

**LAPORAN PRAKTIKUM**  
**ALGORITMA DAN PEMOGRAMAN**  
**PERULANGAN WHILE DAN DO-WHILE**  
**PEKAN 6**

Disusun Oleh :

KINAYA NOVRYA MANDA  
(2511531016)

Dosen Pengampu :

DR. WAHYUDI, S.T, M.T

Asisten Praktikum :

MUHAMMAD ZAKI AL HAFIZ



DEPARTEMEN INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS ANDALAS

2025

## **KATA PENGANTAR**

Puji syukur penulis sampaikan kehadirat Allah SWT. Salawat dan salam disampaikan kepada Nabi Muhammad SAW. Karena thaufik dan hidayah-Nya, laporan praktikum Java ini dapat diselesaikan dengan baik dan tepat waktu. Laporan ini disusun sebagai salah satu bentuk pertanggungjawaban dari kegiatan praktikum yang telah dilaksanakan, sekaligus sebagai sarana untuk memperdalam pemahaman mengenai konsep dasar serta penerapan bahasa pemrograman Java.

Melalui praktikum ini, penulis memperoleh pengalaman langsung dalam memahami struktur perulangan yang memungkinkan program membuat keputusan dan menjalankan blok kode yang berbeda berdasarkan kondisi tertentu menggunakan Java. Diharapkan laporan ini dapat memberikan gambaran yang jelas mengenai materi yang telah dipelajari serta hasil dari percobaan yang dilakukan selama praktikum berlangsung.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, baik dari segi isi maupun penyajiannya. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa yang akan datang.

Akhir kata, penulis mengucapkan terima kasih kepada dosen pengampu, asisten laboratorium, serta semua pihak yang telah memberikan bimbingan, arahan, dan dukungan sehingga laporan praktikum ini dapat tersusun. Semoga laporan ini dapat bermanfaat bagi pembaca, khususnya dalam memahami dasar-dasar pemrograman Java.

Padang, Oktober 2025

Penulis

## DAFTAR ISI

<b>KATA PENGANTAR.....</b>	<b>i</b>
<b>DAFTAR ISI .....</b>	<b>ii</b>
<b>DAFTAR LAMPIRAN .....</b>	<b>iii</b>
<b>BAB 1 PENDAHULUAN</b>	
1.1 Latar Belakang.....	1
1.2 Tujuan Praktikum.....	2
1.3 Manfaat Praktikum.....	2
<b>BAB 2 PEMBAHASAN</b>	
2.1 Kode Program.....	3
2.2 Langkah Kerja .....	9
2.3 Analisis Hasil.....	18
<b>BAB 3 KESIMPULAN</b>	
3.1 Hasil Praktikum .....	19
3.2 Saran Pengembangan .....	22
<b>DAFTAR KEPUSTAKAAN .....</b>	<b>24</b>

## DAFTAR LAMPIRAN

Kode Program 2.1 (Kode Program perulanganWhile1) .....	3
Kode Program 2.2 (Kode Program Lempardadu) .....	5
Kode Program 2.3 (Kode Program GamePenjumlahan).....	6
Kode Program 2.4 (Kode Program SentinelLoop) .....	7
Kode Program 2.5 (Kode Program doWhile1).....	8
Gambar 2.1 (Langkah 1 perulanganWhile1).....	9
Gambar 2.2 (Langkah 2 perulanganWhile1).....	10
Gambar 2.3 (Langkah 3 perulanganWhile1).....	10
Gambar 2.4 (Langkah 1 perulanganWhile1).....	10
Gambar 2.5 (Langkah 2 perulanganWhile1).....	11
Gambar 2.6 (Langkah 3 Lempardadu).....	11
Gambar 2.7 (Langkah 1 Lempardadu).....	11
Gambar 2.8 (Langkah 2 Lempardadu).....	12
Gambar 2.9 (Langkah 3 Lempardadu).....	12
Gambar 2.10 (Langkah 4 GamePenjumlahan) .....	13
Gambar 2.11 (Langkah 1 GamePenjumlahan) .....	13
Gambar 2.12 (Langkah 2 GamePenjumlahan) .....	13
Gambar 2.13 (Langkah 3 GamePenjumlahan) .....	14
Gambar 2.14 (Langkah 4 GamePenjumlahan) .....	14
Gambar 2.15 (Langkah 1 GamePenjumlahan).....	14
Gambar 2.16 (Langkah 2 GamePenjumlahan) .....	15
Gambar 2.17 (Langkah 3 SentinelLoop) .....	15
Gambar 2.18 (Langkah 4 SentinelLoop) .....	15

Gambar 2.19 (Langkah 1 SentinelLoop) .....	16
Gambar 2.20 (Langkah 2 SentinelLoop) .....	16
Gambar 2.21 (Langkah 3 doWhile1) .....	17
Gambar 2.22 (Langkah 4 doWhile1) .....	17
Gambar 2.23 (Langkah 1 doWhile1) .....	17
Gambar 2.24 (Langkah 2 doWhile1) .....	18
Gambar 3.1 (Hasil Praktikum perulanganWhile1) .....	19
Gambar 3.2 (Hasil Praktikum Lempardadu) .....	20
Gambar 3.3 (Hasil Praktikum GamePenjumlahan) .....	20
Gambar 3.4 (Hasil Praktikum SentinelLoop).....	21
Gambar 3.5 (Hasil Praktikum doWhile1) .....	22

## BAB 1

### PENDAHULUAN

#### 1.1 Latar Belakang

Dalam dunia pemrograman, efisiensi dan efektivitas dalam menulis kode sangat bergantung pada kemampuan untuk mengelola proses yang berulang. Salah satu konsep fundamental yang mendukung hal tersebut adalah perulangan (*looping*). Melalui perulangan, sebuah program dapat menjalankan instruksi yang sama secara berulang hingga suatu kondisi tertentu terpenuhi tanpa harus menuliskan kode berulang kali. Bahasa pemrograman Java menyediakan beberapa struktur perulangan, di antaranya adalah *while* dan *do-while loop*.

Struktur *while loop* digunakan ketika jumlah pengulangan belum diketahui secara pasti dan program perlu memeriksa kondisi terlebih dahulu sebelum menjalankan perintah di dalamnya. Sementara itu, *do-while loop* memiliki perbedaan utama pada cara pengecekan kondisinya — perintah di dalam blok akan dieksekusi terlebih dahulu minimal satu kali sebelum kondisi diperiksa. Hal ini menjadikan *do-while* sangat cocok digunakan dalam situasi di mana tindakan harus dilakukan setidaknya sekali, seperti meminta input dari pengguna.

Melalui praktikum ini, mahasiswa diharapkan mampu memahami dan membedakan cara kerja kedua jenis perulangan tersebut, serta mengetahui penerapannya dalam berbagai kasus nyata. Pemahaman mendalam terhadap konsep perulangan sangat penting karena menjadi dasar dari logika pemrograman, terutama dalam pembuatan program interaktif, pengolahan data berulang, maupun algoritma komputasi yang kompleks. Dengan memahami konsep *while* dan *do-while*, mahasiswa dapat menulis program yang lebih efisien, terstruktur, dan mudah dikembangkan di kemudian hari.

## 1.2 Tujuan Praktikum

Tujuan dilakukannya praktikum ini adalah sebagai berikut :

1. Memahami konsep dasar perulangan (*looping*) dalam bahasa Java.
2. Mengetahui perbedaan antara perulangan *while* dan *do-while*.
3. Mampu menerapkan struktur perulangan untuk menyelesaikan berbagai kasus logika berulang.
4. Melatih kemampuan berpikir logis dan sistematis dalam menulis program.
5. Mengidentifikasi situasi yang tepat untuk menggunakan *while* maupun *do-while* dalam suatu program.

## 1.3 Manfaat Praktikum

Manfaat dilakukannya praktikum ini adalah sebagai berikut :

1. Mahasiswa dapat membuat program yang efisien dengan memanfaatkan perulangan.
2. Meningkatkan kemampuan analisis terhadap kondisi dan logika pengulangan.
3. Mempermudah pembuatan program interaktif seperti input berulang atau simulasi sederhana.
4. Menjadi dasar dalam memahami konsep lanjutan seperti *nested loop* dan *loop control statements*.
5. Membantu mahasiswa dalam mengembangkan kemampuan pemrograman terstruktur dan terarah.

## BAB II

### PEMBAHASAN

#### 2.1 Kode Program

##### 2.1.1 perulanganWhile1

```

1 package Pekan6_2511531016;
2 import java.util.Scanner;
3 public class perulanganWhile1_2511531016 {
4
5     public static void main(String[] args) {
6         int counter=0;
7         String jawab;
8         boolean running = true;
9         //deklarasi scanner
10        Scanner scan = new Scanner(System.in);
11        while (running) {
12            counter++;
13            System.out.println("Jumlah = "+counter);
14            System.out.print("Apakah lanjut (ya / tidak?)");
15            jawab= scan.nextLine();
16            //cek jawab = tidak, perulangan berhenti
17            if (jawab.equalsIgnoreCase("tidak")) {
18                running= false;
19            }
20        }
21        System.out.println("Anda sudah melakukan perulangan sebanyak "+counter+" kali");
22    }
23 }
24 }

```

**Kode Program 2.1 (Kode Program perulanganWhile1)**

#### Uraian Kode Program :

1. *package Pekan6\_2511531016;* → Menunjukkan program ini berada dalam *package* bernama *Pekan6\_2511531016*.
2. *import java.util.Scanner;* → Mengimpor kelas *Scanner* yang digunakan untuk menerima input dari pengguna.
3. *public class perulanganWhile1\_2511531016* → Mendefinisikan kelas utama dengan nama *perulanganWhile1\_2511531016*.
4. *public static void main(String[] args)* → Merupakan metode utama yang dijalankan pertama kali saat program dieksekusi.
5. *int counter = 0;* → Mendeklarasikan variabel *counter* untuk menghitung berapa kali perulangan terjadi.

6. *String jawab;* → Mendeklarasikan variabel *jawab* untuk menyimpan jawaban dari pengguna.
7. *boolean running = true;* → Variabel *running* digunakan sebagai penanda agar perulangan *while* tetap berjalan selama nilainya *true*.
8. *Scanner scan = new Scanner(System.in);* → Membuat objek *Scanner* untuk membaca input dari pengguna melalui *keyboard*.
9. *while (running)* → Menjalankan perulangan selama nilai *running* bernilai *true*.
10. *counter++;* → Menambah nilai *counter* setiap kali perulangan dijalankan.
11. *System.out.println("Jumlah = " + counter);* → Menampilkan jumlah perulangan yang sudah dilakukan.
12. *System.out.println("Apakah lanjut (ya / tidak)?");* → Menampilkan pertanyaan kepada pengguna untuk menentukan apakah perulangan dilanjutkan atau tidak.
13. *jawab = scan.nextLine();* → Membaca input dari pengguna dan menyimpannya ke dalam variabel *jawab*.
14. *if (jawab.equalsIgnoreCase("tidak")) { running = false; }* → Jika pengguna mengetik “tidak”, maka nilai *running* diubah menjadi *false* sehingga perulangan berhenti.
15. *System.out.println("Anda sudah melakukan perulangan sebanyak " + counter + " kali");* → Menampilkan total berapa kali perulangan telah dilakukan setelah program berhenti.

### 2.1.2 Lempardadu

```

1 package Pekan6_2511531016;
2 import java.util.Random;
3 public class Lempardadu_2511531016 {
4
5     public static void main(String[] args) {
6         Random rand = new Random();
7         int tries = 0;
8         int sum = 0;
9         while (sum != 7) {
10             // roll the dice once
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 }
20 }

```

**Kode Program 2.2 (Kode Program Lempardadu)**

#### Uraian Kode Program :

1. *package Pekan6\_2511531016;* → Menunjukkan program berada di paket *Pekan6\_2511531016*.
2. *import java.util.Random;* → Mengimpor kelas *Random* untuk menghasilkan angka acak.
3. *public class Lempardadu\_2511531016* → Mendefinisikan kelas utama program.
4. *public static void main(String[] args)* → Metode utama yang dijalankan pertama kali.
5. *Random rand = new Random();* → Membuat objek untuk menghasilkan angka acak.
6. *int tries = 0;*, *int sum = 0;* → Menyimpan jumlah percobaan dan hasil penjumlahan dadu.
7. *while (sum != 7)* → Perulangan berjalan hingga total dua dadu bernilai 7.
8. *int dadu1 dan dadu2 = rand.nextInt(6) + 1;* → Menghasilkan angka acak antara 1–6 untuk masing-masing dadu.
9. *sum = dadu1 + dadu2;* → Menjumlahkan nilai kedua dadu.

10. `System.out.println(...);` → Menampilkan hasil lemparan dan totalnya.
11. `tries++;` → Menambah jumlah percobaan setiap kali lemparan dilakukan.
12. `System.out.println("You won after...");` → Menampilkan jumlah percobaan saat hasil 7 tercapai.

### 2.1.3 GamePenjumlahan

```

1 package Pekan6_2511531016;
2
3 import java.util.Random;
4
5
6 public class GamePenjumlahan_2511531016 {
7     public static void main(String[] args) {
8         Scanner console = new Scanner(System.in);
9         Random rand = new Random();
10        //play until user gets 3 wrong
11        int points = 0;
12        int wrong = 0;
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.");
22    }
23    //memuat soal penjumlahan dan ditampilkan ke user
24    public static int play(Scanner console, Random rand) {
25        //print the operands being added, and sum them
26        int operands = rand.nextInt(4) + 2;
27        int sum = rand.nextInt(10) + 1;
28        System.out.print(sum);
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(" = ");
35
36        //read user's guess and report wheter it was correct
37
38        int guess = console.nextInt();
39        if (guess == sum) {
40            return 1;
41        } else {
42            System.out.println("Wrong! The answer was " + sum);
43            return 0;
44        }
45    }
46 }
47
48

```

**Kode Program 2.3 (Kode Program GamePenjumlahan)**

### Uraian Kode Program :

1. *package Pekan6\_2511531016;* → Menunjukkan bahwa program ini berada dalam paket bernama *Pekan6\_2511531016*.
2. *import java.util.Random;* → Mengimpor kelas *Random* untuk menghasilkan angka acak.
3. *public class GamePenjumlahan\_2511531016* → Mendefinisikan kelas utama program dengan nama *GamePenjumlahan\_2511531016*.
4. *Scanner console = new Scanner(System.in);* → Membuat objek *Scanner* untuk menerima input dari pengguna.
5. *Random rand = new Random();* → Membuat objek *Random* untuk menghasilkan soal penjumlahan acak.
6. *while (wrong < 3)* → Perulangan berlangsung sampai pengguna menjawab salah sebanyak 3 kali.
7. *play(console, rand);* → Memanggil metode *play* untuk menampilkan soal dan memeriksa jawaban pengguna.
8. *if (guess == sum)* → Mengecek apakah jawaban pengguna benar; jika benar mendapat poin, jika salah menambah jumlah kesalahan.
9. *System.out.println("You earned " + points + " total points.");* → Menampilkan jumlah poin yang diperoleh pengguna di akhir permainan.

### 2.1.4 SentinelLoop

```

1 package Pekan6_2511531016;
2 import java.util.Scanner;
3 public class SentinelLoop_2511531016 {
4
5     public static void main(String[] args) {
6         Scanner console = new Scanner(System.in);
7         int sum = 0;
8         int number=12;        // "dummy value", anything but 0
9
10        while (number != 0) {
11            System.out.print("Masukkan angka (0 untuk keluar) : ");
12            number = console.nextInt();
13            sum = sum + number;
14        }
15        System.out.println("totalnya adalah " + sum) ;
16    }
17
18 }
```

**Kode Program 2.4 (Kode Program SentinelLoop)**

### Uraian Kode Program :

1. *package Pekan6\_2511531016;* → Menandakan program berada di paket *Pekan6\_2511531016*.
2. *import java.util.Scanner;* → Mengimpor kelas *Scanner* untuk membaca input dari pengguna.
3. *public class SentinelLoop\_2511531016* → Mendefinisikan kelas utama dengan nama *SentinelLoop\_2511531016*.
4. *int sum = 0;* → Variabel untuk menyimpan total penjumlahan.
5. *int number = 12;* → Nilai awal agar perulangan bisa dimulai.
6. *while (number != 0)* → Perulangan berjalan selama nilai *number* bukan 0.
7. *System.out.print("Masukkan angka...")* → Meminta pengguna memasukkan angka.
8. *sum = sum + number;* → Menambahkan angka yang dimasukkan ke total.
9. *System.out.println("totalnya adalah " + sum);* → Menampilkan hasil total penjumlahan setelah pengguna memasukkan angka 0.

### 2.1.5 doWhile1

```

1  package Pekan6_2511531016;
2  import java.util.Scanner;
3  public class doWhile1_2511531016 {
4
5      public static void main(String[] args) {
6          Scanner console = new Scanner(System.in);
7          String phrase;
8          do {
9              System.out.print("Input Password: ");
10             phrase = console.next();
11         } while (!phrase.equals("abcd"));
12     }
13 }
14
15 }
```

Kode Program 2.5 (Kode Program doWhile1)

### Uraian Kode Program :

1. *package Pekan6\_2511531016;* → Menunjukkan program berada dalam paket *Pekan6\_2511531016*.
2. *import java.util.Scanner;* → Mengimpor kelas *Scanner* untuk membaca input dari pengguna.
3. *public class doWhile1\_2511531016* → Mendefinisikan kelas utama dengan nama *doWhile1\_2511531016*.
4. *Scanner console = new Scanner(System.in);* → Membuat objek *Scanner* untuk membaca input dari *keyboard*.
5. *String phrase;* → Mendeklarasikan variabel untuk menyimpan input pengguna.
6. *do { ... } while (!phrase.equals("abcd"));* → Melakukan perulangan untuk meminta input password dan akan berhenti hanya jika pengguna mengetik "abcd"

## 2.2 Langkah Kerja

### 2.2.1 perulanganWhile1

1. Membuat kelas baru bernama *perulanganWhile1* dalam *package Pekan6\_2511531016*.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static  
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass:

Interfaces:

Which method stubs would you like to create?  
☒ public static void main(String[] args)  
☐ Constructors from superclass  
☐ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

**Gambar 2.1 (Langkah 1 perulanganWhile1)**

2. Program dijalankan dan masuk ke *main*.

```
public static void main(String[] args) {
```

**Gambar 2.2 (Langkah 2 perulanganWhile1)**

3. Deklarasikan beberapa variabel, yaitu *counter* untuk menghitung jumlah perulangan, *jawab* untuk menyimpan input pengguna, dan *running* yang diatur bernilai *true* agar perulangan bisa dimulai. Selain itu, buat juga objek *Scanner* dengan perintah *Scanner scan = new Scanner(System.in);* yang berfungsi untuk membaca input dari *keyboard*.

```
int counter=0;
String jawab;
boolean running = true;
//deklarasi scanner
Scanner scan = new Scanner(System.in);
```

**Gambar 2.3 (Langkah 3 perulanganWhile1)**

4. *While (running)* program menjalankan perulangan selama nilai *running* masih *true*. Di dalamnya, *counter++* menambah jumlah perulangan, lalu program menampilkan pesan *Jumlah = ...* dan menanyakan “Apakah lanjut (ya/tidak)?” kepada pengguna.

```
while (running) {
    counter++;
    System.out.println("Jumlah = "+counter);
    System.out.print("Apakah lanjut (ya / tidak?)");
    jawab= scan.nextLine();
```

**Gambar 2.4 (Langkah 4 perulanganWhile1)**

5. Jika pengguna mengetik “tidak”, maka pada bagian *if* nilai *running* diubah menjadi *false*, sehingga perulangan berhenti. Terakhir, program

menampilkan hasil “*Anda sudah melakukan perulangan sebanyak ... kali*” sebagai penutup setelah *loop* selesai dijalankan.

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

**Gambar 2.5 (Langkah 5 perulanganWhile1)**

## 2.2.2 Lempardadu

1. Membuat kelas baru bernama *Lempardadu* dalam *package* *Pekan6\_2511531016*.

Source folder:  Browse...

Package:  Browse...

☐ Enclosing type:  Browse...

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static  
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass:  Browse...

Interfaces:  Add... Remove

Which method stubs would you like to create?  
☒ public static void main(String[] args)  
☐ Constructors from superclass  
☐ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

**Gambar 2.6 (Langkah 1 Lempardadu)**

2. Program dijalankan dan masuk ke *main*.

```
public static void main(String[] args) {
```

**Gambar 2.7 (Langkah 2 Lempardadu)**

3. Buat objek *Random* untuk menghasilkan angka acak, dan variabel *tries* untuk menghitung jumlah percobaan.

```
Random rand = new Random();
int tries = 0;
int sum = 0;
```

**Gambar 2.8 (Langkah 3 Lempardadu)**

4. Program akan melempar dua dadu acak dan menjumlahkan hasilnya. Jika hasilnya bukan 7, perulangan diulang dan *tries* bertambah.

```
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 2.9 (Langkah 4 Lempardadu)**

### 2.2.3 GamePenjumlahan

1. Membuat kelas baru bernama *GamePenjumlahan* dalam *package Pekan6\_2511531016*.

The screenshot shows the 'New Class' dialog in Eclipse. The 'Source folder' is 'Prakalpro\_2025\_B\_2511531016/src', the 'Package' is 'Pekan6\_2511531016', and the 'Name' is 'GamePenjumlahan\_2511531016'. The 'Modifiers' section has 'public' selected. The 'Superclass' is 'java.lang.Object'. The 'Interfaces' section is empty. The 'Which method stubs would you like to create?' section has 'public static void main(String[] args)' checked. The 'Do you want to add comments?' section has 'Generate comments' checked.

**Gambar 2.10 (Langkah 1 GamePenjumlahan)**

2. Program dijalankan dan masuk ke *main*.

```
public static void main(String[] args) {
```

**Gambar 2.11 (Langkah 2 GamePenjumlahan)**

3. Buat objek *Scanner* dan *Random*, serta variabel *points* untuk menghitung skor benar dan *wrong* untuk menghitung jumlah jawaban salah.

```
Scanner console = new Scanner(System.in);
Random rand = new Random();
//play until user gets 3 wrong
int points =0;
int wrong =0;
```

**Gambar 2.12 (Langkah 3 GamePenjumlahan)**

4. Program akan terus berjalan selama jumlah jawaban salah kurang dari 3. Jika pemain menjawab benar (*result* > 0), *poin* bertambah satu, namun jika salah, *wrong* bertambah satu.

```
while (wrong < 3) {
    int result = play(console, rand); //play one game
    if (result > 0) {
        points++;
    } else {
        wrong++;
    }
}
```

**Gambar 2.13 (Langkah 4 GamePenjumlahan)**

5. Setelah pemain salah 3 kali, perulangan berhenti dan total *poin* yang didapat ditampilkan.

```
System.out.println("You earned " + points + " total points.");
}
```

**Gambar 2.14 (Langkah 5 GamePenjumlahan)**

6. *Method play()* membuat soal penjumlahan acak untuk pengguna. Program menentukan jumlah angka yang dijumlahkan, menampilkan angka pertama, lalu menambahkan beberapa angka acak lainnya menggunakan perulangan *for*, dan mencetak tanda “=” di akhir sebagai tempat jawaban.

```
//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(" = ");
}
```

**Gambar 2.15 (Langkah 6 GamePenjumlahan)**

7. Program memeriksa jawaban pengguna. Jika *guess* sama dengan *sum*, program memberi nilai 1 (benar), jika tidak menampilkan jawaban yang benar dan memberi nilai 0 (salah).

```
//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 2.16 (Langkah 7 GamePenjumlahan)**

## 2.2.4 SentinelLoop

1. Membuat kelas baru bernama *SentinelLoop* dalam *packagePekan6\_2511531016*.

Source folder: Prakalpro\_2025\_B\_2511531016/src

Package: Pekan5

☐ Enclosing type:

Name: PerulanganFor4

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static  
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?  
☒ public static void main(String[] args)  
☐ Constructors from superclass  
☐ Inherited abstract methods

**Gambar 2.17 (Langkah 1 SentinelLoop)**

2. Program dijalankan dan masuk ke *main*.

```
public static void main(String[] args) {
```

**Gambar 2.18 (Langkah 2 SentinelLoop)**

3. Deklarasikan beberapa variabel yaitu, *sum* untuk menampung total, *number* sebagai nilai awal, dan *console* untuk membaca input.

```
Scanner console = new Scanner(System.in);
int sum = 0;
int number=12;      // "dummy value", anything but 0
```

**Gambar 2.19 (Langkah 3 SentinelLoop)**

4. Perulangan dijalankan selama *number* tidak sama dengan 0. Program meminta pengguna memasukkan angka dan dijumlahkan kedalam *sum*. Setelah pengguna memasukkan angka 0, perulangan berhenti dan total jumlah seluruh angka ditampilkan.

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

**Gambar 2.20 (Langkah 4 SentinelLoop)**

### 2.2.5 doWhile1

1. Membuat kelas baru bernama *doWhile1* dalam *package Pekan6\_2511531016*.

Source folder: Prakalpro\_2025\_B\_2511531016/src Browse...

Package: Pekan6\_2511531016 Browse...

☐ Enclosing type: Browse...

Name: doWhile1\_2511531016

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static  
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?  
☒ public static void main(String[] args)  
☐ Constructors from superclass  
☐ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

**Gambar 2.21 (Langkah 1 doWhile1)**

2. Program dijalankan dan masuk ke *main*.

```
public static void main(String[] args) {
```

**Gambar 2.22 (Langkah 2 doWhile1)**

3. Buat objek *Scanner* untuk membaca input dan variabel *phrase* untuk menyimpan *password* yang dimasukkan.

```
Scanner console = new Scanner(System.in);  
String phrase;
```

**Gambar 2.23 (Langkah 3 doWhile1)**

4. Program meminta input *password* dan akan terus mengulang selama pengguna belum mengetikkan “abcd”. Jika sudah benar, perulangan akan berhenti.

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

**Gambar 2.24 (Langkah 4 doWhile1)**

### 2.3 Analisis Hasil

Berdasarkan hasil percobaan dari kelima kode program yang telah dijalankan, menunjukkan penerapan konsep struktur kendali perulangan dan percabangan dalam bahasa pemrograman Java. Berdasarkan teori dasar pemrograman, perulangan digunakan untuk mengeksekusi perintah berulang kali selama kondisi tertentu terpenuhi (Schildt, 2018).

Pada program PerulanganWhile1, konsep *while loop* digunakan untuk mengulangi proses hingga pengguna memilih berhenti. Program LemparDadu menerapkan *while loop* untuk mensimulasikan lemparan dua dadu hingga hasilnya berjumlah 7, menggambarkan prinsip pengulangan berbasis kondisi acak. Sementara itu, GamePenjumlahan menggunakan kombinasi *while*, *for*, dan *if-else* untuk membuat permainan interaktif berbasis logika penjumlahan acak, yang memperlihatkan penerapan kontrol alur program secara dinamis.

Program SentinelLoop menggunakan nilai *sentinel* (angka 0) sebagai tanda berhenti, sesuai teori *loop termination* yang digunakan untuk menjumlahkan input pengguna secara berulang. Terakhir, DoWhile1 menunjukkan konsep *post-test loop*, yaitu perulangan yang dijalankan minimal sekali sebelum pengecekan kondisi, dalam hal ini digunakan untuk meminta input password hingga benar.

Secara keseluruhan, kelima program ini memperlihatkan bagaimana teori dasar kontrol alur, seperti *looping*, dapat digunakan secara efektif untuk membangun logika interaktif pada aplikasi Java sederhana.

## BAB III KESIMPULAN

### 3.1 Hasil Praktikum

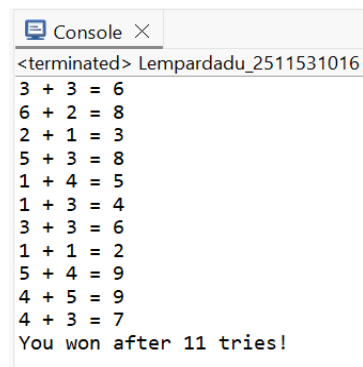
#### 3.1.1 perulanganWhile1

```
<terminated> perulanganWhile1_2511531016 [Java Application] C:\
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 3.1 (Hasil Praktikum perulanganWhile1)**

Output di atas menunjukkan hasil program Java yang menggunakan perulangan *while* untuk menghitung jumlah pengulangan berdasarkan input pengguna. Pada setiap tahap, program menampilkan nilai “Jumlah” dan menanyakan apakah ingin melanjutkan perulangan atau tidak. Jika pengguna mengetik “ya”, perulangan berlanjut dan nilai “Jumlah” bertambah satu. Proses ini berulang hingga pengguna mengetik “tidak”, lalu program menampilkan pesan bahwa perulangan dilakukan sebanyak 4 kali sesuai jumlah input “ya” yang diberikan sebelumnya.

### 3.1.2 Lempardadu



```

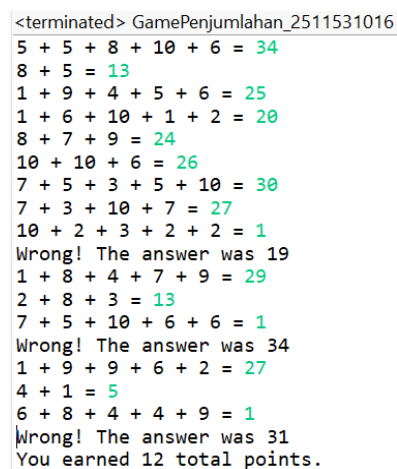
Console X
<terminated> Lempardadu_2511531016
3 + 3 = 6
6 + 2 = 8
2 + 1 = 3
5 + 3 = 8
1 + 4 = 5
1 + 3 = 4
3 + 3 = 6
1 + 1 = 2
5 + 4 = 9
4 + 5 = 9
4 + 3 = 7
You won after 11 tries!

```

**Gambar 3.2 (Hasil Praktikum Lempardadu)**

Output di atas menunjukkan hasil dari sebuah program permainan sederhana yang melibatkan pelemparan dua dadu, kemudian menjumlahkan hasilnya pada setiap percobaan. Setiap baris menampilkan dua angka dari lemparan dadu beserta hasil penjumlahan keduanya. Permainan ini diulang sebanyak 11 kali hingga akhirnya muncul pesan *"You won after 11 tries!"*, yang menandakan bahwa kondisi kemenangan tercapai pada percobaan ke-11.

### 3.1.3 GamePenjumlahan



```

<terminated> GamePenjumlahan_2511531016
5 + 5 + 8 + 10 + 6 = 34
8 + 5 = 13
1 + 9 + 4 + 5 + 6 = 25
1 + 6 + 10 + 1 + 2 = 20
8 + 7 + 9 = 24
10 + 10 + 6 = 26
7 + 5 + 3 + 5 + 10 = 30
7 + 3 + 10 + 7 = 27
10 + 2 + 3 + 2 + 2 = 1
Wrong! The answer was 19
1 + 8 + 4 + 7 + 9 = 29
2 + 8 + 3 = 13
7 + 5 + 10 + 6 + 6 = 1
Wrong! The answer was 34
1 + 9 + 9 + 6 + 2 = 27
4 + 1 = 5
6 + 8 + 4 + 4 + 9 = 1
Wrong! The answer was 31
You earned 12 total points.

```

**Gambar 3.3 (Hasil Praktikum GamePenjumlahan)**

Output di atas merupakan hasil dari sebuah permainan penjumlahan angka di mana pemain harus menghitung hasil penjumlahan beberapa angka yang diberikan dalam satu baris. Jika jawaban pemain benar, maka hasil penjumlahan akan ditampilkan di sebelah kanan dan pengguna mendapatkan poin. Jika jawaban salah, maka akan muncul pesan *"Wrong! The answer was ..."* yang berisi jawaban yang benar. Pada akhir permainan, total *poin* yang didapat pemain sebanyak 12, yang menunjukkan jumlah soal yang dijawab dengan benar.

### 3.1.4 SentinelLoop

```
<terminated> SentinelLoop_2511531016 [Java App
Masukkan angka (0 untuk keluar) : 7
Masukkan angka (0 untuk keluar) : 1
Masukkan angka (0 untuk keluar) : 8
Masukkan angka (0 untuk keluar) : 0
totalnya adalah 16
```

**Gambar 3.4 (Hasil Praktikum SentinelLoop)**

Output tersebut menunjukkan bahwa pengguna memasukkan beberapa angka secara berurutan, dimulai dengan angka 7, kemudian 1, kemudian 8, dan akhirnya 0. Setiap kali pengguna memasukkan angka, sistem menjumlahkan angka-angka tersebut secara akumulatif, kecuali angka 0 yang digunakan sebagai indikator untuk keluar dari proses penginputan. Setelah pengguna memasukkan angka 0, sistem menampilkan hasil total, yaitu 16.

### 3.1.5 doWhile1

```
<terminated> doWhile1_2511531016
Input Password: kinaya123
Input Password: novrya2008
Input Password: abcd
```

**Gambar 3.5 (Hasil Praktikum doWhile1)**

Output tersebut menunjukkan serangkaian input *password* yang dimasukkan oleh pengguna, yaitu *kinaya123*, *novrya2008*, dan *abcd*. Dalam program ini, *password* *abcd* berfungsi sebagai indikator untuk keluar atau menghentikan proses input *password*. Artinya, sistem akan terus menerima input *password* selama *password* yang dimasukkan bukan *abcd*. Ketika pengguna memasukkan *abcd*, proses input berhenti sebagai tanda keluar dari program. *Password* lainnya, seperti *kinaya123* dan *novrya2008*, dianggap sebagai input yang valid selama proses berjalan sebelum keluar dengan indikator tersebut.

## 3.2 Saran Pengembangan

Berdasarkan hasil analisis dari kelima kode program perulangan *while* dan *do while* yang telah dijalankan, terdapat beberapa saran pengembangan agar program menjadi lebih interaktif, efisien, dan bermanfaat dalam penerapan nyata.

Pada program *PerulanganWhile1* dan *SentinelLoop*, dapat ditambahkan validasi input agar pengguna tidak memasukkan nilai yang tidak sesuai, serta menampilkan ringkasan hasil yang lebih informatif. Pada program *DoWhile1*, sistem *verifikasi password* bisa dikembangkan dengan membatasi jumlah percobaan dan menambahkan sistem keamanan sederhana seperti *masking* input.

Untuk program LemparDadu, dapat diperluas menjadi simulasi permainan sederhana dengan beberapa pemain atau tampilan skor setiap ronde. Sedangkan GamePenjumlahan bisa dikembangkan menjadi permainan edukatif dengan level kesulitan bertahap dan sistem skor otomatis.

Dengan pengembangan tersebut, program tidak hanya menjadi latihan logika dasar pemrograman, tetapi juga dapat digunakan sebagai media pembelajaran interaktif yang lebih menarik dan bermanfaat.

## DAFTAR PUSTAKA

- [1] H. Schildt, *Java: The Complete Reference*, 12th ed. New York: McGraw-Hill Education, 2021.
- [2] J. Gosling, B. Joy, G. Steele, and G. Bracha, *The Java Language Specification*, 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2014.
- [3] D. Liang, *Introduction to Java Programming and Data Structures*, 12th ed. Boston: Pearson Education, 2020.
- [4] Oracle, “*The Java Tutorials: Control Flow Statements*,” *Oracle Documentation*, 2024. [Online]. Available: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html>
- [5] P. Deitel and H. Deitel, *Java How to Program*, 11th ed. Boston: Pearson Education, 2019.