LAPORAN PRATIKUM

ALGORITMA DAN PEMROGRAMAN

“Perulangan Menggunakan *while loop* dan *do-while loop*”

disusun Oleh:

Muhammad Yasin Habiburrahman

2511532016

Dosen Pengampu:

Dr. Wahyudi. S.T.M.T

Asisten Pratikum:

Muhammad Zaki Al Hafiz

DEPARTEMEN INFORMATIKA FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN 2025

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan tugas kelompok ini dengan baik. Shalawat serta salam semoga senantiasa tercurah kepada junjungan kita Nabi Muhammad SAW, keluarga, sahabat, serta para pengikutnya hingga akhir zaman.

Tugas kelompok ini kami susun dengan tujuan untuk memenuhi salah satu tugas mata kuliah Algoritma dan Pemrograman, dengan tema "*Perulangan menggunakan while loop dan do-while loop*", yang mencakup bagaimana cara kerja dan penggunaan dari *while loop* dan *do-while loop*

Kami menyadari bahwa penyusunan tugas ini masih jauh dari sempurna, baik dari segi materi maupun penyajiannya. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari semua pihak demi kesempurnaan tugas kami di masa mendatang.

Akhir kata, kami mengucapkan terima kasih kepada dosen pembimbing yang telah memberikan arahan, serta kepada seluruh anggota kelompok yang telah bekerja sama dengan baik sehingga tugas ini dapat terselesaikan tepat waktu. Semoga tugas ini dapat bermanfaat bagi kami khususnya, dan bagi pembaca pada umumnya.

Padang, 5 November 2025

Muhammad Yasin Habiburrahman

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam pemrograman, perulangan merupakan salah satu struktur kontrol yang sangat penting karena memungkinkan suatu blok kode dijalankan secara berulang berdasarkan kondisi tertentu. Salah satu bentuk perulangan yang umum digunakan dalam bahasa Java adalah *while loop*. Berbeda dengan for *loop* yang jumlah perulangannya sudah diketahui sejak awal, *while loop* digunakan ketika jumlah iterasi belum pasti dan bergantung pada kondisi logika yang diberikan.

Struktur *while loop* bekerja dengan cara mengevaluasi kondisi terlebih dahulu sebelum menjalankan blok perintah di dalamnya. Jika kondisi bernilai true, maka perintah akan dieksekusi; sebaliknya, jika kondisi bernilai false, maka perulangan akan berhenti. Oleh karena itu, *while loop* sering digunakan pada situasi di mana proses pengulangan perlu dilakukan sampai kondisi tertentu terpenuhi, misalnya membaca data dari pengguna, melakukan validasi input, atau menjalankan program hingga suatu nilai target tercapai.

Melalui praktikum ini, mahasiswa diharapkan dapat memahami cara kerja struktur perulangan *while*, menerapkan logika berhenti (termination condition) dengan benar, serta menghindari terjadinya perulangan tak berujung (infinite *loop*).

1.2 Tujuan

Tujuan dari praktikum ini adalah sebagai berikut:

1. Memahami konsep dasar perulangan *while* pada bahasa pemrograman *Java*.
2. Mampu menuliskan dan menjalankan program dengan struktur *while loop* secara benar.
3. Menganalisis kondisi logika yang memengaruhi jalannya perulangan.
4. Mengidentifikasi dan menghindari kesalahan umum seperti *infinite loop* dalam penulisan kode.

1.3 Manfaat Praktikum

Melalui praktikum ini, mahasiswa diharapkan dapat memperoleh manfaat berupa pemahaman yang lebih mendalam tentang konsep perulangan dinamis dalam pemrograman. Mahasiswa juga diharapkan mampu membangun logika program yang adaptif berdasarkan kondisi tertentu serta meningkatkan kemampuan berpikir algoritmik yang sistematis. Selain itu, pemahaman terhadap *while loop* akan menjadi dasar penting untuk mempelajari bentuk perulangan lain seperti *do-while* dan penerapannya dalam pemrosesan data berulang

BAB II
PEMBAHASAN

2.1 Perulangan dengan *while-loop*

While-loop adalah salah satu perulangan pada bahasa pemrograman Java yang berguna jika seseorang ingin melakukan perulangan secara terus menerus sampai kondisinya tidak lagi memenuhi. Perbedaan antara *while loop* dan *for loop* adalah kita tidak tahu pasti kapan *while loop* akan berhenti. Berbeda dengan *for loop* yang batasannya telah diatur.

Struktur dasar dari *while loop* adalah sebagai berikut:

```
While(kondisi){
  //isi looping
}
```

```
package pekan6_2511532016;
import java.util.Scanner;

public class perulanganWhile1_2511532016 {

    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? (Yes/No)");
        jawab = scan.nextLine();
        if(jawab.equalsIgnoreCase("no")) running = false;
    }
    scan.close();


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

}
```

```
<terminated> perulanganWhile1_2511532016 [Java Application] C
Jumlah = 6
Apakah lanjut? (Yes/No)
Jumlah = 7
Apakah lanjut? (Yes/No)
Jumlah = 8
Apakah lanjut? (Yes/No)
Jumlah = 9
Apakah lanjut? (Yes/No)
Jumlah = 10
Apakah lanjut? (Yes/No)no
Anda sudah melakukan perulangan sebanyak 10 kali
```

Gambar 2.1 – Kode perulangan*While*1

Kode program di atas mendemonstrasikan bagaimana *while loop* berjalan berdasarkan *input* dari pengguna. Kondisi *while loop* di atas adalah ia akan terus mengulang selagi nilai variabel *running* bernilai *true*. Diberikan pula variabel jawab untuk menerima masukan dari pengguna, dan *counter* untuk menghitung seberapa kali *while loop* telah melakukan perulangan.

Setiap kali *while loop* mencapai akhir dari programnya, ia akan mengeluarkan seberapa kali ia telah mengulang saat itu, serta pertanyaan kepada pengguna mengenai apakah ingin melanjutkan perulangan atau tidak. Perulangan dapat dihentikan oleh pengguna jika pengguna menjawab “No”. Saat itu juga, perulangan *while* akan berhenti dan akan mengeluarkan seberapa banyak *while loop* telah berulang.

2.2 Simulasi Melempar Sepasang Dadu Menggunakan *while loop*

While loop dapat digunakan untuk bermain permainan sederhana yang berdasarkan keberuntungan. Kode programnya sebagai berikut.

```
1  package pekan6_2511532016;
2  import java.util.Random;
3
4  public class LemparDadu_2511532016 {
5
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
12             int dadu1 = rand.nextInt(6) + 1;
13             int dadu2 = rand.nextInt(6) + 1;
14
15             sum = dadu1 + dadu2;
16
17
18             System.out.println(dadu1 + " + " + dadu2 + " = " + sum);
19             tries++;
20         }
21         System.out.println("Anda menang setelah " + tries + " kali percobaan!");
22     }
23
24 }
```

```
<terminated> LemparDadu_2511532016 [Java App
5 + 1 = 6
1 + 4 = 5
2 + 3 = 5
6 + 3 = 9
2 + 3 = 5
5 + 3 = 8
1 + 3 = 4
5 + 6 = 11
5 + 2 = 7
Anda menang setelah 9 kali percobaan!
```

Gambar 2.2 – Kode Program LemparDadu

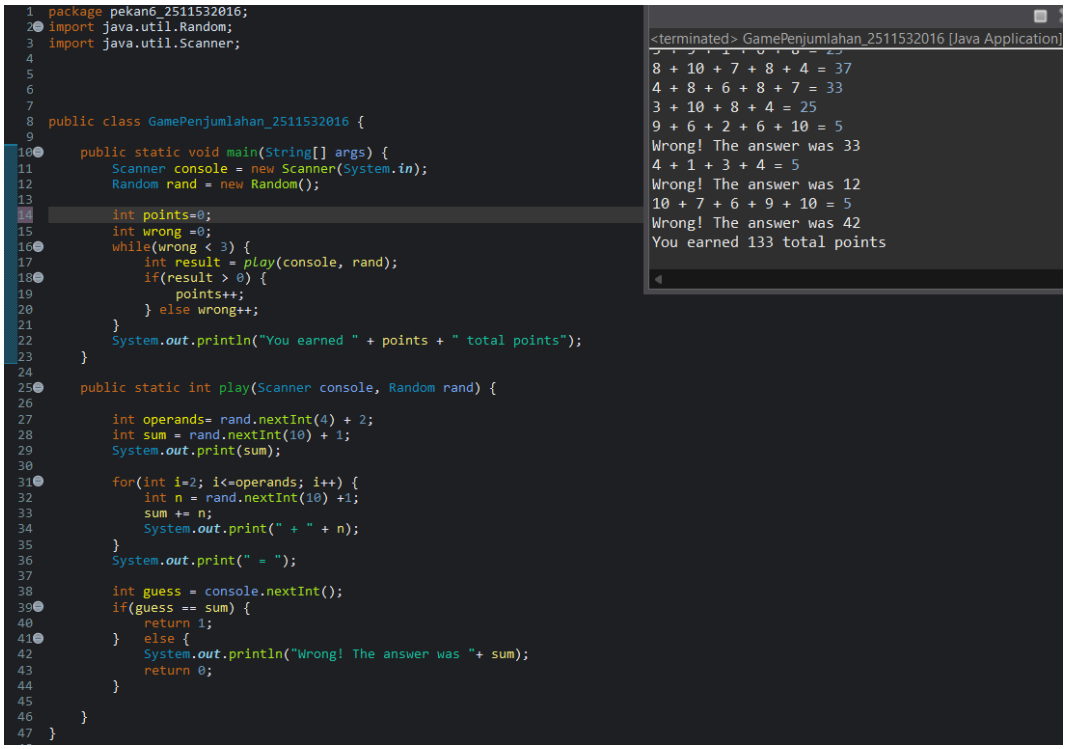
Kode program di atas menggunakan *method* yang berguna untuk memilih acak sebuah bilangan, yaitu *Random.* Method *Random* adalah Method yang memilih sembarang bilangan bulat. Jika diberikan suatu argumen n yang merupakan bilangan bulat positif, maka nilai kembalian dari method tersebut adalah bilangan bulat positif dari 1-n.

Kode program di atas merupakan contoh penggunaan *while loop* untuk mensimulasikan permainan lempar dua buah dadu. Program ini memanfaatkan *method* nextInt() dari kelas *Random* untuk menghasilkan bilangan acak antara 1 sampai 6, yang merepresentasikan nilai setiap sisi dadu.

Struktur *while loop* digunakan agar program terus melempar dadu selama jumlah kedua dadu (*sum*) belum sama dengan 7. Pada setiap perulangan, dua angka acak dihasilkan, dan masing-masing disimpan dalam variabel dadu1 dan dadu2. Kemudian, nilai keduanya dijumlahkan dan hasilnya disimpan dalam variabel sum.

Selama kondisi sum != 7 masih *true*, perulangan akan terus berjalan, dan setiap hasil lemparan akan ditampilkan ke layar. Setiap kali dadu dilempar, variabel *tries* ditambah satu untuk menghitung jumlah percobaan. Jika hasil penjumlahan kedua dadu sama dengan 7, kondisi menjadi *false* sehingga perulangan berhenti, dan program menampilkan pesan kemenangan beserta jumlah percobaannya.

2.3 Kode Program Bermain Hitungan 2-5 Angka



```java
package pekan6_2511532016;
import java.util.Random;
import java.util.Scanner;



public class GamePenjumlahan_2511532016 {

    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;
        }

    }
}
```

```
<terminated> GamePenjumlahan_2511532016 [Java Application]
8 + 10 + 7 + 8 + 4 = 37
4 + 8 + 6 + 8 + 7 = 33
3 + 10 + 8 + 4 = 25
9 + 6 + 2 + 6 + 10 = 5
Wrong! The answer was 33
4 + 1 + 3 + 4 = 5
Wrong! The answer was 12
10 + 7 + 6 + 9 + 10 = 5
Wrong! The answer was 42
You earned 133 total points
```

Gambar 2.3 – Kode Program GamePenjumlahan

Program di atas merupakan contoh penerapan *while loop* untuk membuat permainan penjumlahan sederhana. Dalam permainan ini, pengguna diminta untuk menjawab soal penjumlahan dengan beberapa angka acak yang dihasilkan secara otomatis oleh komputer. Program akan terus berjalan selama jumlah kesalahan pengguna belum mencapai tiga kali.

Di dalam *method* main(), dua buah variabel utama digunakan untuk mengatur permainan: points untuk menghitung skor benar, dan wrong untuk menghitung jumlah jawaban salah. Kondisi *while loop while* (wrong < 3) memastikan bahwa permainan akan terus berlangsung sampai pengguna melakukan tiga kesalahan.

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

Setelah seluruh bilangan ditampilkan, program meminta pengguna untuk memasukkan hasil penjumlahan. Jika jawaban pengguna sesuai dengan nilai sum, maka *method* mengembalikan nilai 1 sebagai tanda jawaban benar. Jika salah,

program menampilkan pesan kesalahan beserta jawaban yang benar, lalu mengembalikan nilai 0.

Nilai hasil pemanggilan *method* play() kemudian digunakan dalam *while loop* utama untuk memperbarui skor atau jumlah kesalahan. Ketika pengguna telah melakukan tiga kesalahan, kondisi *while* tidak lagi terpenuhi, dan program mencetak total skor akhir dengan pesan:

You earned x total points

2.4 Perulangan dengan Sentinel Value

```java
package pekan6_2511532016;
import java.util.Scanner;

public class SentinelLoop_2511532016 {

    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        int sum = 0;
        int number = 67;

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

}
```

```
<terminated> SentinelLoop_2511532016 [Java A
Masukkan angka (0 untuk keluar): 2
Masukkan angka (0 untuk keluar): 5
Masukkan angka (0 untuk keluar): 1
Masukkan angka (0 untuk keluar): 1
Masukkan angka (0 untuk keluar): 5
Masukkan angka (0 untuk keluar): 3
Masukkan angka (0 untuk keluar): 2
Masukkan angka (0 untuk keluar): 1
Masukkan angka (0 untuk keluar): 6
Masukkan angka (0 untuk keluar): 0
Totalnya adalah 26
```

Gambar 2.4 – Kode Program Sentinel*Loop*

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

Pertama, dua variabel utama 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 dalam variabel sum. Ketika pengguna akhirnya memasukkan angka 0, kondisi perulangan menjadi *false*, sehingga proses perulangan berhenti, dan program menampilkan total seluruh angka yang telah dijumlahkan.

2.5 Perulangan *do-while* untuk Validasi Mengecek *Password*

```java
package pekan6_2511532016;
import java.util.Scanner;

public class doWhile1_2511532016 {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
            String phrase;
            do {
                System.out.print("Input Password: ");
                phrase = console.nextLine();
            } while (!phrase.equals("abcd"));
            console.close();
        }
}
```

```
<terminated> doWhile1_2511532016 [Ja
Input Password: duh, lupa
Input Password: hehe
Input Password: gatau
Input Password: GULINGKAN PRAB
Input Password: abcd
```

Gambar 2.5 – Kode Program do*While*1

Program di atas menunjukkan penerapan struktur perulangan *do-while* pada bahasa pemrograman *Java*. Perbedaan utama antara *do-while loop* dan *while loop* terletak pada urutan eksekusinya: *do-while* akan selalu menjalankan blok perintah *setidaknya satu kali*, kemudian memeriksa kondisi pada bagian akhir.

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

Jika kondisi tersebut bernilai *true* (artinya kata sandi yang dimasukkan bukan "abcd"), maka perulangan akan dijalankan kembali dan pengguna diminta untuk menginput ulang. Sebaliknya, jika pengguna mengetik "abcd", kondisi menjadi *false*, dan perulangan berhenti.

BAB III
KESIMPULAN

Berdasarkan hasil praktikum yang telah dilakukan mengenai struktur perulangan *while* dan *do-while* pada bahasa pemrograman *Java*, dapat disimpulkan bahwa kedua jenis perulangan ini berfungsi untuk menjalankan serangkaian perintah secara berulang selama kondisi logika tertentu terpenuhi. Perulangan *while* melakukan pengecekan kondisi terlebih dahulu sebelum mengeksekusi blok perintah, sehingga perulangan mungkin tidak dijalankan sama sekali jika kondisi awal tidak terpenuhi. Sebaliknya, *do-while* selalu mengeksekusi blok perintah minimal satu kali, kemudian baru memeriksa kondisi di bagian akhir.

Melalui berbagai contoh program, seperti simulasi lempar dadu, permainan penjumlahan, *sentinel loop*, dan validasi kata sandi, mahasiswa dapat memahami bahwa perulangan *while* cocok digunakan ketika jumlah iterasi tidak diketahui sejak awal, sedangkan *do-while* ideal untuk situasi di mana setidaknya satu kali proses harus dijalankan sebelum evaluasi kondisi. Praktikum ini juga menekankan pentingnya penentuan kondisi berhenti (*termination condition*) yang tepat agar tidak terjadi *infinite loop* yang menyebabkan program berjalan tanpa henti.

Secara keseluruhan, pemahaman terhadap konsep *while* dan *do-while loop* memberikan dasar penting dalam pengembangan logika pemrograman yang dinamis. Pengetahuan ini akan sangat berguna ketika mahasiswa mempelajari struktur kontrol yang lebih kompleks seperti *nested loop*, pengolahan data dengan *array*, serta algoritma yang memerlukan pengulangan bersyarat dan interaktif.

DAFTAR PUSTAKA

[1] Oracle. *Java Documentation*. https://docs.oracle.com/javase/