LAPORAN PRAKTIKUM

ALGORITMA PEMOGRAMAN

PERULANGAN WHILE DAN DO-WHILE PADA PEMOGRAMAN JAVA

Disusun Oleh :

Aliifah Felda Mufarrihati Salwaa

2511531011

Dosen Pengampu :

Dr. Wahyudi, S.T., M.T.

Asisten Praktikum :

Aufan Taufiqurrahman

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN 2025

KATA PENGANTAR

Laporan praktikum ini disusun sebagai salah satu bentuk pertanggungjawaban kegiatan praktikum algoritma pemograman yang membahas tentang perulangan while dan do-while pada pemograman Java. Melalui laporan ini penulis dapat memahami materi praktikum secara mendalam dan dapat lebih teliti, teratur, serta memiliki kemampuan yang baik dalam penulisan kode pemograman sesuai kaidah akademik. Sehingga laporan ini dapat menjadi sarana belajar, dokumentasi kegiatan, dan referensi praktikum berikutnya.

Penulis menyadari bahwa laporan ini masih memiliki banyak kekurangan, baik dari isi maupun penyajiannya. Oleh karena itu, saran dan kritik sangat penulis harapkan guna untuk menyempurnakan laporan berikutnya.

Padang, 4 November 2025

Aliifah Felda Mufarrihati Salwaa
2511531011

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang Praktikum

Perulangan *while* dan *do-while* merupakan salah satu dari sistem kontrol yang dapat membuat sebuah kode pemograman menjadi lebih efektif dan efisien. David J. Eck (2022: 76) berpendapat bahwa sebenarnya kita hanya membutuhkan satu struktur kontrol dari setiap kategori untuk memiliki pemograman yang serba guna. Lebih dari itu, pemakaian struktur kontrol adalah kenyamanan dari masing-masing pengguna.

Maka dari itu, pada praktikum kali ini kita akan membahas tentang lanjutan dari sistem operasi pengulangan untuk mengetahui sistem operasi mana yang tepat bagi sebuah kode pemograman. Pada praktikum kali ini, kita akan mengenal dan memahami tentang perulangan *while* dan *do-while.* Perulangan *while* hampir mirip dengan perulangan *for* dimana sistem akan melakukan perulangan hingga kondisi dari perulangan tersebut bernilai *false*. Sementara itu untuk *do-while,* perulangan setidaknya dilakukan sebanyak satu kali karena pemeriksaan kondisi dilakukan pada bagian bawah.

Untuk dapat lebih mengetahui dan memahami tentang perulangan *while* dan *do-while,* dibutuhkan praktikum yang sistematis dan dapat menjelaskan fungsi serta cara kerja dari perulangan ini. Oleh karena itu, dilakukan praktikum pada pemograman bahasa Java agar mahasiswa lebih memahami bagaimana penggunaan perulangan *while* dan *do-while.*

1.2 Tujuan Praktikum

Tujuan dari pelaksanaan praktikum ini adalah sebagai berikut:

1. Mengetahui bagaimana cara penggunaan perulangan *while* dan *do-while* pada bahasa pemograman Java.
2. Mengetahui langkah-langkah dalam membuat program Java dengan menggunakan perulangan *while* dan *do-while*.
3. Mengetahui hasil dari program Java yang menggunakan perulangan *while* dan *do-while*

1.3 Manfaat Praktikum

Manfaat dari pelaksanaan praktikum ini adalah sebagai berikut:

1. Memahami penulisan kode pemograman Java terutama dalam menggunakan perulangan *while* dan *do-while*.
2. Mampu mengimplementasikan perulangan *while* dan *do-while* dalam pembuatan kode program Java kedepannya.
3. Meningkatkan kemampuan dalam penulisan kode program yang efektif dan efisien pada bahasa pemograman Java dengan menggunakan struktur kontrol yang tepat.

BAB II
PEMBAHASAN

2.1. Perulangan *while* dan *do-while* pada pemograman Java

Perulangan *while* dan *do-while* merupakan salah satu dari enam struktur kontrol yang ada pada pemograman Java. Struktur perulangan ini digunakan ketika kita ingin mengeksekusi satu atau lebih pernyataan secara berulang hingga kondisi yang kita inginkan bernilai salah (*false*).

Struktur perulangan juga sudah kita bahas pada praktikum sebelumnya, yakni praktikum perulangan *for*. Walaupun memiliki fungsi dasar yang sama, tetapi perulangan *for* dan perulangan *while* memiliki perbedaan dalam penggunaan. Perulangan *for* kita gunakan ketika kita sudah mengetahui jumlah iterasi, sementara perulangan *while* kita gunakan ketika kita tidak mengetahui iterasi yang pasti dan hanya memiliki kondisi. Eksekusi pada perulangan *while* akan terus berjalan hingga kondisi yang kita miliki bernilai *false.* Bentuk paling sederhana dari perulangan *while* adalah sebagai berikut:

```
while (condition) statement;
```

Berbeda dari perulangan *while,* perulangan *do-while* melakukan eksekusi terlebih dahulu baru mengecek apakah kondisinya bernilai *true/false*. Sejatinya, perulangan *do-while* ini akan melakukan setidaknya sekali eksekusi dalam program. setelah melakukan eksekusi dan pengecekan, barulah program akan menentukan apakah perulangan akan dilanjutkan atau tidak. Bentuk paling sederhana dari perulangan *do-while* adalah sebagai berikut:

```
do { statements; } while(condition);
```

2.2. Praktikum “perulanganWhile1”

```
1. package pekan6_2511531011;
2. 
3. import java.util.Scanner;
4. 
5. public class perulanganWhile1_2511531011 {
6. 
7.     public static void main(String[] args) {
8. 
9.           int counter=0;
10.          String jawab;
11.          boolean running=true;
12.          //deklarasi scanner
13.          Scanner scan = new Scanner(System.in);
14.          while (running) {
15.                counter ++;
16.                System.out.println("Jumlah = "+counter);
17.                System.out.print("Apakah lanjut (ya/tidak?) = ");
18.                jawab= scan.nextLine();
19.                //cek jawab = tidak, perulangan berhenti
20.                if (jawab.equalsIgnoreCase("tidak")) {
21.                      running=false;
22.                }
23. 
24.          }
25.          System.out.println("Anda sudah melakukan perulangan
   sebanyak "+counter+" kali");
26.    }
27. 
28. }
```

Kode Program 2.1: Praktikum “perulanganWhile1”

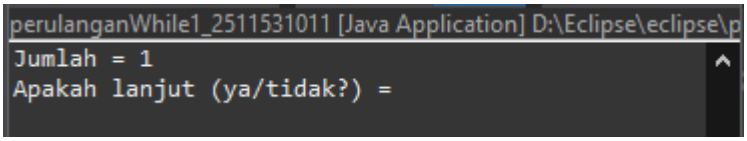
Pada kode program ini kita menggunakan bentuk perulangan *while* dengan 4 statement dan dikombinasikan dengan statement kondisional *if*. Disini kita juga akan meminta *input* dari user untuk menentukan apakah program perulangan akan terus dilakukan atau tidak. Berikut penjelasan lengkap tentang kode program diatas:

1. Pada baris kode 3 kita melakukan inisialisasi *scanner* agar program dapat mengambil *input* dari user.
2. Pada baris kode 9-11 kita melakukan inisialisasi tipe data variabel, yakni: integer counter=0, string jawab, dan boolean running=true
3. Pada baris kode 13 kita melakukan deklarasi scanner scan.
4. Pada baris kode 14 kita melakukan perulangan *while* dengan kondisi running=true. Dimana di dalam perulangan tersebut terdapat beberapa statement, yakni:

a. counter++; dimana nilai dari variabel counter akan terus bertambah selama nilai running=true
b. print “jumlah=”+counter; dimana program akan menampilkan nilai dari counter yang kita dapatkan
c. print “apakah lanjut (ya/tidak)=”; dimana kita akan meminta *input* dari user
d. Pada baris kode 18 kita akan memasukkan *input* dari user ke dalam variabel jawab.
e. Kemudian kita akan melakukan pengecekan dengan statement kondisional *if*, dimana jika variabel jawab= “tidak”, maka nilai dari variabel running akan berubah menjadi false.

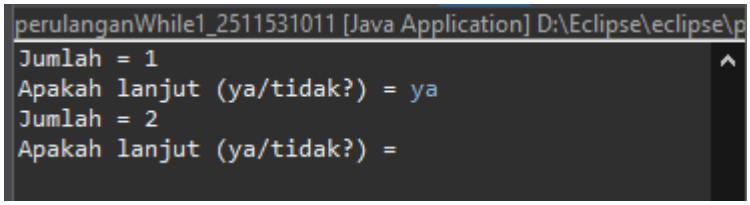
5. Pada baris kode 25, setalah program perulangan berakhir, program akan mengeluarkan *output* berupa kalimat “Anda sudah melakukan perulangan sebanyak”+nilai akhir dari variabel counter.

Dari kode program yang telah kita tuliskan, kita akan mendapatkan *output* awal sebagai berikut:



Gambar 2.1: *output* awal kode program “perulanganWhile1”

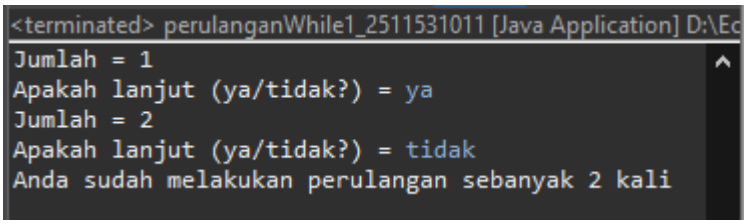
Dari sini, program akan meminta *input* dari user yang akan menentukan apakah perulangan akan terus dilakukan atau tidak. Jika kita menuliskan selain kata “tidak”, maka program akan menampilkan *output* berikutnya sebagai berikut:



Gambar 2.2: *output* 2 dari kode program “perulanganWhile1”

Program akan terus mengeluarkan *output* yang sama hingga user memasukkan kata “tidak”. Hal ini terjadi karena kita menggunakan string Java

equalsIgnoreCase() dimana metode ini akan membandingkan antara dua buah string dengan mengabaikan huruf besar dan huruf kecil. Dalam kasus ini, program akan membandingkan antara *input* yang diberikan user (variabel jawab) dengan kata "tidak". Jika ditemukan kesamaan, maka program akan menjalankan perintah selanjutnya yakni menjadikan variabel running bernilai *false* dan melakukan print out. Berikut contoh *output* jika user melakukan *input* "tidak":



Gambar 2.3: *output* 3 dari kode program "perulanganWhile1"

2.3. Praktikum "Lempardadu"

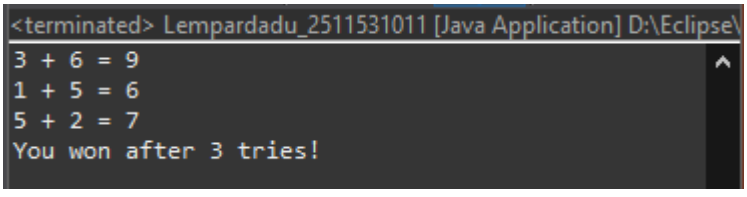
```
1.  package pekan6_2511531011;
2.
3.  import java.util.Random;
4.
5.  public class Lempardadu_2511531011 {
6.
7.      public static void main(String[] args) {
8.              Random rand = new Random();
9.              int tries = 0;
10.             int sum = 0;
11.             while (sum!=7) {
12.                     //roll the dice once
13.                     int dadu1 = rand.nextInt(6) + 1;
14.                     int dadu2 = rand.nextInt(6) + 1;
15.                     sum = dadu1 + dadu2;
16.                     System.out.println(dadu1+" + "+dadu2+" = "+sum);
17.                     tries++;
18.             }
19.             System.out.println("You won after "+tries+" tries!");
20.     }
21.
22. }
```

Kode Program 2.2: praktikum "Lempardadu"

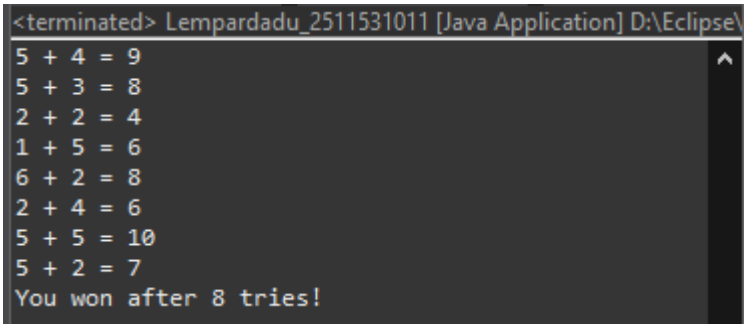
Pada kode program ini kita akan membuat suatu permainan sederhana, yakni permainan lempar dadu. Permainan lempar dadu ini menggunakan perulangan *while* untuk menentukan kemenangan melalui angka random. Perulangan akan

terus dilakukan hingga angka random yang keluar menghasilkan jumlah=7. Berikut untuk penjelasan lebih lengkap tentang kode program diatas:

1. Pada baris kode 3 kita melakukan import java.util.Random agar kita dapat menggunakan angka random pada program.
2. Pada baris kode 8 kita melakukan deklarasi Random rand.
3. Pada baris 9-10 kita melakukan inisialisasi integer tries=0 dan sum=0
4. Pada baris kode 11 kita melakukan perulangan *while* dengan kondisi sum!=7 dan dengan statement berikut:
    a. Inisialisasi integer dadu1=rand.nextInt(6) + 1; dimana variabel dadu1 merupakan angka random dari 1-6.
    b. Inisialisasi integer dadu2=rand.nextInt(6) + 1; dimana variabel dadu1 merupakan angka random dari 1-6.
    c. Melakukan penjumlahan dari nilai variabel dadu1+dadu2 dan dimasukkan dalam variabel sum.
    d. Memberikan *output* dengan mencetak nila variabel dadu1+dadu2+sum
    e. Tries++; dimana nilai dari variabel tries akan terus bertambah hingga kondisi dari sum!=7 bernilai false.
5. Pada baris kode 19, setelah perulangan berhenti, program akan mengeluarkan *output* berupa “You won after”+tries+”tries!”



Gambar 2.4: *output* 1 kode program “Lempardadu”



Gambar 2.5: *output* 2 kode program "Lempardadu"

Output yang dihasilkan oleh program ini akan selalu berbeda karena angka yang dihasilkan akan selalu random. Program akan berhenti dan kita dikatakan menang ketika hasil penjumlahan dari variabel dadu1 dengan dadu2 adalah 7.

2.4. Praktikum "PerulanganFor3"

```
1.  package pekan6_2511531011;
2.
3.  import java.util.Random;
4.  import java.util.Scanner;
5.
6.  public class GamePenjumlahan_2511531011 {
7.
8.      public static void main(String[] args) {
9.            Scanner console = new Scanner(System.in);
10.           Random rand = new Random();
11.           //play until user gets 3 wrong
12.           int points = 0;
13.           int wrong = 0;
14.           while (wrong<3) {
15.                 int result = play(console, rand);  //play one game
16.                 if (result>0) {
17.                       points++;
18.                 } else {
19.                       wrong++;
20.                 }
21.           }
22.           System.out.println("You earned "+points+" total points.");
23.     }
24.     //membuat soal penjumlahan dan ditampilkan ke user
25.     public static int play(Scanner console, Random rand) {
26.           //print the operands being added, and sum them
27.           int operands = rand.nextInt(4) + 2;
28.           int sum = rand.nextInt(10) + 1;
29.           System.out.print(sum);
30.           for (int i=2; i<=operands; i++) {
31.                 int n = rand.nextInt(10) + 1;
32.                 sum += n;
```

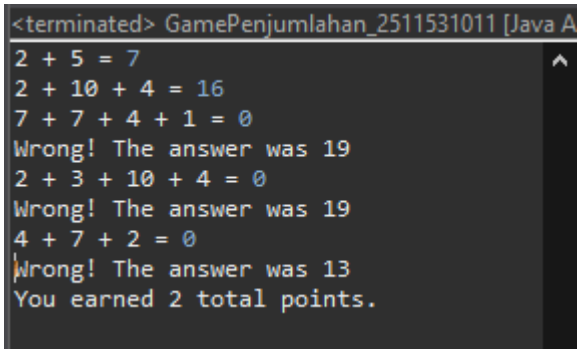
```
33.                 System.out.print(" + " + n);
34.           }
35.           System.out.print(" = ");
36.           // read users guess and report whether it was correct
37.           int guess = console.nextInt();
38.           if (guess==sum) {
39.                 return 1;
40.           } else {
41.                 System.out.println("Wrong! The answer was "+sum);
42.                 return 0;
43.           }
44.     }
45. }
```

Kode Program 2.3: Praktikum "GamePenjumlahan"

Pada kode program ini, kita juga akan melakukan permainan sederhana yang berkaitan dengan penjumlahan, dimana kita akan mendapatkan poin selama jawaban kita benar dan akan dinyatakan kalah jika menjawab salah sebanyak 3 kali. Berikut rincian dari program diatas:

1. Pada baris kode 3-4 kita melakukan impor scanner untuk memasukkan *input* dari user dan random untuk menghasilkan angka random.
2. Pada baris kode 12-13 kita melakukan inisialisasi variabel points=0 dan wrong=0
3. Pada baris kode 14 kita menggunakan perulangan *while* dengan kondisi wrong<3 dan dengan statement:
    a. Inisialisasi result=play(console, rand); dimana kita akan melakukan satu kali permainan.
    b. Menggunakan statement *if* dimana jika result>0, maka nilai points akan bertambah, dan jika lain dari itu, maka nilai wrong akan bertambah.
4. Pada baris kode 22 setelah perulangan berakhir, program akan menghasilkan *output* berupa jumlah point yang kita dapatkan.
5. Pada baris kode 25 kita mendeklarasikan metode play.
6. Pada baris kode 27 kita akan menghasilkan jumlah *operand* (angka yang akan dijumlahkan) secara acak dari 2-5
7. Pada baris kode 28 kita menginisialisasi variabel sum dengan angka acak pertama dari 1-10

8. Pada baris kode 29 program akan mengeluarkan *output* berupa nilai dari variabel sum.
9. Pada baris 30 kita menggunakan perulangan *for* dengan kondisi i<=operands statement:
    a. Menghasilkan nilai n random dari 1-10
    b. Menambahkan angka acak n ke variabel sum, sehingga sum menyimpan total penjumlahan semua *operand*
    c. Menghasilkan *output* berupa "+"+nilai n
10. Pada baris kode 35 jika perulangan *for* sudah berhenti, maka program menghasilkan *output* berupa "="
11. Pada baris kode 37 kita memasukkan *input* dari user ke dalam variabel *guess*.
12. Pada baris kode 38 kita menggunakan statement *if* dimana:
    a. Jika angka yang dimasukkan oleh user sama dengan nilai dari variabel sum (guess==sum) maka metode mengembalikan nilai 1 (untuk menghitung jumlah poin).
    b. Jika guess!=sum maka program akan mengeluarkan *output* berupa kalimat "Wrong! The answer was" dan menampilkan nilai sum (jawaban yang benar). Metode juga akan mengembalikan nilai 0 untuk menandakan jawaban salah.



Gambar 2.6: *output* kode program "GamePenjumlahan"

Gambar 2.6 merupakan salah satu *output* yang dihasilkan oleh program "GamePenjumlahan" dimana disini user menjawab 2 soal dengan benar dan 3 soal

salah. Program akan berhenti ketika user salah dalam menjawab sebanyak 3 kali dan akan menampilkan point (jumlah benar) yang user dapatkan. Soal yang keluar juga akan berbeda-beda karena kita menggunakan angka random untuk menghasilkan soal.

2.5. Praktikum "SentinelLoop"

```
1. package pekan6_2511531011;
2.
3. import java.util.Scanner;
4.
5. public class SentinelLoop_2511531011 {
6.
7.     public static void main(String[] args) {
8.           Scanner console = new Scanner(System.in);
9.           int sum = 0;
10.          int number = 12; //"dummy value", anything but 0
11.
12.          while (number != 0) {
13.                System.out.print("Masukkan angka (0 untuk keluar)");
14.                number = console.nextInt();
15.                sum = sum + number;
16.          }
17.          System.out.println("totalnya adalah "+sum);
18.          console.close();
19.    }
20.
21. }
```

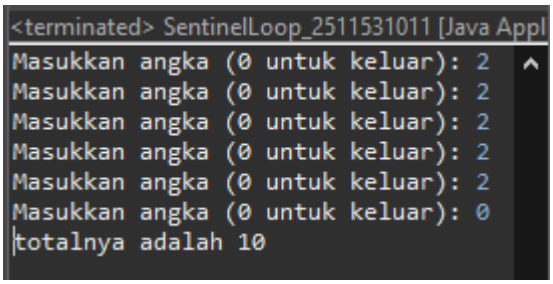
Kode Program 2.4: Praktikum "SentinelLoop"

Pada kode program kali ini kita akan melakukan *sentinel loop* dimana program akan berhenti ketika kita memasukkan satu kata kunci. Program ini menggunakan perulangan *while*. Berikut penjelasan lengkap tentang kode program diatas:

1. Pada baris kode 3 kita inisialisasi java.util.Scanner untuk melakukan *input*.
2. Pada baris kode 9-10 kita inisialisasi variabel sum=0 dan variabel number=12; dimana nilai number dapat berupa angka apapun kecuali 0.
3. Pada baris kode 12 kita melakukan perulangan *while* dengan kondisi (number!=0) dimana:
    a. Program akan meminta *input* berupa angka dari user
    b. Memasukkan *input* ke dalam variabel number

c. Menjumlahkan nilai variabel sum dengan variabel number dan memasukkannya ke dalam variabel sum.

4. Setelah perulangan *while* berakhir, program akan mengeluarkan *output* berupa nilai dari variabel sum.



Gambar 2.7: salah satu *output* kode program "SentinelLoop"

Dari *output* ini kita dapat menyimpulkan bahwa program akan terus berulang hingga kita memasukkan nilai sentinel. Nilai sentinel ini dapat diartikan sebagai nilai khusus yang dapat menghentikan program perulangan, disini kita menggunakan angka 0. Program akan terus berjalan hingga user memasukkan nilai 0. Di akhir, program akan menampilkan jumlah dari semua angka yang pernah kita masukkan ke dalam program.

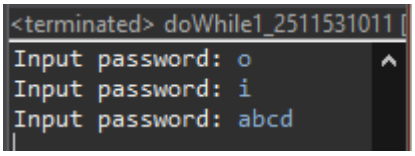
2.6. Praktikum "doWhile1"

```
1. package pekan6_2511531011;
2. 
3. import java.util.Scanner;
4. 
5. public class doWhile1_2511531011 {
6. 
7.     public static void main(String[] args) {
8.           Scanner console = new Scanner(System.in);
9.           String phrase;
10.          do {
11.                System.out.print("Input password: ");
12.                phrase = console.next();
13.          } while (!phrase.equals("abcd"));
14.          console.close();
15. 
16.    }
17. 
18. }
```

Kode Program 2.5: kode program praktikum "doWhile1"

Pada kode pemograman kali ini, kita akan menggunakan perulangan *do while* yang sedikit berbeda dibandingkan dengan perulangan *while*. Pada perulangan *do while*, program akan mengeksekusi statement setidaknya satu kali sebelum nilainya diperiksa. Berikut penjelasan lengkap tentang kode praktikum diatas:

1. Pada baris kode 3 kita melakukan inisialisasi java.util.Scanner untuk bisa meminta *input* dari user.
2. Pada baris kode 8-9 kita melakukan inisialisasi variabel scanner console dan variabel string phrase.
3. Pada baris kode 10 kita melakukan perulangan *do while*, dimana:
    a. Sistem melakukan *do* : meminta *input* dari user dan memasukkan input dari user ke dalam variabel phrase
    b. Sistem melakukan pengecekan dan apabila nilai variabel phrase tidak sama dengan “abcd”, maka perulangan akan terus berjalan.



Gambar 2.8: *output* kode program “doWhile1”

Gambar 2.8 merupakan hasil atau *output* dari kode program perulangan *do while*. Disini kita dapat menyatakan bahwasanya program meminta input terlebih dahulu baru melakukan perulangan. Jika kondisi dari perulangan bernilai *true*, maka program akan terus meminta input dari user. Jika kondisi perulangan bernilai *false*, barulah perulangan akan berhenti.

BAB III
KESIMPULAN

3.1. Kesimpulan

Kita mengetahui bahwasanya sistem perulangan merupakan salah satu sistem operasi yang sangat kita butuhkan ketika hendak membuat sebuah kode program yang efektif dan efisien. Pada praktikum kali ini kita membahas tentang sistem perulangan *while* dan *do while*. Sistem perulangan *while* kita lakukan ketika kita tidak mengetahui jumlah iterasi yang akan kita gunakan. Sementara itu sistem perulangan *do while* kita gunakan ketika kita hendak melakukan eksekusi terhadap statement terlebih dahulu baru kemudian melakukan pengecekan kondisi.

Perulangan *while* pada praktikum kali ini dapat kita gunakan untuk membuat game sederhana. Sementara itu perulangan *do while* kita gunakan untuk memasukkan kode password. Ini membuktikan bahwasanya perulangan *while* dan *do while* memiliki peran yang cukup penting ketika kita hendak menghasilkan suatu program yang interaktif dengan pengguna.

3.2. Saran

Terkadang sulit untuk menentukan kita akan memakai perulangan *while* atau *do while.* Untuk lebih memahami tentang kedua jenis perulangan ini, disarankan untuk terus melakukan latihan dengan mencoba banyak program. Berbagai eksperimen dalam membuat program juga akan banyak membantu agar kita lebih memahami bagaimana program tersebut bekerja.

DAFTAR PUSTAKA

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

[2] J. David, *Introduction to Programming Using Java*, 9th ed. New York, NY, 2022

[3] "Java While Loop," *W3Schools* [Daring]. Tersedia pada: https://www.w3schools.com/java/java_while_loop.asp [Diakses: 06-Nov-2025]

[4] "Java String equalsIgnoreCase() Method," *W3Schools* [Daring]. Tersedia pada: https://www.w3schools.com/java/ref_string_equalsignorecase.asp [Diakses: 06-Nov-2025]

[5] "Java util Random nextInt," *GeeksforGeeks* [Daring]. Tersedia pada: https://www.geeksforgeeks.org/java/java-util-random-nextint-java/ [Diakses: 06-Nov-2025]

[6] "Java Nested Loops," *W3Schools* [Daring]. Tersedia pada: https://www.w3schools.com/java/java_for_loop_nested.asp [Diakses: 06-Nov-2025]