LAPORAN PRAKTIKUM

PEMROGRAMAN ALGORITMA DAN PEMROGRAMAN

"IMPLEMENTASI STRUKTUR PERULANGAN DALAM BAHASA PEMROGRAMAN JAVA"

disusun oleh

Hafizh Habibullah

2511531002

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

Asisten Praktikum: Muhammad Zaki Al Hafiz

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN 2025

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Allah SWT. karena atas rahmat dan karunia-Nya laporan praktikum Algoritma dan Pemrograman ini dapat diselesaikan dengan baik.

Laporan ini disusun untuk memenuhi tugas praktikum pada mata kuliah Algoritma dan Pemrograman dengan topik *Implementasi Struktur Perulangan dalam Bahasa Pemrograman Java.* Melalui laporan ini, penulis berusaha menjelaskan hasil praktikum yang meliputi kode program, penjelasan tentang cara kerja program yang dibuat, serta hasil output dari program.

Penulis sangat mengharapkan kritik dan saran yang membangun agar dapat menjadi bahan perbaikan di kemudian hari. Semoga laporan ini dapat memberikan manfaat bagi penulis maupun pembaca.

Padang, 8 November 2025

Hafizh Habibullah

DAFTAR ISI

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Struktur Perulangan adalah salah satu elemen penting dalam pemrograman karena ini memungkinkan eksekusi perintah secara berulang tanpa perlu menulis kode yang sama berulang kali. Dalam Bahasa pemrograman Java, terdapat beberapa jenis perulangan seperti *while, do-while,* dan *for* yang dapat digunakan sesuai kebutuhan logika program.

Melalui penerapan perulangan, proses komputasi yang bersifat berulang dapat dilakukan secara efisien dan terstruktur. Pada praktikum ini, struktur perulangan diimplementasikan dalam berbagai bentuk program sederhana, seperti permainan penjumlahan, simulasi lempar dadu, dan Latihan logika berbasis *loop*. Dengan begitu, pemahaman mengenai konsep dasar perulangan dapat diperkuat melalui penerapan langsung dalam program interaktif.

1.2 Tujuan

Adapun tujuan dari praktikum ini adalah:

1. Memahami konsep dasar dan cara kerja struktur perulangan dalam Java.
2. Mengimplementasikan berbagai jenis perulangan dalam program sederhana.
3. Melatih kemampuan logika dalam merancang alur program yang menggunakan proses berulang.
4. Menguhubungkan konsep teoritis perulangan dengan penerapan praktis dalam membuat program interaktif.

1.3 Manfaat

Manfaat dari Praktikum ini adalah:

1. Menambah pemahaman tentang penggunaan perulangan secara efektif dalam pemrograman Java.
2. Meningkatkan keterampilan dalam menyusun program yang efisien dan mudah dipahami.

3. Menjadi dasar dalam pengembangan program yang lebih kompleks nantinya.
4. Membiasakan diri dalam berpikir logis dalam menyelesaikan permasalahan menggunakan kode.

BAB 2
PEMBAHASAN

2.1 Perulangan dengan *while-loop*

While-loop adalah salah satu perulangan dalam bahasa pemrograman Java yang digunakan jika pengguna ingin melakukan perulangan secara terus-menerus hingga kondisinya tidak lagi memenuhi. Yang membedakan *while-loop* dan *for-loop* adalah waktu *while-loop* berhenti tidak akan diketahui, sedangkan pada *for-loop* kita bisa mengatur batasannya.

```java
package pekan6_2511531002;

import java.util.Scanner;

public class perulanganWhile1_2511531002 {
    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.print("Apakah lanjut (ya/tidak?)");
            jawab = scan.nextLine();

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

Gambar 2. 1 - Kode Program perulanganWhile1

Dalam kode program ini, dibuat kode program simpel yang menghitung berapa banyak perulangan yang dilakukan sesuai input dari pengguna. Selama program ini dijalankan, program akan menambahkan *counter* setiap kali program diulang, yang diikuti dengan output yang menampilkan jumlah perulangan.

Program juga menampilkan output yang menanyakan pengguna jika perulangan ingin dilanjutkan atau tidak, yang nantinya hasil input dari pengguna digunakan untuk menentukan apakah perulangan akan dilanjutkan atau dihentikan. Jika pengguna memberi input “tidak”, maka perulangan akan berhenti. Terakhir, program akan menampilkan output yang menunjukkan berapa banyak perulangan yang sudah dilakukan oleh pengguna. Hasil outputnya adalah sebagai berikut:

Hasil Output
Jumlah = 1 Apakah lanjut (ya/tidak?) ya Jumlah = 2 Apakah lanjut (ya/tidak?) ya Jumlah = 3 Apakah lanjut (ya/tidak?) ya Jumlah = 4 Apakah lanjut (ya/tidak?) ya Jumlah = 5 Apakah lanjut (ya/tidak?) ya Jumlah = 6 Apakah lanjut (ya/tidak?) tidak Anda sudah melakukan perulangan sebanyak 6 kali

Tabel 2. 1 - Hasil Output dari perulanganWhile1

2.2 Simulasi Permainan Melempar Dadu dengan *while-loop*

Program selanjutnya adalah program permainan Melempar Dadu yang menggunakan *method* ***Random*** yang gunanya adalah untuk memilih bilangan bulat secara acak. Jika diberikan suatu argumen n yang merupakan bilangan bulat positif, maka *method* tersebut akan memilih bilangan bulat positif dari 1 sampai n.

```
1  package pekan6_2511531002;
2
3  import java.util.Random;
4
5  public class Lempardadu_2511531002 {
6      public static void main(String[] args) {
7          Random rand = new Random();
8          int tries = 0;
9          int sum = 0;
10         while (sum != 7) {
11             int dadu1 = rand.nextInt(6) + 1;
12             int dadu2 = rand.nextInt(6) + 1;
13             sum = dadu1 + dadu2;
14             System.out.println(dadu1 + " + " + dadu2 + " = " + sum);
15             tries++;
16         }
17         System.out.println("You won after " + tries + " tries!");
18     }
19 }
```

Gambar 2. 2 - Kode Program MembuatKue

Program menggunakan *utility* ***Random*** pada dua buah dadu, yaitu **dadu1** dan **dadu2**. Lalu, program akan menjumlahkan hasil nilai acak dari kedua dadu. Program kemudian akan menampilkan output yang menunjukkan penjumlahan dari kedua dadu tersebut,

Program akan terus berulang jika hasil penjumlahan dadu tidak bernilai **7**. Program juga akan menambahkan **tries** (jumlah perulangan program) sampai kondisi yang dituju sudah tidak sesuai (hasil penjumlahan dadu = 7). Terakhir, jumlah **tries** akan ditampilkan bersamaan dengan pernyataan bahwa pengguna sudah memenangkan permainan setelah berapa kali percobaan. Hasil output kurang lebih akan terlihat seperti berikut:

Hasil Output
1 + 2 = 3 1 + 5 = 6 3 + 2 = 5 2 + 4 = 6 4 + 6 = 10 4 + 3 = 7 You won after 6 tries!

Tabel 2. 2 - Hasil Output dari Lempardadu

2.3 Kode Program Bermain Hitungan 2-5 Angka

```java
package pekan6_2511531002;

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

public class GamePenjumlahan_2511531002 {
    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.");
    }
    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;
        }
    }
}
```

Gambar 2. 3 - Kode Program GamePenjumlahan

Program diatas merupakan contoh penerapan *while-loop* untuk membuat sebuah permainan penjumlahan sederhana. Dalam permainan ini, pengguna diminta untuk menjawab soal penjumlahan dengan beberapa angka acak yang dihasilkan secara otomatis oleh program. Program akan terus berjalan selama pengguna belum menjawab pertanyaan dengan salah sebanyak 3 kali.

Di dalam *method* main(), digunakan dua buah variabel utama: **points** untuk menghitung skor benar, dan **wrong** untuk menghitung jumlah jawaban yang salah.

Kondisi *while-loop* **while (wrong < 3)** memastikan bahwa permainan akan terus berjalan sampai pengguna menjawab pertanyaan dengan salah sebanyak 3 kali.

Selama perulangan, *method* play() dipanggil untuk menjalankan satu sesi permainan. Di dalam *method* ini, program menentukan berpaa banyak angka yang akan dijumlahkan secara acak melalui **rand.nextInt(4) + 2**, sehingga setiap soal terdiri dari dua sampai lima bilangan. Nilai awal penjumlahkan ditetapkan dengan **sum = rand.nextInt(10) + 1**. Kemudian sebuah perulangan *for-loop* digunakan untuk menambahkan bilangan acak lainnya sambil menampilkan ekspresi matematika ke layar.

Setelah semua bilangan ditampilkan, program kemudian meminta pengguna untuk memasukkan hasil penjumlahan. Jika jawaban pengguna sesuai dengan nilai **sum** (jumlah dari semua bilangan), maka *method* mengembalikan nilai 1 sebagai penanda jawaban benar. Jika salah, program akan menampilkan pesan kesalahan disertai jawaban yang benar, lalu mengembalikan nilai 0.

Nilai hasil pemanggilan *method* play() kemudian digunakan dalam *while-loop* utama untuk memperbarui skor atau jumlah kesalahan. Setelah pengguna melakukan 3 kesalahan, kondisi *while* tidak lagi terpenuhi, dan program akan mencetak total skor akhir dengan pesan: **”You earned x total points”.**

Hasil Output
4 + 1 + 1 = 6 2 + 7 + 7 + 9 + 10 = 24 Wrong! The answer was 35 6 + 5 + 8 + 6 + 7 = 30 Wrong! The answer was 32 9 + 9 + 3 + 2 + 5 = 28 4 + 1 + 4 = 9 6 + 3 + 10 + 6 = 25 8 + 8 + 2 = 18 3 + 5 + 5 = 12 Wrong! The answer was 13 You earned 5 total points.

Tabel 2. 3 - Hasil Output dari GamePenjumlahan

2.4 Perulangan dengan Sentinel Value

Program ini merupakan contoh penerapan *while loop* dengan konsep *sentinel value*. Pada struktur perulangan ini, proses perulangan akan terus dijalankan sampai pengguna memasukkan nilai tertentu yang berfungsi sebagai sinyal untuk menghentikan program. Dalam kasus ini, angka penanda tersebut adalah angka **0**.

```java
package pekan6_2511531002;

import java.util.Scanner;

public class SentinelLoop_2511531002 {
    public static void main(String[] args) {
        Scanner console = new Scanner (System.in);
        int sum = 0;
        int number = 12;

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

Gambar 2. 4 - Kode Program SentinelLoop

Pertama, dua variabel untama dideklarasikan, yaitu **sum** untuk menyimpan total hasil penjumlahan, dan **number** sebagai variabel penampung input dari pengguna. Perulangan **while** menggunakan kondisi **number != 0**, yang berarti program akan terus meminta masukan angka selama nilai yang diberikan bukan 0.

Setiap angka yang dimasukkan oleh pengguna ditambahkan ke variabel sum. Setelah pengguna memasukkan angka 0, kondisi perulangan menjadi *false*, sehingga proses perulangan berhenti, dan program akan menampilkan total dari seluruh angka yang sudah dijumlahkan.

Hasil Output
Masukkan angka (0 untuk keluar): 6
Masukkan angka (0 untuk keluar): 7
Masukkan angka (0 untuk keluar): 41
Masukkan angka (0 untuk keluar): 6
Masukkan angka (0 untuk keluar): 7
Masukkan angka (0 untuk keluar): 0
totalnya adalah 67

Tabel 2. 4 - Hasil Output dari GamePenjumlahan

2.5 Perulangan do-while untuk pengecekan validasi password

Program ini memunjukkan penerapan struktur perulangan *do-while* pada pemrograman Java. Perbedaan dari *do-while* dan *while-loop* terletak pada urutan eksekusinya: *do-while* akan selalu menjalankan blok perintah **setidaknya satu kali**, kemudian memeriksa kondisi pada bagian akhir.

```java
package pekan6_2511531002;

import java.util.Scanner;

public class doWhile1_2511531002 {
    public static void main(String[] args) {
    Scanner console = new Scanner(System.in);
        String phrase;
        do {
            System.out.print("Input Password: ");
            phrase = console.next();
        } while (!phrase.equals("abcd"));
        console.close();
    }
}
```

```
<terminated> doWhile1_251
Input Password: uhh
Input Password: lupa
Input Password: hehe
Input Password: abcd
```

Gambar 2. 5 - Kode Program doWhile1

Pada program ini, variabel *phrase* digunakan untuk menyimpan input dari pengguna. Perintah *do* menandakan bahwa program akan terlebih dahulu meminta pengguna untuk mengetikkan sebuat *password*. Setelah input diterima, kondisi **!phrase.equals(“abcd”)** dievaluasi.

Jika kondisi tersebut bernilai *true* (kata sandi yang dimasukkan bukan “abcd”), maka program akan menjalankan perulangan dan pengguna diminta untuk menginput ulang passwordnya. Sebaliknya, jika pengguna memasukkan input “abcd”, kondisi berubah menjadi *false*, dan perulangan akan berhenti.

BAB III
KESIMPULAN

3.1 Kesimpulan

Dari hasil praktikum mengenai struktur perulangan *while* dan *do-while* pada bahasa pemrograman Java, dapat disimpulkan bahwa keduanya digunakan untuk menjalankan perintah secara berulang berdasarkan kondisi logika tertentu. Perulangan *while* memeriksa kondisi di awal sebelum dijalankan, sedangkan *do-while* mengeksekusi minimal satu kali sebelum memeriksa kondisi di akhir.

Melalui contoh program seperti lempar dadu, permainan penjumlahan, dan sentinel loop, konsep perulangan dapat dipahami secara lebih jelas. Praktikum ini juga menegaskan pentingnya menentukan kondisi berhenti dengan tepat agar program tidak berjalan tanpa batas.

3.2 Saran Pengembangan

Untuk pengembangan selanjutnya, program dapat dibuat lebih interaktif, misalnya dengan menambahkan sistem skor, level kesulitan, atau antarmuka grafis sederhana. Selain itu, konsep perulangan juga bisa diterapkan pada pengolahan data yang lebih kompleks seperti penggunaan array atau perulangan bersarang untuk memperluas pemahaman logika pemrograman.

DAFTAR PUSTAKA

[1] Oracle, “The while and do-while Statements,” The Java™ Tutorials: Language Basics – Control Flow Statements, Oracle, 2024. Available: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html