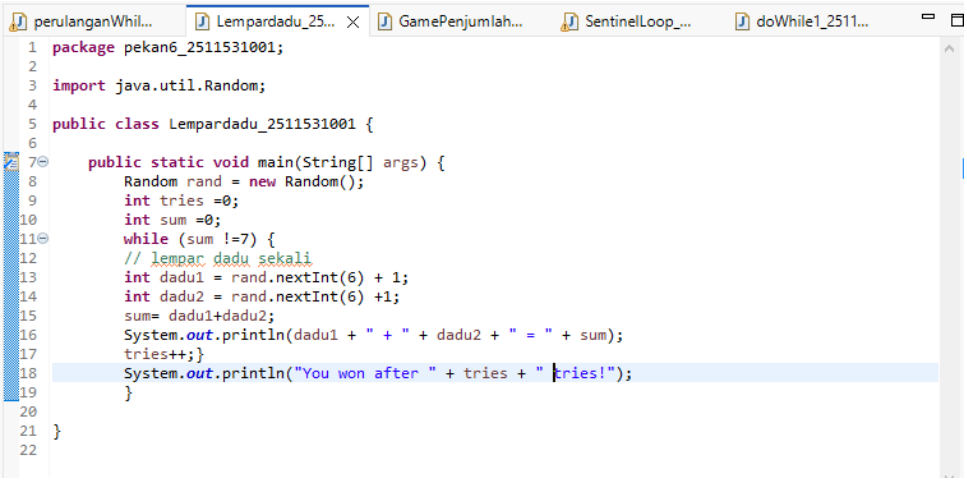
Pada program ini, kita juga menggunakan percabangan. Percabangan ini masih berada dalam perulangan *while* yang kita gunakan. Apabila sang penggunan sudah memberikan jawaban “Tidak” maka percabangan akan bernilai *true* dan *running = false* akan diesksekusi. Sehingga perulangan akan berhenti dan memberikan ouput yang menyatakan kepada pengguna jumlah perulangan atau jumlah pengguna menjawab “Ya” pada program tersebut. Pada percabangan dalam perulangan di program ini kita juga menggunakan salah satu dari *method string* yaitu *equalsIgnoreCase.* Penggunaan dari *method* ini memungkinkan bahwa blok kode percabangan *if* akan membuat semua kemungkinan input jawab “Tidak” itu bernilai benar. Contohnya; jika kita menjawab “tidak” dengan huruf “t” tidak kapital, maka percabangan tetap akan bernilai *true*. Hal ini dikarenakan *method string* yaitu *equalsIgnoreCase* akan

membaca inputan user dengan huruf kecil. Sehingga selama kata yang dijawab oleh pengguna adalah “tidak” dengan berbagai variasinya, percabangan akan tetap bernilai *true*. Adapun hasil output dari program ini adalah sebagai berikut



Gambar 2.2 : Output Program LemparDadu

2.2 Program Lempar Dadu



Gambar 2.3 : Kode Program LemparDadu

2.2.1 Langkah-Langkah

Adapun Langkah-Langkah untuk membuat program ini adalah;

1. *Import classRandom* dari *package java.util.Random* untuk menghasilkan angka acak yang akan digunakan sebagai nilai dadu dalam permainan.

2. Menginisialisasi variabel:

   - "rand" dengan tipe data *Random* untuk membuat objek generator angka acak.
   - "sum" dengan tipe data int (bilangan bulat) dan di-set nilainya awalnya 0 digunakan untuk menyimpan jumlah dua dadu.
   - "tries" dengan tipe data int dan di-set nilainya awalnya 0 digunakan untuk menghitung jumlah percobaan hingga mendapatkan hasil 7.

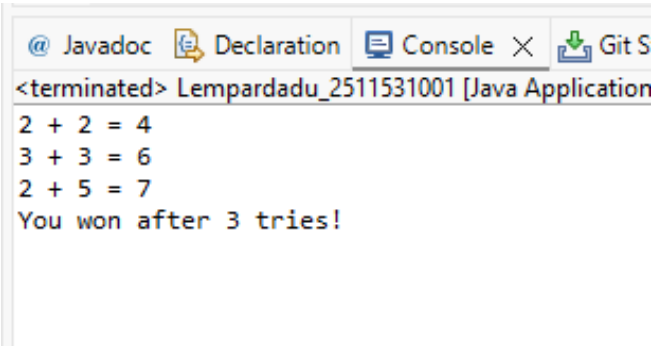
3. Menggunakan perulangan *while* dengan kondisi sum != 7, yang artinya blok kode di dalamnya akan terus diulang selama nilai sum belum sama dengan 7.

4. Di dalam blok perulangan *while*:

   - Membuat dua variabel dadu1 dan dadu2 bertipe int, masing-masing diisi dengan angka acak antara 1 hingga 6 menggunakan *method* nextInt(6) + 1.
   - Menghitung jumlah kedua dadu dan menyimpannya ke dalam variabel sum dengan rumus sum = dadu1 + dadu2.
   - Mencetak hasil lemparan dadu beserta jumlahnya menggunakan *System.out.println*(dadu1 + " + " + dadu2 + " = " + sum);.

- Menambahkan nilai tries sebanyak 1 setiap kali perulangan berjalan, menggunakan tries++;.

5. Setelah keluar dari perulangan *while* (ketika sum == 7), mencetak pesan akhir "*You won after* [jumlah percobaan] *tries!*" menggunakan *System.out.println("You won after " + tries + " tries!")*;.



Gambar 2.4 : Output Kode Program Program Lempar Dadu

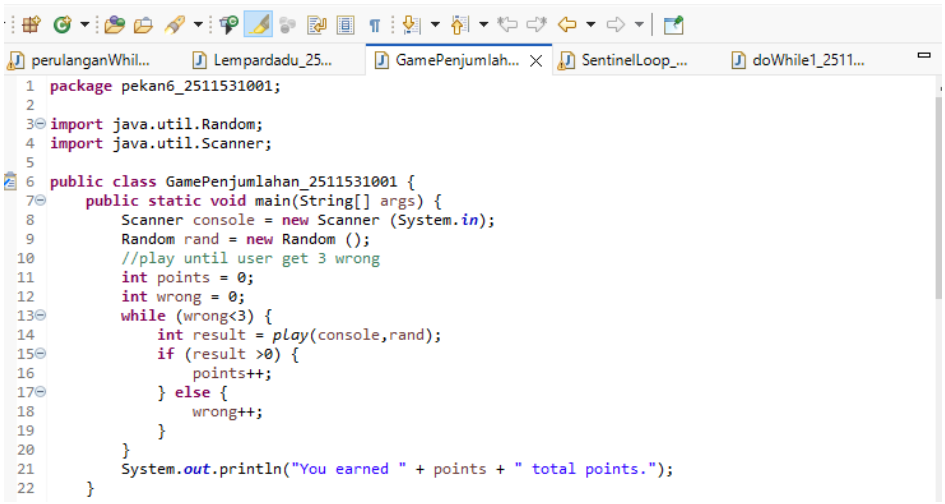
2.2.2 Penjelasan Program

Alur kerja dari program ini adalah mensimulasikan permainan lempar dua dadu secara berulang hingga jumlah kedua dadu bernilai 7. Program akan terus melakukan lemparan dadu (menghasilkan dua angka acak antara 1-6) dan menjumlahkannya. Jika jumlahnya belum 7, maka perulangan akan terus berjalan dan mencatat jumlah percobaan yang telah dilakukan. Ketika akhirnya jumlah dua dadu sama dengan 7, perulangan akan berhenti dan program akan menampilkan pesan kemenangan beserta jumlah percobaan yang diperlukan.

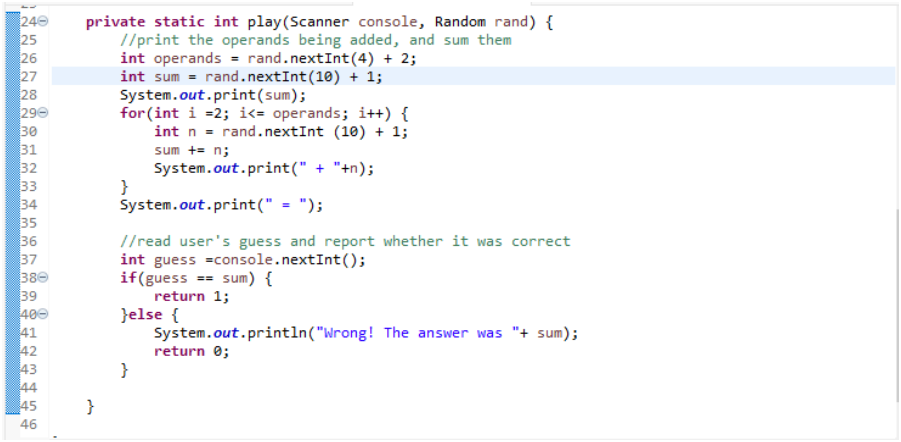
Peran utama dari konsep perulangan *while* di sini adalah untuk terus menjalankan proses lempar dadu selama kondisi sum != 7 masih terpenuhi. Setiap iterasi, nilai sum dihitung ulang, dan jika belum 7, maka tries akan bertambah. Ini memastikan bahwa program tidak akan berhenti sampai tujuan (jumlah 7) tercapai.

Pada program ini juga digunakan class *Random* untuk menghasilkan nilai acak yang realistis seperti lemparan dadu nyata. *Method* nextInt(6) menghasilkan angka dari 0 hingga 5, sehingga ditambahkan 1 agar hasilnya menjadi 1 hingga 6 — sesuai dengan jumlah mata dadu. Selain itu, penggunaan variabel tries memungkinkan kita melacak seberapa banyak usaha yang dibutuhkan untuk mencapai target, memberikan in*for*masi tambahan kepada pengguna setelah permainan selesai.

2.3 Program Game Penjumlahan



Gambar 2.5 : Kode Program GamePenjumlahan (1)



Gambar 2.6 : Kode Program GamePenjumlahan (2)

2.3.1 Langkah-Langkah

1. *Import classRandom* dari package java.util.*Random* untuk menghasilkan angka acak sebagai *operand* dalam soal penjumlahan, dan *Import classScanner* dari *package java.util.Scanner* untuk membaca input jawaban pengguna.
2. Menginisialisasi variabel:
   - "console" dengan tipe data *Scanner* untuk membaca input dari pengguna melalui System.in.
   - "rand" dengan tipe data *Random* untuk menghasilkan angka acak.
   - "points" dengan tipe data int, di-set nilainya awalnya 0 digunakan untuk menyimpan skor pemain.
   - "wrong" dengan tipe data int, di-set nilainya awalnya 0 digunakan untuk menghitung jumlah jawaban salah.
3. Menggunakan perulangan *while* dengan kondisi wrong < 3, artinya blok kode di dalamnya akan terus diulang selama jumlah kesalahan belum mencapai 3 kali.
4. Di dalam blok perulangan *while*:
   - Memanggil *method* play(console, rand) yang akan menghasilkan soal penjumlahan acak dan meminta pengguna menjawab.
   - Menyimpan nilai return dari *method* play(...) ke dalam variabel result.
     Jika result > 0 (berarti jawaban benar), maka nilai points ditambah 1 menggunakan points++.
   - Jika result == 0 (berarti jawaban salah), maka nilai wrong ditambah 1 menggunakan wrong++.
5. Setelah keluar dari perulangan *while* (ketika wrong == 3), mencetak pesan akhir "You earned [jumlah poin] total points."

menggunakan *System.out.println*("You earned " + points + " total points.");.

6. *Method* play(*Scanner* console, *Random* rand) memiliki fungsi sebagai berikut:
    - Menghasilkan jumlah *operand* acak antara 2 hingga 5 menggunakan rand.nextInt(4) + 2.
    - Menghasilkan angka pertama (*operand* pertama) antara 1 hingga 10 menggunakan rand.nextInt(10) + 1, lalu menyimpannya ke variabel sum.
    - Mencetak angka pertama tersebut ke layar.
    - Menggunakan perulangan *for* sebanyak *operand*s - 1 kali untuk menghasilkan *operand* berikutnya, menambahkannya ke sum, dan mencetaknya ke layar berserta tanda tambah.
    - Mencetak tanda sama dengan (=) setelah semua *operand* dicetak.
    - Membaca input pengguna menggunakan console.nextInt() dan menyimpannya ke variabel guess.

o Membandingkan guess dengan sum:

- Jika benar (guess == sum), maka *method* mengembalikan nilai 1.

- Jika salah (guess != sum), maka mencetak pesan "*Wrong! The answer was* [jawaban benar]" dan mengembalikan nilai 0.

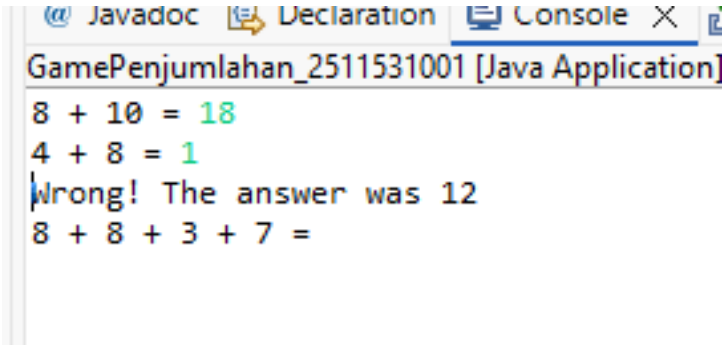
2.3.2 Penjelasan Program

Alur kerja dari program ini adalah mensimulasikan permainan matematika sederhana di mana pemain harus menjawab soal penjumlahan acak yang terdiri dari 2 hingga 5 angka. Setiap soal dibuat secara dinamis oleh program menggunakan generator angka acak. Pemain diberi kesempatan maksimal 3 kali salah — jika melebihi batas tersebut, permainan akan berhenti dan menampilkan total poin yang berhasil dikumpulkan.

Peran utama dari konsep perulangan *while* di sini adalah untuk terus menjalankan permainan selama jumlah kesalahan belum mencapai 3. Setiap kali pemain menjawab benar, poin bertambah; jika salah, counter kesalahan bertambah. Ketika counter kesalahan mencapai 3, perulangan berhenti dan program memberikan laporan akhir.

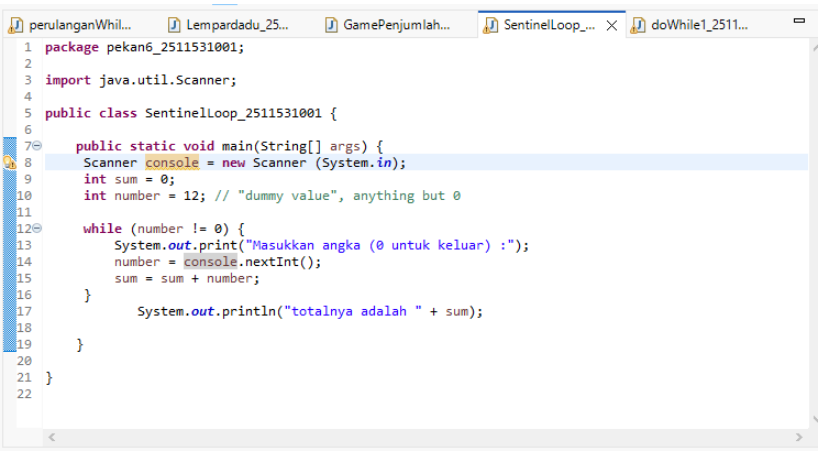
Method play(...) berfungsi sebagai modul inti permainan. Di dalamnya, program membangkitkan soal secara acak: jumlah *Scanner* (2–5), nilai tiap *Scanner* (1–10), lalu menjumlahkannya. Soal tersebut dicetak ke layar dalam bentuk string seperti "3 + 7 + 2 =", kemudian menunggu input dari pengguna. Jika jawaban benar, *method* mengembalikan nilai positif (1) yang akan menambah poin. Jika salah, *method* mengembalikan nilai 0, yang akan menambah counter kesalahan serta memberikan umpan balik langsung kepada pengguna berupa jawaban yang benar.

Penggunaan *for* loop di dalam *method* play(...) memungkinkan pencetakan soal penjumlahan yang dinamis, tanpa perlu menulis ulang kode untuk setiap jumlah *Scanner*. Selain itu, penggunaan *Scanner* dan *Random* memastikan interaksi yang responsif dan variasi soal yang tidak monoton, sehingga meningkatkan pengalaman bermain.



Gambar 2.7 : Ouput Program Game Penjumlahan

2.4 Program Sentinel Loop



Gambar 2.8 : Kode Program SentinelLoop (1)

2.4.1 Langkah-Langkah

Adapun langkah-langkah untuk membuat program ini adalah;

1. *Import classScanner* dari *package java.util.Scanner* untuk membaca input angka dari pengguna.
2. Menginisialisasi variabel:
   - "console" dengan tipe data *Scanner* untuk membaca input dari System.in.
     sum dengan tipe data int, di-set nilainya awalnya 0 digunakan untuk menyimpan total penjumlahan semua angka yang dimasukkan.
   - "*number*" dengan tipe data int, di-set nilainya awalnya 12 (nilai dummy) digunakan sebagai nilai awal agar perulangan while dapat berjalan pertama kali.
3. Menggunakan perulangan while dengan kondisi *number* != 0, artinya blok kode di dalamnya akan terus diulang selama nilai *number* belum sama dengan 0.
4. Di dalam blok perulangan while:

- Mencetak pesan “Masukkan angka (0 untuk keluar) :” ke layar menggunakan *System.out.print*(...).
- Membaca input angka dari pengguna menggunakan console.nextInt() dan menyimpannya ke variabel *number*.
- Menambahkan nilai *number* ke dalam variabel sum menggunakan sum = sum + *number*.

5. Setelah keluar dari perulangan while (ketika pengguna memasukkan angka 0), mencetak pesan akhir “totalnya adalah [jumlah total]” menggunakan *System.out.println*("totalnya adalah " + sum);.

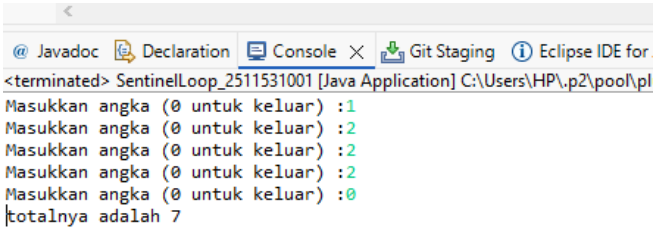
2.4.2 Penjelasan Program

Alur kerja dari program ini adalah menghitung total penjumlahan dari sejumlah angka yang dimasukkan oleh pengguna, hingga pengguna memasukkan angka 0 yang berfungsi sebagai sentinel value atau nilai penghenti perulangan.

Peran utama dari konsep perulangan while di sini adalah untuk terus menerima input angka dari pengguna selama nilai yang dimasukkan bukan 0. Nilai 0 tidak dijumlahkan ke dalam total, melainkan hanya berfungsi sebagai sinyal untuk menghentikan perulangan. Ini merupakan contoh klasik penggunaan *sentinel-controlled loop*, di mana perulangan dikendalikan oleh nilai khusus (sentinel) yang diberikan oleh pengguna.

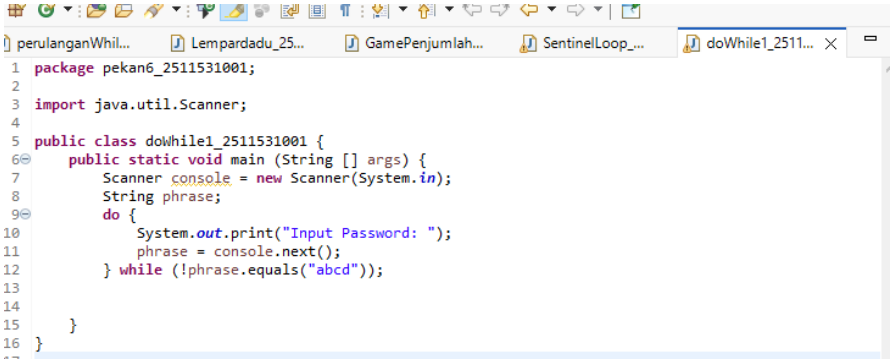
Variabel *number* diinisialisasi dengan nilai 12 (atau nilai apa pun selain 0) agar perulangan while dapat berjalan minimal satu kali karena kondisi *number* != 0 akan bernilai *true* pada iterasi pertama. Setiap kali pengguna memasukkan angka (kecuali 0), nilai tersebut ditambahkan ke sum. Proses ini terus berulang sampai pengguna memasukkan 0, lalu program menampilkan total keseluruhan.

Adapun untuk output dari program ini adalah:



Gambar 2.9 : Ouput Program Sentinel Loop

2.5 Program do*While*



Gambar 2.10 : Kode Program do*While*

2.5.1 Langkah-Langkah

Adapun langkah-langkah untuk membuat program ini adalah;

1. *Import class Scanner* dari *package java.util.Scanner* untuk membaca input password dari pengguna.
2. Menginisialisasi variabel:
   - "console" dengan tipe data *Scanner* untuk membaca input dari System.in.
   - "phrase" dengan tipe data String digunakan untuk menyimpan input password yang dimasukkan oleh pengguna.
3. Menggunakan perulangan *do-while* yang memiliki blok kode di dalam kurung kurawal { } dan kondisi di akhir while (!*phrase.equals*("abcd")).

Artinya: blok kode akan dijalankan minimal satu kali, lalu akan terus diulang selama nilai *phrase* tidak sama dengan "abcd".

4. Di dalam blok do:

   - Mencetak pesan “Input Password: ” ke layar menggunakan *System.out.print*(...).
   - Membaca input string dari pengguna menggunakan console.next() dan menyimpannya ke variabel *phrase*.

5. Setelah keluar dari perulangan *do-while* (ketika *phrase* bernilai "abcd"), program akan berhenti secara otomatis karena tidak ada pernyataan atau output tambahan setelah blok prulangan.

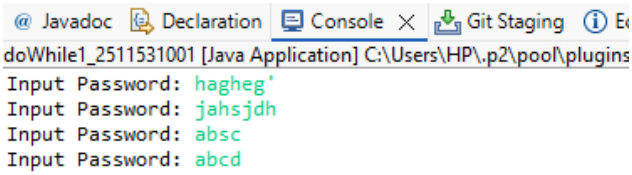
2.5.2 Penjelasan Program

Alur kerja dari program ini adalah meminta pengguna memasukkan password hingga password yang dimasukkan benar, yaitu "abcd". Program akan terus menampilkan pesan “Input Password: ” dan menunggu input dari pengguna minimal sekali, bahkan jika pengguna langsung memasukkan password yang benar sejak awal.

Peran utama dari konsep perulangan *do-while* di sini adalah untuk memastikan bahwa blok kode di dalamnya selalu dijalankan paling sedikit satu kali, sebelum kondisi perulangan diperiksa. Ini sangat berguna dalam kasus seperti autentikasi atau login, di mana kita ingin meminta input dari pengguna setidaknya satu kali, baru kemudian memeriksa apakah input tersebut valid.

Pada program ini, kondisi !*phrase*.*equals*("abcd") berarti: “selama *phrase* tidak sama dengan "abcd", maka ulangi”. Jika pengguna memasukkan "abcd", maka kondisi menjadi *false*, dan perulangan berhenti. Penggunaan *method* equals(...) pada tipe data String memastikan perbandingan dilakukan berdasarkan isi string, bukan referensi objek.

Adapun output dari program ini adalah



Gambar 2.11 : Ouput program *Do While*

BAB III

KESIMPULAN

Berdasarkan praktikum pertemuan minggu ke-6, kita dapat menyimpulkan bahwa strukutr perulangan *while, Do While* dan *sentinel loop* merupakan konsep yang sangat penting dalam pemrograman karena membawa banyak kemudahan bagi programmer untuk mennghasilkan program yang efektif dan efisien. Dengan penggunaan ketiga konsep yang telah dipelajari dalam pertemuan ini kita dapat mengeksekusi blok kode secara berulang berdasarkan kondisi tertentu tanpa membuat kode blok yang sama berulang-ulang.

Perulangan *while* adalah konsep yang tepat digunakan saat kita sebagai programmer tidak mengetahui jumlah literasi yang diperlukan untuk perulangan eksekusi suatu kode blok program. Kosep perulangan *while* ini juga akan memastikan semua kode blok yang akan dieksekusi adalah kode yang memenuhi kondisi yang ada, dengan cara melakukan pengecekan kondisi di awal. Hal ini menyebabkan adanya kemungkinan blok perulangan tidak dijalankan sama sekali jika kondisi bernilai salah. Sebaliknya, perulangan *Do While* menjamin setidaknya satu kali eksekusi blok kode karena pengecekan kondisi dilakukan di akhir literasi.

Sementara itu *sentinel loop* merupakan teknik perulangan berbasis nilai penanda (*sentinel value)* yang digunakan untuk menghentikan eksekusi perulangan. Konsep ini umumnya digunakan saat input pengguna tidak dapat diprediksi jumlahnya.

Secara keseluruhan, pemahaman terhadap ketiga jenis perulangan ini memungkinakan programmer menulis kode ayang lebih efisien, fleksibel. Dan sesuai dengan kebutuhan logika program. Pemilihan jenis perulangan yang tepat akan berdampak langsung terhadap kejelasan, keandalan, dan per*for*ma program.

DAFTAR PUSTAKA

[1] Duniailkom, “Perulangan While dan Do-While dalam Bahasa Java,” *Duniailkom,* 2023. [Daring]. Tersedia: https://www.duniailkom.com/tutorial-java-perulangan-while-dan-do-while/ . [Diakses: 6 Nov 2025].

[2] Duniailkom, “Cara Menggunakan Method equals() dan equalsIgnoreCase() pada String Java,” *Duniailkom*, 2022. [Daring]. Tersedia: https://www.duniailkom.com/tutorial-java-string-method-equals-equalsignorecase/ . [Diakses: 6 Nov 2025].

[7] Codepolitan, “Pengertian dan Contoh Sentinel Value dalam Pemrograman,” *Codepolitan*, 2023. [Daring]. Tersedia: https://www.codepolitan.com/pengertian-sentinel-value-dalam-pemrograman/ . [Diakses: 6 Nov 2025].