LAPORAN PRATIKUM

ALGORITMA PEMROGRAMAN DAN PEMROGRAMAN

PERULANGAN DO DAN DO - WHILE

DISUSUN OLEH:

Rayya Syaqinah Putri Hasibuan

2511532007

DOSEN PENGAMPU:

Dr. Wahyudi, S.T. M.T

ASISTEN PRATIKUM:

Aufan Taufiqurrahman

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga laporan praktikum ini dapat diselesaikan dengan baik. Laporan ini disusun sebagai salah satu bentuk pertanggungjawaban pelaksanaan praktikum Algoritma Pemrograman dengan judul Perulangan *While dan Do-While*. Laporan ini bertujuan untuk mendokumentasikan prosedur, hasil, serta pembahasan dari kegiatan praktikum, sekaligus sebagai sarana pembelajaran bagi penulis untuk lebih memahami materi terkait.

Penulis menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan demi penyempurnaan laporan di masa mendatang. Akhir kata, penulis mengucapkan terima kasih kepada dosen pengampu, asisten pratikum, serta semua pihak yang telah membantu sehingga laporan ini dapat terselesaikan.

Padang, 03 November 2025

Rayya Syaqinah Putri Hasibuan

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam JavaScript, perulangan (*looping*) digunakan untuk menjalankan kode secara berulang hingga kondisi tertentu terpenuhi. Dua jenis perulangan yang sering dipakai adalah *while* dan *do-while*. Perulangan *while* mengecek kondisi sebelum menjalankan blok kode, sedangkan *do-while* mengeksekusi kode terlebih dahulu minimal satu kali sebelum memeriksa kondisi. Pemahaman terhadap keduanya penting agar programmer dapat menulis kode yang efisien, logis, dan sesuai kebutuhan program.

1.2 Tujuan Pratikum

Tujuan praktikum perulangan *while* dan *do-while* pada JavaScript adalah untuk memahami perbedaan cara kerja kedua struktur perulangan tersebut, melatih kemampuan dalam menentukan kondisi yang tepat agar program berjalan sesuai logika, serta mengasah keterampilan menulis kode yang efisien dan terhindar dari kesalahan seperti *infinite loop*, sehingga peserta mampu menerapkan konsep perulangan secara efektif dalam berbagai kasus pemrograman.

1.3 Manfaat Pratikum

Praktikum ini memberikan pengalaman langsung dalam memahami cara kerja perulangan *while* dan *do-while* pada JavaScript. Melalui penerapan kode secara langsung, peserta dapat melihat bagaimana kedua jenis perulangan bereaksi terhadap kondisi yang berbeda, serta bagaimana menghindari kesalahan seperti *infinite loop*. Selain itu, praktikum ini membantu meningkatkan kemampuan berpikir logis dan analitis dalam menentukan kondisi perulangan yang tepat. Dengan demikian, peserta dapat mengembangkan kemampuan menulis program yang efektif, efisien, dan sesuai dengan standar logika pemrograman yang baik.

BAB II
PEMBAHASAN

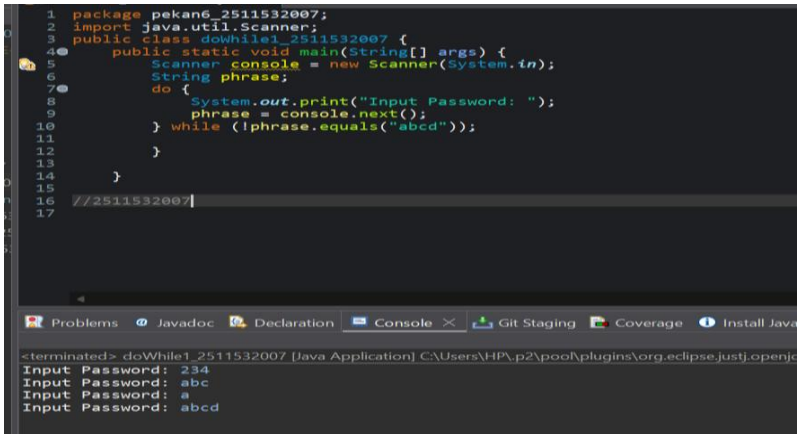
2.1 Deskripsi Materi Pratikum

Perulangan while: Iterasi suatu kondisi dapat dilakukan dengan bantuan pernyataan while. Perulangan akan terus berjalan selama kondisinya benar; jika salah, perulangan akan berakhir.

Perulangan do-while dan while serupa, tetapi perulangan do-while mengharuskan proses dijalankan setidaknya sekali.

Perulangan sentinel adalah jenis pola perulangan yang menghentikan eksekusi dengan menggunakan nilai sentinel, yang merupakan nilai penanda unik. Biasanya, nilai ini misalnya -1 untuk input positif tidak disertakan dalam data yang valid. Perulangan sentinel berguna untuk memproses data hingga kondisi tertentu terpenuhi tanpa mengetahui jumlah iterasi sebelumnya. Perulangan ini sering kali menggunakan perulangan while atau do-while sebagai dasarnya.

2.2 Do While 1



Gambar 2.2 kode program dan output ”doWhile1”

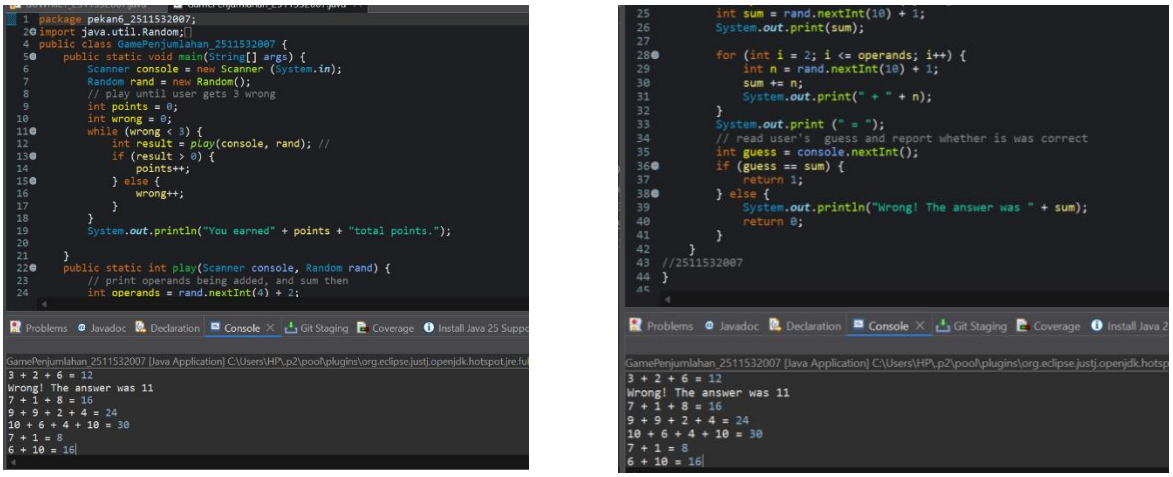
Langkah pengerjaan:

1. Membuat package pekan6 dan *new class* untuk pemrograman doWhile1 dengan nama “doWhile1_2511532007”.
2. Eksekusi program dimulai dari method *public static void main(String[] args).*

3. Menginput *Scanner console = new Scanner(System.in);* membuat objek Scanner untuk menerima input pengguna.
4. Menginput *String phrase;* mendeklarasikan variabel untuk menyimpan input password.
5. Bagian *do { ... } while (...);* adalah struktur perulangan do-while, yang akan menjalankan perintah di dalam {} minimal satu kali.
6. Menginput *System.out.print("Input Password: ");* menampilkan pesan agar pengguna memasukkan password.
7. Inputan pada *phrase = console.next();* untuk menyimpan input pengguna ke dalam variabel *phrase.*
8. Kondisi !*phrase.equals("abcd")* berarti selama password tidak sama dengan "abcd", program akan terus mengulang. Jika password benar (“abcd”), perulangan berhenti.
9. *Running* program.
10. Masukkan password yang salah beberapa kali, lalu masukkan “abcd” untuk menghentikan program. Dan hasil output dapat dilihat pada gambar 2.2

Aplikasi akan terus meminta kata sandi hingga pengguna memasukkan "abcd". Perintah akan dijalankan setidaknya sekali karena menggunakan perulangan do-while, meskipun kondisi awalnya salah. Hal ini menunjukkan bahwa perulangan do-while berfungsi dengan baik dalam skenario seperti login atau validasi kata sandi di mana pengguna harus memasukkan setidaknya satu input sebelum kondisi diuji.

2.3 Game Penjumlahan



```
package pekan6_2511532007;
import java.util.Random;
public class GamePenjumlahan_2511532007 {
    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); //
            if (result > 0) {
                points++;
            } else {
                wrong++;
            }
        }
        System.out.println("You earned" + points + "total points.");

    }
    public static int play(Scanner console, Random rand) {
        // print operands being added, and sum then
        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 (" = ");
        // read user's  guess and report whether is was correct
        int guess = console.nextInt();
        if (guess == sum) {
            return 1;
        } else {
            System.out.println("Wrong! The answer was " + sum);
            return 0;
        }
    }
//2511532007
}
```

```
3 + 2 + 6 = 12
Wrong! The answer was 11
7 + 1 + 8 = 16
9 + 9 + 2 + 4 = 24
10 + 6 + 4 + 10 = 30
7 + 1 = 8
6 + 10 = 16
```

Gambar 2.3 kode program dan output ”Game Penjumlahan”.

Langkah pengerjaan:

1. Membuat package pekan6 dan *new class* untuk pemrograman Game Penjumlahan dengan nama “GamePenjumlahan_2511532007”.
2. Eksekusi program dimulai dari method *public static void main(String[] args).*
3. Menginisialisasi *int points* = 0 dan *int wrong* = 0
4. Menginput *import java.util.Random;* yang digunakan untuk menghasilkan angka acak dan *import java.util.Scanner;* digunakan agar program dapat menerima input dari pengguna
5. Penginputan perulangan *while (wrong < 3)* membuat permainan terus berjalan sampai pengguna salah tiga kali.
6. Penginputan *play()* di setiap putaran guna untuk menjalankan dan menampilkan soal penjumlahan acak.
7. *Run* program
8. Dari hasil kode program ini ketika di *run* akan menghasilkan output dengan menghasilkan angka penjumlahan acak untuk dimainkan seperti pada gambar 2.3.

Untuk analisis hasil, program ini adalah permainan penjumlahan sederhana yang menggunakan loop while untuk mengatur jumlah kesempatan pemain. Setiap soal dibuat secara acak dengan bantuan Random, sehingga setiap percobaan akan menghasilkan kombinasi angka berbeda. Ketika pengguna menjawab benar, skor

bertambah satu; jika salah, jumlah kesalahan naik satu. Setelah tiga kali salah, permainan berakhir.

2.4 Lempar Dadu

```
import java.util. Random;
public class LemparDadu_2515532007 {

    public static void main(String[] args) {
        Random rand = new Random();
        int tries = 0;
        int sum = 0;
        while (sum != 7) {
            // roll the dice once
            int dadu1 = rand.nextInt(6) + 1;
            int dadu2 = rand.nextInt(6) + 1;
            sum = dadu1 + dadu2;
            System.out.println (dadu1 + " + " + dadu2  + " = "+ sum);
            tries++;
        }
        System.out.println("You won after " + tries + " tries!");
    }
//2511532007
}

Javadoc
Problems  Declaration  Console  Git Staging  Coverage  Install Java 25 Support  Eclipse IDE for Java Deve
<terminated> LemparDadu_2515532007 [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x8
3 + 2 = 5
6 + 3 = 9
5 + 4 = 9
2 + 5 = 7
You won after 4 tries!
```

Gambar 2.4 kode program dan output ”LemparDadu”.

Langkah pengerjaan:

1. Membuat package pekan6 dan *new class* untuk pemrograman Lempar Dadu dengan nama “LemparDadu_2511532007”.
2. Eksekusi program dimulai dari method *public static void main(String[] args).*
3. Menginput import java.util.Random; yang digunakan untuk menghasilkan angka acak pada dadu.
4. Menginisialisasi variabel *int tries* = 0 untuk menghitung jumlah percobaan dan int sum = 0 untuk menyimpan hasil penjumlahan dua dadu.
5. Menggunakan perulangan *while* (sum != 7) agar program terus melempar dadu sampai hasil kedua dadu berjumlah 7.
6. Di dalam perulangan, dua dadu diacak menggunakan *rand.nextInt(6) + 1*, kemudian hasilnya dijumlah dan ditampilkan di layar.
7. Setelah hasil penjumlahan kedua dadu sama dengan 7, perulangan berhenti dan program menampilkan jumlah percobaan dengan teks “*You won after ... tries*!”.
8. *Run* program untuk melihat hasil simulasi lempar dadu seperti pada tampilan output pada gambar 2.4.

Program ini menunjukkan penggunaan perulangan while yang bergantung pada kondisi acak, serta memperlihatkan bagaimana pengulangan berhenti saat kondisi tertentu tercapai.

2.5 Perulangan While 1

```java
package pekan6_2511532007;
import java.util.Scanner;
public class PerulanganWhile1_2515532007 {
    public static void main(String[] args) {
        int counter=0;
        String jawab;
        boolean running = true;
        //deklarasi scanner
        Scanner scan = new Scanner(System.in);
        while (running) {
            counter++;
            System.out.println("Jumlah = "+counter);
            System.out.print("Apakah lanjut (ya/tidak?)");
            jawab= scan.nextLine();
            //cek jawab = tidak, perulangan berheti
            if (jawab.equalsIgnoreCase("tidak")) {
                running = false;
            }

        }

        System.out.println("Anda sudah melakukan perulangan sebanyak "+counter+" kali");
    }
```

Problems | Declaration | Console | Git Staging | Coverage | Install Java 25 Support | Eclipse IDE for Java Developers 202

<terminated> PerulanganWhile1_2515532007 [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

```
Jumlah = 1
Apakah lanjut (ya/tidak?)ya
Jumlah = 2
Apakah lanjut (ya/tidak?)tidak
Anda sudah melakukan perulangan sebanyak 2 kali
```

Gambar 2.5 kode program dan output ”PerulanganWhile1”

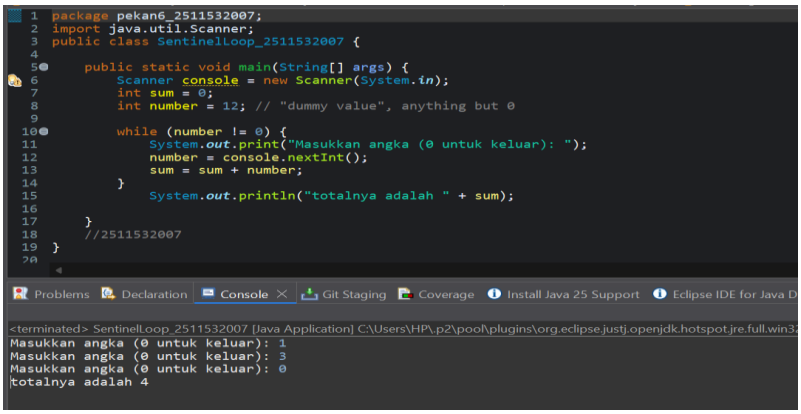
Langkah pengerjaan:

1. Membuat *package* ”pekan6_2511532007” dan *new class* dengan nama “PerulanganWhile1_2515532007”.
2. Eksekusi program dimulai dari method *public static void main(String[] args).*
3. Menginput import *java.util.Scanner*; agar program bisa membaca input dari pengguna lewat keyboard.
4. Mendeklarasikan variabel int counter = 0 untuk menghitung berapa kali perulangan dijalankan.
5. Mendeklarasikan variabel *boolean running = true* sebagai penanda apakah perulangan masih berjalan atau tidak.
6. Membuat objek *Scanner scan = new Scanner(System.in);* untuk menerima input pengguna.
7. Menggunakan *while (running)* agar perulangan terus dijalankan selama nilai *running* bernilai *true.*

8. Di dalam perulangan, program akan menampilkan teks “Jumlah = ...”, lalu bertanya “Apakah lanjut (ya/tidak?)” untuk meminta keputusan pengguna.
9. Jika pengguna mengetik “tidak”, maka kondisi *if (jawab.equalsIgnoreCase("tidak"))* akan mengubah nilai running menjadi false, sehingga perulangan berhenti.
10. Setelah keluar dari loop, program menampilkan total berapa kali pengguna melakukan perulangan dengan kalimat “Anda sudah melakukan perulangan sebanyak ... kali”.
11. *Run* program untuk melihat hasil output.

Program ini akan terus mengulang dan menampilkan jumlah perulangan selama pengguna menjawab “ya”. Begitu pengguna mengetik “tidak”, program langsung berhenti dan menampilkan total berapa kali perulangan dilakukan.

2.6 Sentinel Loop



Gambar 2.6 adalah kode program dan hasil output dari program ”SentinelLoop”.

Langkah pengerjaan:

1. Membuat *package* ”pekan6_2511532007” dan *new class* dengan nama “SentinelLoop_2511532007”.
2. Eksekusi program dimulai dari method *public static void main(String[] args).*
3. Menginput *import java.util.Scanner*; agar program dapat membaca input angka dari pengguna.

4. Mendeklarasikan variabel int sum = 0 untuk menyimpan total penjumlahan angka yang dimasukkan.
5. Mendeklarasikan variabel *int number = 12*; sebagai nilai awal sementara (*dummy value*) agar kondisi perulangan dapat dimulai. Nilai ini bisa berupa angka apa pun selain 0.
6. Menggunakan perulangan *while (number != 0)* sehingga program akan terus meminta input angka selama pengguna belum memasukkan angka 0.
7. Di dalam loop, program menampilkan teks "Masukkan angka (0 untuk keluar):", lalu membaca input pengguna menggunakan *console.nextInt()*.
8. Setiap angka yang dimasukkan akan ditambahkan ke variabel sum dengan perintah *sum = sum + number;*.
9. Jika pengguna memasukkan 0, kondisi perulangan tidak terpenuhi lagi dan loop berhenti.
10. Setelah keluar dari perulangan, program menampilkan total hasil penjumlahan dengan kalimat "totalnya adalah ...".
11. *Run* program untuk melihat hasilnya yang dapat juga dilihat pada gambar 2.6.

Program ini memperlihatkan bagaimana perulangan *while* digunakan untuk menjumlahkan data secara terus-menerus hingga pengguna memberi sinyal untuk berhenti (sentinel). Konsep ini sering digunakan dalam pemrosesan data yang jumlah inputnya tidak diketahui di awal.

BAB III
KESIMPULAN DAN SARAN

3.1 KESIMPULAN

Dari hasil praktikum perulangan menggunakan struktur *while, do-while*, dan sentinel loop, dapat disimpulkan bahwa setiap bentuk perulangan memiliki peran dan karakteristik yang berbeda dalam mengelola alur program. *Do-while* memastikan perulangan dijalankan setidaknya sekali, sentinel loop menggunakan nilai penanda untuk menghentikan proses masukan, dan *while loop* digunakan ketika jumlah pengulangan tidak pasti dan bergantung pada suatu kondisi. Eksperimen ini memperjelas logika kontrol, kriteria penghentian, dan hubungan antara masukan pengguna dan proses perulangan.

3.2 SARAN

Dalam pembuatan program yang menggunakan perulangan, sebaiknya perhatikan kondisi berhenti (loop control) agar tidak terjadi *infinite loop* yang membuat program berjalan tanpa henti. Selain itu, gunakan jenis perulangan yang paling sesuai dengan kebutuhan logika program—misalnya do-while untuk proses input yang harus dijalankan minimal sekali, dan while untuk pengulangan yang bergantung pada keputusan pengguna. Disarankan juga untuk selalu menambahkan pesan yang informatif pada output agar pengguna lebih mudahmemahami jalannya program.

DAFTAR PUSTAKA

[1] N. L. P. S. Adnyani, "Rancang Bangun Aplikasi Pembelajaran Dasar Pemrograman Berbasis Mobile Phone," Universitas Udayana, 2016. [Online]. Available: http://download.garuda.kemdikbud.go.id/article.php?article=809400&val=13211&title=RANCANG%20BANGUN%20APLIKASI%20PEMBELAJARAN%20DASAR%20PEMROGRAMAN%20BERBASIS%20MOBILE%20PHONE. . [Diakses: 04-Nov-2025].

[2] D. Cajic, “Beginner Java Exercise: Sentinel Values and Do-While Loops,” *dino cajic blog*, 13 Jun. 2015. [Online]. Available: https://dinocajic.blogspot.com/2015/06/beginner-java-exercise-sentinel-values.html. [Diakses: 04-Nov-2025].