LAPORAN PRATIKUM

PEMROGRAMAN ALGORITMA DAN PEMROGRAMAN

“PERULANGAN WHILE DAN DO WHILE”

disusun Oleh:

AFIF RAHMAN SALEH

NIM 251153105

Dosen Pengampu: Dr. WAHYUDI, S.T, M.T

Asisten Pratikum: AUFAN TAUFIQURRAHAMAN

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN

2025

DAFTAR ISI

KATA PENGANTAR

Puji syukur kami panjatkan atas kehadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya laporan praktikum ini dapat diselesaikan dengan baik. Laporan ini disusun sebagai bentuk penugasan dalam kegilatan praktikum mata kuliah Algoritma dan Pemograman khususnya pada materi " Perulangan while dan do while ".

Dalam penyusunan laporan ini, penulis berusaha untuk menjelaskan secara sistematis mulai dari latar belakang, tujuan, dan manfaat praktikum, hingga pembahasan kode program, analisis hasil, serta kesimpulan dan saran untuk pengembangan lebih lanjut. Materi yang dibahas diharapkan dapat menjadi referensi bagi mahasiswa atau praktikan dalam memahami dan mengimplementasikan algoritma perulangan secara efektif.

Padang, 2025

Penulis

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perulangan merupakan salah satu konsep dasar dalam algoritma pemrograman yang kode perintah secara berulang hingga kondisi tertentu terpenuhi. Dalam bahasa pemrograman Java, terdapat beberapa jenis perulangan, termasuk perulangan while dan do while. Perulangan while memeriksa kondisi sebelum mengeksekusi blok kode, sehingga jika kondisi awal tidak terpenuhi, blok kode tidak akan dijalankan sama sekali. Sebaliknya, perulangan do while mengeksekusi blok kode setidaknya satu kali terlebih dahulu, baru kemudian memeriksa kondisi untuk melanjutkan perulangan.

Konsep ini penting dalam pengembangan perangkat lunak karena banyak aplikasi memerlukan pemrosesan data berulang, seperti validasi input, iterasi array, atau simulasi proses berulang. Praktikum ini difokuskan pada penerapan kedua jenis perulangan tersebut dalam Java untuk memberikan pemahaman mendalam tentang perbedaannya, penggunaannya, dan potensi kesalahan yang mungkin terjadi, seperti infinite loop jika kondisi tidak dikelola dengan baik.

1.2 Tujuan Praktikum

Praktikum ini bertujuan untuk:

1. Memahami prinsip kerja perulangan while dan do while dalam bahasa pemrograman Java.
2. Mengimplementasikan kedua jenis perulangan tersebut melalui contoh kode sederhana untuk membandingkan perilaku dan penggunaannya.
3. Mengidentifikasi kelebihan dan kekurangan masing-masing perulangan serta situasi di mana salah satunya lebih sesuai digunakan.
4. Melatih kemampuan debugging dan penanganan kesalahan, seperti menghindari infinite loop melalui kontrol kondisi yang tepat.

1.3 Manfaat Praktikum

Pelaksanaan praktikum ini memberikan manfaat sebagai berikut:

1. Meningkatkan pemahaman konsep algoritma berulang, yang merupakan dasar untuk struktur kontrol lanjutan dalam pemrograman.
2. Mengembangkan keterampilan praktis dalam menulis kode Java yang efisien.
3. Membantu mahasiswa atau peserta dalam memecahkan masalah dunia nyata yang melibatkan iterasi, seperti pengolahan data besar atau simulasi sistem.
4. Mendorong pemikiran logis dan analitis melalui eksperimen dengan variasi kondisi perulangan, yang berguna untuk pengembangan aplikasi interaktif atau otomatis.

BAB II
PEMBAHASAN

2.1 Kode 1 : perulanganWhile1_2511531005

```java
import java.util.Scanner;

public class perulanganWhile1_2511531005 {

    public static void main(String[] args) {

        int counter=0;
        String jawab;
        boolean running = true ;

        Scanner scan = new Scanner (System.in);
        while (running) {
            counter++;
            System.out.println("jumlah = " + counter);
            System.out.println("apakah lanjut ( ya / tidak) ?");
            jawab  = scan.nextLine();

            if (jawab.equalsIgnoreCase("tidak")) {
                running = false;
            }
        }
        System.out.println("anda sudah melakukan perulangan sebanyak "+counter + " kali");
    }

}
```

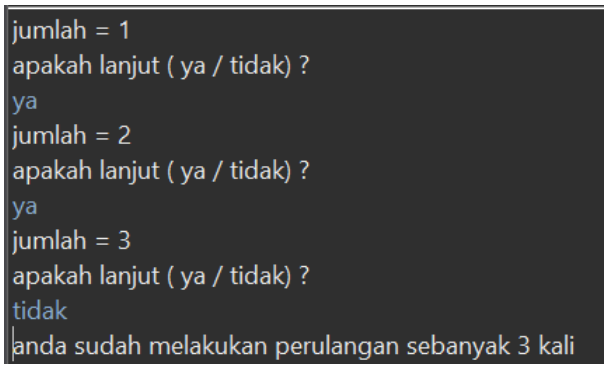
Kode Program 2.1

Kode ini menunjukan perulangan while untuk menghitung jumlah iterasi berdasarkan input pengguna. Program menggunakan variabel running sebagai kondisi untuk mengontrol perulangan, dan counter untuk menghitung putaran. Pengguna diminta konfirmasi untuk melanjutkan atau berhenti.

Langkah kerja :

1. Inisialisasi: counter = 0, running = true, dan objek Scanner untuk input.
2. Masuk ke loop while (running): Jika running true, eksekusi blok.
3. Increment counter, cetak nilai counter, dan minta input "ya/tidak".
4. Jika input "tidak", set running = false untuk keluar loop.
5. Loop berulang hingga kondisi false, lalu cetak total counter.

Output:



Output 2.1

Analisis hasil :

Program menghasilkan output berupa jumlah perulangan yang dilakukan pengguna, misalnya "anda sudah melakukan perulangan sebanyak 3 kali" jika pengguna memilih "tidak" setelah 3 kali. Ini menunjukkan kontrol kondisional dalam while, di mana loop berhenti saat kondisi tidak terpenuhi, sesuai teori bahwa while cocok untuk skenario di mana jumlah iterasi tidak diketahui sebelumnya (indefinite loop). Referensi: Dalam algoritma, ini mirip sentinel-controlled loop, di mana sentinel ("tidak") menghentikan proses, mencegah infinite loop jika kondisi dikelola baik.

2.2 Kode 2 : perulanganWhile1_2511531005

```java
import java.util.Random;

public class Lempardadu_2511531005 {

    public static void main(String[] args) {
        Random rand = new Random();
        int tries =0 ;
        int sum = 0 ;
        while (sum !=7 ) {

            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!");
    }

}
```

Kode Program 2.2

Kode ini mensimulasikan pelemparan dua dadu hingga jumlahnya mencapai 7, menggunakan perulangan while dengan kondisi sum != 7. Variabel tries menghitung jumlah percobaan.

Langkah kerja :

1. Inisialisasi: Objek Random, tries = 0, sum = 0.
2. Masuk loop while (sum != 7): Jika sum bukan 7, lanjut.
3. Buat ulang dua angka acak (1-6) untuk dadu, hitung sum, cetak hasil, dan increment tries.
4. Loop berulang hingga sum == 7, lalu cetak total tries.

Output:

```
1 + 2 = 3
5 + 1 = 6
1 + 5 = 6
2 + 5 = 7
you won after 4 tries!
```

Output 2.2

Analisis hasil :

Output menampilkan setiap pelemparan (misalnya "2 + 5 = 7") dan total percobaan (misalnya "you won after tries!"). Ini menggambarkan penggunaan while untuk simulasi probabilistik, di mana loop berjalan hingga kondisi spesifik tercapai. Referensi: Berdasarkan teori probabilitas, peluang sum=7 adalah 6/36, sehingga loop ini efisien untuk eksperimen acak.

2.3 Kode 3 : Lempardadu_2511531005

```
import java.util.Random;
import java.util.Scanner;

public class GamePenjumlahan_2511531005 {

    public static void main(String[] args) {
        Scanner console = new Scanner (System.in);
        Random rand = new Random ();

        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.");
    }
```

```java
    public static int play (Scanner console, Random rand) {
        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("=");

        int guess = console.nextInt();
        if (guess == sum) {
            return 1;
        }else {
            System.out.println("wrong! the answer was " + sum );
            return 0;
        }
    }
}
```

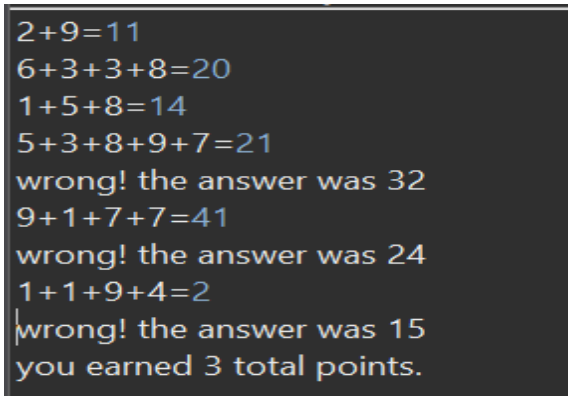
Kode Program 2.3

Kode ini membuat game penjumlahan sederhana dengan perulangan while yang berhenti jika salah 3 kali. Menggunakan method play untuk menghasilkan soal acak dan memeriksa jawaban.

Langkah kerja :

1. Inisialisasi: points = 0, wrong = 0, objek Scanner dan Random.
2. Loop while (wrong < 3): Jika wrong < 3, panggil play.
3. Dalam play: Generate operand acak (2-5), hitung sum, cetak soal, ambil input, bandingkan dengan sum.
4. Jika benar, increment points; jika salah, increment wrong dan cetak jawaban benar.
5. Loop berulang hingga wrong == 3, lalu cetak total points.

Output:



Output 2.3

Analisis hasil :

Program menghasilkan soal seperti "2+9=11" dan seterusnya. Ketika kita salah dalam menginputkan jawaban sebanyak 3 kali, maka program berhenti dan mencetak skor akhir (misalnya "you earned 3 total points."). Ini menunjukkan while untuk game loop dengan batas kesalahan, memastikan interaksi hingga kondisi tercapai. Teori: Ini adalah contoh bounded loop dengan nested logic, di mana while mengontrol sesi hingga kondisi error limit, sesuai prinsip error-handling dalam algoritma. Referensi: Mirip dengan adaptive learning systems, di mana loop berhenti berdasarkan performa, meningkatkan engagement pengguna.

2.4 Kode 4 : perulanganWhile1_2511531005

```java
import java.util.Scanner;

public class doWhile1_2511531005 {

    public static void main(String[] args) {
        Scanner console = new Scanner (System.in);
        String pharase;
        do {
            System.out.println("input password : ");
            pharase = console.next();
        }while ( !pharase.equals("abcd"));

    }

}
```

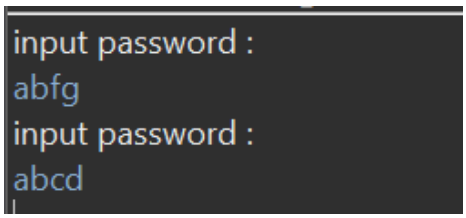
Kode Program 2.4

Kode ini menggunakan perulangan do while untuk validasi password, meminta input hingga benar ("abcd").

Langkah kerja :

1. Inisialisasi: Objek Scanner, variabel phrase.
2. Masuk loop do: Eksekusi blok (minta input password).
3. Setelah eksekusi, periksa kondisi while (!phrase.equals("abcd")): Jika tidak sama, ulang loop.
4. Loop berulang hingga input benar, lalu keluar.

Output:



Output 2.4

Analisis hasil :

Program meminta input berulang hingga "abcd" dimasukkan. Ini menekankan bahwa do while menjamin setidaknya satu eksekusi, cocok untuk validasi. Teori:

Berbeda dari while, do while ideal untuk input validation karena menghindari skip jika kondisi awal false. Referensi: Dalam "Effective Java" oleh Joshua Bloch, do while direkomendasikan untuk loop yang harus berjalan minimal sekali, seperti autentikasi, mengurangi risiko infinite loop jika kondisi salah.

2.5 Kode 5 : perulanganWhile1_2511531005

```java
import java.util.Scanner;

public class SentinelLoop_2511531005 {

    public static void main(String[] args) {
        Scanner console = new Scanner (System.in);
        int sum = 0;
        int number = 12;

        while ( number !=0) {
            System.out.println("masukkan angka (0 untuk keluar)");
            number = console.nextInt();
            sum = sum + number ;
        }
         System.out.println("totalnya adalah " + sum);
    }

}
```

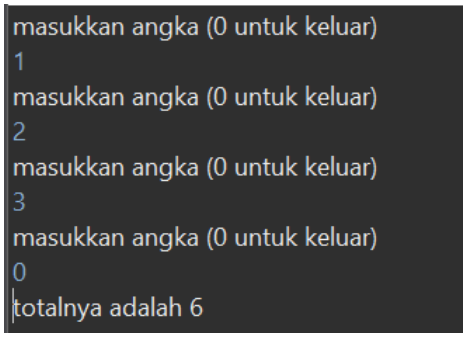
Kode Program 2.5

Kode ini menggunakan perulangan while sebagai sentinel loop, menjumlahkan angka hingga input 0.

Langkah kerja :

1. Inisialisasi: sum = 0, number = 12.
2. Loop while (number != 0): Jika number bukan 0, lanjut.
3. Minta input number, tambahkan ke sum.
4. Loop berulang hingga input 0, lalu cetak total sum.

Output:



Output 2.5

Analisis hasil :

Output menampilkan total penjumlahan (misalnya "totalnya adalah 6") setelah input 0. Ini mendemonstrasikan sentinel loop, di mana 0 adalah sentinel untuk berhenti. Teori: while dengan sentinel mirip do while dalam memungkinkan fleksibilitas input, tapi memeriksa kondisi dulu. Referensi: Berdasarkan algoritma dasar, ini efisien untuk data stream tanpa ukuran tetap, seperti dalam pemrosesan file, mencegah loop tak terbatas dengan sentinel yang jelas.

Secara keseluruhan, kode-kode ini menunjukkan aplikasi praktis perulangan, dengan while untuk kondisi awal dan do while untuk eksekusi minimal, mendukung pembelajaran algoritma kontrol alur. Analisis ini didasarkan pada prinsip efisiensi dan keamanan loop dalam Java.

BAB III
KESIMPULAN

3.1 Kesimpulan

Praktikum ini menunnjukkan implementasi perulangan while dan do while dalam bahasa pemrograman Java melalui lima contoh kode program yang beragam, mulai dari kontrol interaktif pengguna hingga simulasi probabilistik dan validasi input. Dari analisis, dapat disimpulkan bahwa perulangan while lebih cocok untuk skenario di mana kondisi harus diperiksa sebelum eksekusi, seperti dalam loop dengan jumlah iterasi tidak pasti atau sentinel-controlled, sedangkan do while ideal untuk memastikan eksekusi minimal sekali, seperti dalam validasi password.

Hasil praktikum menunjukkan bahwa kedua jenis perulangan ini meningkatkan efisiensi kode dengan mengurangi redundansi, namun memerlukan pengelolaan kondisi yang ketat untuk menghindari infinite loop. Secara keseluruhan, praktikum ini memperkuat pemahaman tentang struktur kontrol alur dalam algoritma pemrograman, yang esensial untuk pengembangan aplikasi interaktif dan simulasi.

3.2 Saran

Untuk meningkatkan kualitas praktikum kedepannya, disarankan untuk menambahkan variasi kode yang lebih kompleks, seperti integrasi dengan array atau exception handling, agar mahasiswa dapat menghadapi tantangan dunia nyata. Selain itu, saran untuk mahasiswa adalah mempelajari referensi tambahan tentang algoritma kontrol alur guna mendalami aplikasi praktis, seperti dalam pengembangan game atau sistem otomatis.

DAFTAR PUSTAKA

[1] H. Schildt, Java: A Beginner's Guide, 6th ed. New York: McGraw-Hill, 2014.

[2] Oracle Corporation, "The Java Tutorials: Control Flow Statements," [Online]. Available: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html. [Accessed: Oct. 15, 2023].

[3] J. Bloch, Effective Java, 3rd ed. Boston: Addison-Wesley, 2018.

[4] R. Sedgewick and K. Wayne, Introduction to Programming in Java: An Interdisciplinary Approach, Boston: Addison-Wesley, 2007.