LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN

LAPORAN PRAKTIKUM PEKAN 6

Disusun Oleh:

Syasya Halwa Gazwani

(2511531018)

Dosen Pengampu:

Dr. Wahyudi, S.T, M.T

Asisten Praktikum:

Muhammad Zaki Al Hafiz

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

PADANG

2025

KATA PENGANTAR

Assalamu'alaikum warahmatullahi wabarakatuh,

Segala puji bagi Allah SWT yang telah memberikan kesempatan serta kemudahan kepada penulis untuk dapat menyelesaikan Laporan Praktikum pada mata kuliah Algoritma Pemrograman, sehingga Laporan Praktikum ini dapat dikumpulkan dengan tepat waktu. Atas rahmat dan karunianya Laporan Praktikum dapat terselesaikan dengan baik. Sholawat serta salam juga penulis sampaikan kepada baginda tercinta yaitu Nabi Muhammad SAW. yang kita nantikan syafa'atnya di akhirat nanti. Laporan Praktikum ini bertujuan untuk menambah wawasan para pembaca untuk lebih memperdalam ilmu yang ada pada makalah ini.

Dalam penyusunan Laporan Praktikum ini, penulis mengalami banyak kesulitan dan penulis menyadari bahwa Laporan Praktikum ini masih jauh dari kesempurnaan. Untuk itu, penulis sangat mengharapkan kritik dan saran yang membangun demi kesempurnaan pada Laporan Praktikum ini. Penulis mengucapkan terima kasih kepada bapak Dr. Wahyudi, S.T, M.T. selaku dosen mata kuliah Algoritma Pemrograman yang telah memberikan tugas ini sehingga dapat menambah pengetahuan dan wawasan sesuai dengan mata kuliah yang penulis tekuni. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah membagi sebagian pengetahuannya sehingga penulis dapat menyelesaikan Laporan Praktikum ini.

Padang, November 2025

Penulis

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Pemrograman atau *progamming* merupakan sebuah proses menulis, menguji dan memperbaiki *(debug),* serta memelihara kode yang dapat membangun suatu program komputer. Kode ini ditulis dalam berbagai bahasa pemrograman atau sering disebut juga bahasa komputer. Tujuan dari pemrograman adalah untuk memuat suatu program yang dapat melakukan suatu perhitungan atau "pekerjaan" sesuai dengan keinginan pemrograman. Untuk melakukan pemrograman, diperlukan keterampilan dalam algoritma, logika, bahasa pemrograman, dan pada banyak kasus, pengetahuan-pengetahuan lain seperti matematika. Bahasa Pemrograman, atau biasanya disebut bahasa komputer atau *computer language programming*, yaitu sebuah instruksi standar untuk mengendalikan sebuah komputer. Seperangkat aturan *syntax* dan *semantic* yang digunakan untuk memberikan sebuah definisi pada program komputer dikenal sebagai bahasa pemrograman. Dengan menggunakan bahasa ini, seorang *programmer* dapat dengan tepat menentukan data yang akan diproses oleh komputer, bagaimana penyimpanan dan transfernya, dan tindakan apa yang harus diambil dalam berbagai keadaan.

1.2 Tujuan

1.2.1 Memahami konsep dasar tipe data dalam bahasa pemrograman Java serta penerapannya dalam pembuatan program sederhana.

1.2.2 Mengambangkan keterampilan analisis terhadap hasil keluaran program sesuai dengan logika yang diterapkan.

1.3 Manfaat Praktikum

1.3.1 Menambah wawasan dan keterampilan mahasiswa dalam mengelola data menggunakan bahasa pemrograman Java sebagai dasar untuk pemrograman yang lebih lanjut.

1.3.2 Memberikan pemahaman dasar kepada mahasiswa mengenai penggunaan tipe data dalam bahasa pemrograman Java, sehingga dapat menjadi bekal dalam mengembangkan program yang lebih kompleks di tahap berikutnya.

BAB II PEMBAHASAN

2.1 PerulanganWhile1

2.1.1 Kode Program

```
1  package pekan6_2511531018;
2
3  import java.util.Scanner;
4
5  public class PerulanganWhile1_2511531018 {
6      public static void main(String[] args) {
7
8          int counter=0;
9          String jawab;
10         boolean running = true;
11         //deklarasi scanner
12         Scanner scan= new Scanner(System.in);
13         while (running) {
14             counter++;
15             System.out.println("Jumlah = "+counter);
16             System.out.print("Apakah lanjut (ya / tidak?)");
17             jawab= scan.nextLine();
18             //cek jawab = tidak, perulangan berhenti
19             if (jawab.equalsIgnoreCase("tidak")) {
20                 running= false;
21
22             }
23
24         }
25         System.out.println("Anda sudah melakukan perulangan sebanyak "+counter+" kali");
26     }
27
28 }
```

Gambar 2.1.1 Kode program PerulanganWhile1.

2.1.2 Langkah Kerja

2.1.2.1 Program diawali dengan deklarasi *package* dan import *java.util.Scanner* untuk menggunakan *clasa Scanner* dalam membaca *input* dari pengguna.

2.1.2.2 Di dalam *class* PerulanganWhile1_2511531018, *metode main* didefinisikan sebagai titik awal eksekusi program.

2.1.2.3 *Int counter* = 0 untuk menghitung jumlah perulangan, *String* jawab untuk menampung jawaban pengguna, dan *boolean running* = *true* sebagai kondisi kontrol untuk perulangan.

```
8          int counter=0;
9          String jawab;
10         boolean running = true;
```

Gambar 2.1.2.3 Deklarasi 3 variabel.

2.1.2.4 Membuat objek *Scanner* untuk membaca *input* dari *keyboard* dan program masuk ke perulangan *while (running)* yang akan terus berjalan selama nilai *running* adalah *true.*

```
12          Scanner scan= new Scanner(System.in);
13          while (running) {
```

Gambar 2.1.2.4 Objek *Scanner* dan Perulangan *While.*

2.1.2.5 Nilai *counter* ditambah satu setiap kali *loop* berjalan *(counter++)*, Program menampilkan teks **"Jumlah = " + *counter*** untuk menunjukkan berapa kali perulangan telah dilakukan, Program meminta *input* pengguna, jawaban pengguna lalu dibaca, dan program memeriksa kondisi.

```
14              counter++;
15              System.out.println("Jumlah = "+counter);
16              System.out.print("Apakah lanjut (ya / tidak?)");
17              jawab= scan.nextLine();
18              //cek jawab = tidak, perulangan berhenti
19              if (jawab.equalsIgnoreCase("tidak")) {
20                  running= false;
```

Gambar 2.1.2.5 Nilai *counter* hingga program membaca kondisi.

2.1.2.6 Program berakhir setelah menampilkan jumlah total perulangan yang dilakukan oleh pengguna.

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

Gambar 2.1.2.6 Akhir program.

2.1.3 Analisis Hasil

```
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?)tidak
Anda sudah melakukan perulangan sebanyak 4 kali
```

Gambar 2.1.3 Hasil *output* kode program PerulanganWhle1.

Hasil dari program tersebut menunjukkan jumlah berapa kali pengguna melakukan perulangan berdasarkan respon yang diberikan. Setiap kali pengguna mengetik “ya”, program akan menambah nilai *counter* dan menampilkan jumlah perulangan yang telah dilakukan. Proses ini terus berulang sampai pengguna mengetik “tidak”. Pada saat itu, kondisi perulangan akan berhenti dan program menampilkan pesan akhir yang berisi total banyaknya perulangan yang dilakukan selama program berjalan. Dengan demikian, hasil akhirnya bergantung pada berapa kali pengguna memilih untuk melanjutkan sebelum akhirnya mengetik “tidak”.

2.2 LemparDadu

2.2.1 Kode Program

```java
package pekan6_2511531018;

import java.util.Random;

public class Lempardadu_2511531018 {

    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!");
    }

```

Gambar 2.2.1 Kode program LemparDadu.

2.2.2 Langkah Kerja

2.2.2.1 Program mendeklarasikan *package* dan mengimpor *class* **java.util.Random** untuk menghasilkan angka acak.

2.2.2.2 Di dalam *class* **Lempardadu_2511531018**, terdapat *metode main* yang menjadi titik awal eksekusi program.

2.2.2.3 Objek dibuat untuk menghasilkan nilai acak yang akan digunakan sebagai hasil lemparan dadu.

```
8           Random rand = new Random();
```

Gambar 2.2.2.3 Objek digunakan sebagai hasil.

2.2.2.4 *Int tries* = 0 untuk menghitung berapa kali dadu dilempar, *int sum* = 0 untuk menyimpan jumlah dari dua dadu yang dilempar, lalu program masuk le perulangan *while*.

```
9           int tries = 0;
10          int sum = 0;
11          while (sum !=7) {
```

Gambar 2.2.2.4 Deklarasi variabel.

2.2.2.5 Dadu pertama dan kedua dilempar dengan menghasilkan angka acak dari 1 hingga 6. Nilai *sum* dihitung dari hasil penjumlahan dan program menampilkan hasil lemparan.

```
13              int dadu1 = rand.nextInt(6) + 1;
14              int dadu2 = rand.nextInt(6) + 1;
15              sum = dadu1 + dadu2;
```

Gambar 2.2.2.5 Proses dalam perulangan.

2.2.2.6 Setelah keluar dari perulangan, program menampilkan pesan akhir *"You won after " + tries + " tries!"*, yang berarti program memberi tahu berapa kali pengguna “melempar dadu” sebelum mendapatkan jumlah 7. Program selesai dijalankan setelah menampilkan jumlah percobaan yang diperlukan untuk memperoleh hasil 7 dari dua dadu acak.

```
16              System.out.println(dadu1 + " + " + dadu2 + " = " + sum);
17              tries++;
18          }
19          System.out.println("You won after " + tries + " tries!");
```

Gambar 2.2.2.6 Cetak hasil.

2.2.3 Analisis Hasil

```
5 + 2 = 7
You won after 1 tries!
```

Gambar 2.2.3 Hasil *output* kode program LemparDadu.

Hasil *output* pada gambar menunjukkan bahwa pada lemparan pertama, nilai dua dadu adalah 5 dan 2. Ketika kedua nilai tersebut dijumlahkan, hasilnya 7. Karena kondisi perulangan pada program adalah *while (sum != 7)*, maka ketika sum bernilai 7, perulangan langsung berhenti. Program kemudian menampilkan pesan *"You won after 1 tries!"*, yang berarti hasil penjumlahan 7 sudah didapat pada percobaan pertama. Dengan demikian, analisisnya adalah program berhasil mencapai kondisi kemenangan dalam satu kali percobaan karena jumlah kedua dadu langsung menghasilkan angka 7 pada lemparan pertama.

2.3 SentinelLoop

2.3.1 Kode Program

```java
package pekan6_2511531018;

import java.util.Scanner;

public class SentinelLoop_2511531018 {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        int sum = 0;
        int number=12;  // "dummy value", anything but 0

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

Gambar 2.3.1 Kode program SentinelLoop.

2.3.2 Langkah Kerja

2.3.2.1 Objek *Scanner* dibuat untuk menerima input dari pengguna.

```
7          Scanner console = new Scanner(System.in);
```

Gambar 2.3.2.1 Objek *Scanner*.

2.3.2.2 *Int sum* = 0 digunakan untuk menyimpan hasil penjumlahan dari angka-angka yang dimasukkan pengguna dan *int number* = 12 diberikan nilai awal *"dummy"* (bukan nol) agar perulangan dapat dimulai.

```
8          int sum = 0;
9          int number=12;  // "dummy value", anything but 0
```

Gambar 2.3.2.2 Deklarasi 2 variabel.

2.3.2.3 Program masuk ke perulangan *while*, program menampilkan pesan untuk meminta pengguna memasukkan angka dan nilainya dibaca, lalu nilai yang dimasukkan akan ditambahkan ke variabel *sum*.

```
11         while (number != 0) {
12             System.out.print("Masukkan angka (0 untuk keluar): ");
13             number = console.nextInt();
14             sum = sum + number;
```

Gambar 2.3.2.3 Proses dalam perulangan.

2.3.2.4 Setelah keluar dari perulangan, program menampilkan hasil akhir yaitu jumlah dari semua angka yang dimasukkan sebelum pengguna mengetik 0. Program selesai dijalankan setelah menampilkan total penjumlahan tersebut.

```
16         System.out.println("totalnya adalah " + sum);
```

Gambar 2.3.2.4 Hasil akhir.

2.3.3 Analisis Hasil

```
Masukkan angka (0 untuk keluar): 6
Masukkan angka (0 untuk keluar): 2
Masukkan angka (0 untuk keluar): 9
Masukkan angka (0 untuk keluar): 4
Masukkan angka (0 untuk keluar): 0
totalnya adalah 21
```

Gambar 2.3.3 Hasil *output* kode program SentinelLoop.

Hasil dari program tersebut menunjukkan jumlah total dari semua angka yang dimasukkan oleh pengguna sebelum memasukkan angka 0. Program terus meminta *input* angka dan menjumlahkannya satu per satu menggunakan variabel *sum*. Ketika pengguna mengetik angka 0, program menganggapnya sebagai tanda untuk berhenti, sehingga perulangan berakhir. Setelah itu, program menampilkan hasil akhir berupa total penjumlahan semua angka yang telah dimasukkan sebelumnya. Dengan demikian, *output* program bergantung pada nilai-nilai yang dimasukkan oleh pengguna sebelum angka 0 dimasukkan sebagai penanda untuk keluar dari perulangan.

2.4 GamePenjumlahan

2.4.1 Kode Program

```
public class GamePenjumlahan_2511531018 {
    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); //play one game
            if (result > 0) {
                points++;
            } else {
                wrong++;
            }
        }
        System.out.println("You earned " + points + " total points.");
        }
        //memuat soal penjumlahan dan ditampilkan ke user
        public static int play(Scanner console, Random rand) {
            //print the operands being added, and sum them
            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 wheter it was correct
            int guess = console.nextInt();
            if (guess == sum) {
                return 1;
            } else {
                System.out.println("Wrong! The answer was " + sum);
                return 0;
```

Gambar 2.4.1 Kode program GamePenjumlahan.

2.4.2 Langkah Kerja

2.4.2.1 Objek *Scanner* dibuat untuk membaca *input* dari pengguna dan objek *Random* dibuat untuk menghasilkan angka acak.

```
8           Scanner console = new Scanner(System.in);
9           Random rand = new Random();
```

Gambar 2.4.2.1 Objek *Scanner* dan objek *Random.*

2.4.2.2 *int points* = 0 untuk menghitung jumlah jawaban benar (poin pemain) dan *int wrong* = 0 untuk menghitung jumlah jawaban salah.

```
11          int points =0;
12          int wrong =0;
```

Gambar 2.4.2.2 Deklarasi 2 variabel.

2.4.2.3 Program menjalankan perulangan, program menggunakan *play(console, rand)* untuk menjalankan satu ronde permainan penjumlahan. Jika *result* bernilai lebih dari 0 (artinya jawaban benar), maka *points* akan bertambah satu. Jika tidak, *wrong* bertambah satu. Setelah pengguna menjawab salah sebanyak tiga kali, perulangan berhenti. Program kemudian menampilkan pesan akhir *"You earned " + points + " total points."* untuk menunjukkan total poin yang berhasil diperoleh pengguna.

```
13          while (wrong < 3) {
14              int result = play(console, rand); //play one game
15              if (result > 0) {
16                  points++;
17              } else {
18                  wrong++;
19              }
20          }
21          System.out.println("You earned " + points + " total points.");
```

Gambar 2.4.2.3 Proses perulangan.

2.4.2.4 *Metode play* digunakan untuk membuat dan menampilkan soal penjumlahan acak.

```
24          public static int play(Scanner console, Random rand) {
```

Gambar 2.4.2.4 Menggunakan *metode play.*

2.4.2.5 Jumlah angka yang akan dijumlahkan ditentukan secara acak, sehingga jumlah *operand* bisa antara 2 hingga 5 angka. Kemudian, angka pertama dibuat acak dan disimpan dalam variabel *sum*. Angka tersebut juga ditampilkan di layar.

```
26              int operands = rand.nextInt(4) + 2;
27              int sum = rand.nextInt(10) + 1;
28              System.out.print(sum);
```

Gambar 2.4.2.5 Penjumlahan angka secara acak.

2.4.2.6 Program menggunakan perulangan *for* untuk menghasilkan angka-angka tambahan, menambahkannya ke *sum*, dan menampilkannya dengan tanda "+".

```
29              for (int i =2; i<= operands; i++) {
30                  int n = rand.nextInt(10) + 1;
31                  sum += n;
32                  System.out.print(" + " + n);
33              }
34              System.out.print(" = ");
```

Gambar 2.4.2.6 Menggunakan perulangan *for*.

2.4.2.7 Setelah semua angka ditampilkan, program menunggu *input* dari pengguna yang mewakili hasil penjumlahan, program kemudian memeriksa apakah jawaban pengguna sama dengan nilai *sum*.

```
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;
```

Gambar 2.4.2.7 Mencetak hasil.

2.4.3 Analisis Hasil

```
2 + 8 + 3 + 5 = 18
2 + 4 + 5 + 4 = 54
Wrong! The answer was 15
4 + 2 + 6 + 5 + 10 =
```

Gambar 2.4.3 Hasil output kode program GamePenjumlahan.

2.5 doWhile

2.5.1 Kode Program

```
1  package pekan6_2511531018;
2
3  import java.util.Scanner;
4
5  public class doWhile_2511531018 {
6      public static void main(String[] args) {
7          Scanner console = new Scanner(System.in);
8          String phrase;
9          do {
10             System.out.print("Input Password: ");
11             phrase = console.next();
12         } while (!phrase.equals("abcd"));
13     }
14 }
15
```

Gambar 2.5.1 Kode program doWhile.

2.5.2 Langkah Kerja

2.5.2.1 Objek *Scanner console* dibuat untuk menerima *input* dari pengguna.

```
7          Scanner console = new Scanner(System.in);
```

Gambar 2.5.2.1 Objek *Scanner*.

2.5.2.2 Variabel *String phrase* dideklarasikan untuk menyimpan *input* yang dimasukkan oleh pengguna.

```
8          String phrase;
```

Gambar 2.5.2.2 Deklarasi variabel.

2.5.2.3 Program menggunakan perulangan *do-while*, Di dalam blok *do*, program menampilkan teks *"Input Password: "* ke layar, lalu menunggu pengguna mengetikkan sebuah kata. Setelah pengguna mengetik dan menekan *Enter*, program memeriksa apakah *input* tersebut sama dengan *string "abcd"*.

```
9●          do {
10              System.out.print("Input Password: ");
11              phrase = console.next();
12          } while (!phrase.equals("abcd"));
```

Gambar 2.5.2.3 Menggunakan program *do-while*.

2.5.1 Analisis Hasil

```
Input Password: 2345
Input Password: 6743
Input Password: 8hs3
Input Password: abcd
```

Gambar 2.5.1 Hasil *output* kode program doWhile.

Program berfungsi sebagai sistem verifikasi sederhana yang meminta pengguna memasukkan *password* hingga sesuai dengan nilai yang telah ditentukan, yaitu *"abcd"*. Saat program dijalankan, pengguna akan selalu diminta untuk mengetik *password* melalui pesan *"Input Password:"*. Jika pengguna mengetikkan kata yang berbeda dari *"abcd",* program akan terus mengulang dan meminta *input* ulang tanpa henti. Perulangan hanya berhenti ketika pengguna akhirnya memasukkan kata *"abcd"* dengan benar. Setelah itu, program berhenti tanpa menampilkan pesan tambahan. Dengan demikian, hasil program menegaskan bahwa hanya *input* yang tepat, yaitu *"abcd",* yang dapat menghentikan proses perulangan, sedangkan *input* lain dianggap salah dan menyebabkan program terus berjalan.

KESIMPULAN

Seluruh kode program yang telah dilakukan merupakan contoh-contoh penerapan konsep perulangan *(looping)* dan pengendalian alur program menggunakan struktur seperti *while, do-while, dan for*, serta penerapan kondisi logika untuk mengatur kapan program berhenti atau melanjutkan prosesnya.

Secara keseluruhan, semua program tersebut menggambarkan konsep dasar dalam pengulangan dan logika pemrograman Java. Masing-masing menunjukkan variasi kondisi dan cara penghentian perulangan—baik berdasarkan *input* pengguna, hasil acak, maupun nilai sentinel. Program-program ini juga memperlihatkan peran penting variabel kontrol seperti penghitung *(counter, tries, points),* penggunaan objek *Scanner* untuk input, serta objek *Random* untuk menghasilkan nilai acak. Dari seluruhnya dapat disimpulkan bahwa konsep perulangan sangat penting dalam membuat program interaktif dan dinamis yang mampu merespons kondisi atau masukan pengguna secara fleksibel.

DAFTAR PUSTAKA

[1] H. M. Deitel dan P. J. Deitel, *Java: How to Program*, 11th ed. Pearson Education, 2017.

[2] E. Balagurusamy, *Pemrograman Berorientasi Objek dengan Java*. Yogyakarta: Andi Offset, 2010.

[3] R. A. Johnson, *Dasar-Dasar Pemrograman Java*. Jakarta: Informatika, 2019.

[4] W. Sunarto, *Algoritma dan Pemrograman dengan Java*. Bandung: Informatika, 2020.

[5] D. Setiawan, *Logika dan Struktur Data dalam Pemrograman Java*. Yogyakarta: Deepublish, 2021.