LAPORAN PRAKTIKUM ALGORITMA DAN PEMROGRAMAN

PERULANGAN WHILE DAN DO-WHILE

PEMROGRAMAN JAVA

disusun Oleh:

Nama: Amiratul Fadhilah

NIM: 2511532023

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

Asisten Praktikum: Jovantri Immanuel Gulo

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

PADANG

2025

KATA PENGANTAR

Puji syukur ke hadirat Allah Yang Maha Esa atas rahmat dan karunia-Nya, sehingga laporan praktikum dengan judul “Perulangan While dan Do-While” ini dapat diselesaikan dengan baik dan tepat waktu.

Laporan ini disusun sebagai salah satu syarat untuk memenuhi tugas mata kuliah Praktikum Algoritma dan Pemrograman. Tujuan utama dari penyusunan laporan ini adalah untuk memperdalam pemahaman mengenai konsep dasar kontrol alur program, khususnya implementasi struktur perulangan while dan do-while dalam bahasa pemrograman Java. Laporan ini akan menguraikan siktaksis dasar, perbedaan mendasar, dan penerapan kedua struktur perulangan tersebut dalam menangani proses repetitif yang jumlah pengulangannya tidak pasti.

Penulis menyadari bahwa laporan ini masih memiliki banyak kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk perbaikan di masa mendatang. Semoga laporan ini dapat memberikan manfaat serta pemahaman yang lebih mendalam bagi pembaca.

Padang, 06 November 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam aktivitas pemrograman komputer, kita sering kali menemukan kebutuhan untuk menjalankan satu atau serangkaian perintah yang sama secara berulang kali. Menulis ulang kode secara manual untuk setiap pengulangan merupakan cara yang tidak efisien. Metode manual seperti ini tidak hanya memakan waktu dan tenaga, tetapi juga meningkatkan risiko kesalahan dan membuat program menjadi sulit untuk diperbaiki atau dikembangkan.

Untuk mengatasi masalah ini, bahasa pemrograman modern seperti Java menyediakan mekanisme kontrol alur yang dikenal sebagai struktur perulangan atau *looping*. Struktur ini mengizinkan program untuk menjalankan blok kode yang sama secara iteratif selama sebuah kondisi tertentu terpenuhi.

Meskipun perulangan for ideal untuk situasi di mana jumlah pengulangannya telah diketahui sebelumnya, banyak masalah komputasi di dunia nyata memiliki jumlah perulangan yang tidak pasti. Misalnya, programnya harus terus meminta input pengguna sampai input yang diberikan valid, atau program harus terus berjalan sampai pengguna memutuskan untuk berhenti. Untuk menangani hal ini, diperlukan struktur perulangan tak terhitung, yaitu while dan do-while. Menguasai kedua struktur ini sangat penting agar bisa membuat program yang fleksibel dan mampu merespon masukan dari pengguna.

1.2 Tujuan Praktikum

Pelaksanaan praktikum ini memiliki beberapa tujuan khusus sebagai berikut:

1. Memahami konsep dasar, sintaksis, dan mekanisme kerja struktur kontrol perulangan while dan do-while dalam bahasa Java.
2. Menganalisis perbedaan mendasar antara perulangan while dan do-while.
3. Menerapkan perulangan while untuk menangani pengulangan yang dikontrol oleh flag (kondisi boolean).

4. Menerapkan perulangan while untuk menangani pengulangan yang dikontrol oleh nilai sentinel.
5. Menerapkan perulangan do-while untuk kasus validasi input pengguna, di mana program harus menjalankan perintahnya minimal satu kali, seperti meminta input.

1.3 Manfaat Praktikum

Manfaat yang diharapkan setelah menyelesaikan praktikum ini adalah:

1. Meningkatkan kemampuan mahasiswa dalam merancang program Java yang interaktif dan fleksibel menggunakan perulangan while dan do-while.
2. Meningkatnya kemampuan mahasiswa dalam memilih struktur perulangan yang tepat antara for, while, dan do-while.
3. Memberikan fondasi yang kuat untuk memahami algoritma yang lebih kompleks yang bergantung pada pengulangan yang terhitung, seperti pada data stream atau game *loop*.
4. Meningkatkan keterampilan dalam menyusun laporan teknis yang sistematis sebagai bentuk dokumentasi ilmiah dan bahan pembelajaran.

BAB II

PEMBAHASAN

2.1 Dasar Teori

2.1.1 Struktur Kontrol Perulangan (*Looping*)

Perulangan atau *looping* adalah konsep dasar dalam pemrograman yang berfungsi untuk menjalankan satu atau beberapa perintah secara berulang-ulang.

Perintah tersebut akan terus diulang selama sebuah kondisi khusus masih terpenuhi (atau masih bernilai benar). Di dalam bahasa Java, ada tiga jenis perulangan utama: for, while, dan do-while.

2.1.2 Perulangan Tak Terhitung (*Uncounted Loop*)

Berbeda dengan perulangan terhitung (*counted loop*) seperti for, yang jumlah pengulangannya telah ditetapkan sebelumnya, perulangan tak terhitung (*uncounted loop*) adalah sebuah struktur di mana jumlah total pengulangannya tidak diketahui secara pasti saat program dimulai. Mekanisme perulangan ini bergantung sepenuhnya pada sebuah kondisi acuan, dan akan terus dieksekusi selama kondisi tersebut masih terpenuhi. Dalam bahasa Java, struktur yang termasuk dalam kategori ini adalah while dan do-while.

2.1.3 Perulangan While

Perulangan while adalah sebuah mekanisme pengulangan mendasar yang jumlahnya tidak pasti, atau disebut uncounted loop. Ciri utamanya adalah pengecekan kondisi di awal (pre-test). Ini berarti, program akan terlebih dahulu memeriksa sebuah kondisi boolean (benar/salah). Jika kondisi tersebut bernilai benar (true), barulah program akan menjalankan blok perintah di dalamnya. Selesai menjalankan perintah, program akan kembali lagi untuk memeriksa kondisi tersebut. Proses ini akan terus berputar selama kondisi masih bernilai benar. Namun, jika saat pengecekan kondisi itu bernilai salah (false), blok perintah akan dilewati dan pengulangan akan langsung berhenti. Karena pengecekan dilakukan di awal, jika kondisi sudah bernilai salah sejak semula, blok perintah di dalam while tidak akan pernah dieksekusi sama sekali

2.1.4 Perulangan Do-While

Perulangan do-while adalah sebuah variasi dari while yang dikenal dengan mekanisme post-test atau pengujian di akhir. Ciri khas utamanya adalah do-while menjamin blok perintah di dalamnya akan dijalankan minimal satu kali. Hal ini terjadi karena program akan langsung mengeksekusi blok perintah terlebih dahulu, dan baru setelah itu mengevaluasi kondisi untuk menentukan apakah pengulangan perlu dilanjutkan.

2.1.5 Perulangan Berbasis Sentinel dan Flag

Terdapat dua teknik umum untuk mengontrol perulangan while yang jumlahnya tidak pasti. Teknik pertama adalah berbasis sentinel, yang bergantung pada sebuah nilai khusus sebagai sinyal berhenti. Perulangan akan terus berjalan sampai pengguna memasukkan nilai khusus tersebut; contohnya, program yang terus meminta angka dan baru akan berhenti ketika pengguna memasukkan angka 0, seperti yang ditunjukkan pada SentinelLoop_2511532023.java. Teknik kedua adalah berbasis flag, yang menggunakan sebuah variabel boolean (bernilai true atau false) untuk mengontrolnya. Perulangan akan terus berjalan selama flag tersebut true, dan sebuah kondisi di dalam loop—seperti input dari pengguna—dapat mengubah nilai flag itu menjadi false untuk menghentikan perulangan, sebagaimana dicontohkan dalam perulanganWhile1_2511532023.java.

2.2 Langkah Praktikum

Langkah-langkah yang dilakukan dalam pelaksanaan praktikum ini adalah sebagai berikut:

1. Menyiapkan perangkat yang telah dilengkapi dengan Java Development Kit (JDK) dan sebuah Integrated Development Environment (IDE) seperti Eclipse.
2. Membuka IDE dan membuat sebuah project Java baru.
3. Di dalam project, membuat beberapa class Java baru sesuai dengan nama berkas kode program yang akan diuji (misalnya, perulanganWhile1.java, doWhile1.java, dst.).

4. Menuliskan kode program yang telah disediakan.
5. Melakukan kompilasi (*compile*) dan eksekusi (*run*) pada setiap class program.
6. Mengamati dan mencatat keluaran (*output*) yang dihasilkan oleh setiap program.
7. Menganalisis alur eksekusi kode, membandingkan output yang dihasilkan dengan teori, dan memahami bagaimana setiap baris kode berkontribusi pada hasil akhir.
8. Menyusun laporan praktikum sesuai dengan format yang telah ditetapkan.

2.3 Kode Program

2.3.1 Program Perulangan While (perulanganWhile1_2511532023)

```java
package pekan6_2511532023;

import java.util.Scanner;

public class perulanganWhile1_2511532023 {
    public static void main(String[] args) {

        int counter = 0;
        String jawab;
        boolean running = true;
        // deklarasi scanner
        Scanner scan = new Scanner(System.in);
                while (running) {
                    counter++;
                    System.out.println("Jumlah = " +counter);
                    System.out.print("Apakah lanjut (ya / tidak?)" +" ");
                    jawab = scan.nextLine();
                    // cek jawab = tidak, perulangan berhenti
                    if (jawab.equalsIgnoreCase("tidak")) {
                        running = false;
                    }
                }
                System.out.println("Anda sudah melakukan perulangan sebanyak " +counter+" kali");
    }

}
```

Kode Program 2.1

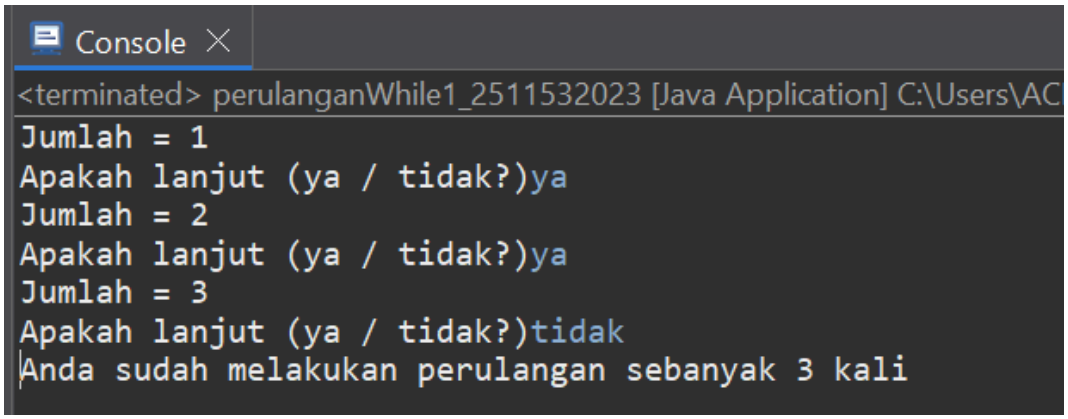
Struktur dan Langkah Program:

1. package pekan6_2511532023;: Mendeklarasikan bahwa *class* ini berada dalam *package* (folder) bernama pekan6_2511532023.

2. import java.util.Scanner;: Mengimpor *class* Scanner dari *library* java.util. *Class* ini diperlukan untuk dapat membaca input dari pengguna melalui keyboard.
3. public class perulanganWhile1_2511532023 {...}: Mendefinisikan sebuah *class* publik dengan nama perulanganWhile1_2511532023.
4. public static void main(String[] args) {...}: Ini adalah *method* utama (*main method*). public static void adalah kata kunci yang menetapkan *method* ini sebagai titik awal (entry point) eksekusi program Java.
5. int counter = 0;: Mendeklarasikan variabel counter dengan tipe data int (integer) dan memberinya nilai awal 0.
6. String jawab;: Mendeklarasikan variabel jawab dengan tipe data String, yang akan digunakan untuk menyimpan input dari pengguna.
7. boolean running = true;: Mendeklarasikan variabel running dengan tipe data boolean dan memberinya nilai awal true. Variabel ini akan digunakan sebagai *flag* (penanda) untuk mengontrol perulangan.
8. Scanner scan = new Scanner(System.in);: Membuat sebuah objek baru dari *class* Scanner yang diberi nama scan. Objek ini akan membaca dari System.in (input standar, yaitu keyboard).
9. while (running) {...}: Memulai struktur perulangan while. Program akan mengevaluasi kondisi di dalam tanda kurung. Selama variabel running bernilai true, blok kode di dalam {...} akan terus dieksekusi.
10. counter++;: (Di dalam *while*) Menambahkan nilai counter sebanyak 1 (satu) di setiap pengulangan.
11. System.out.println("Jumlah = " + counter);: Mencetak nilai counter saat ini ke konsol, diikuti dengan pindah baris.
12. System.out.print("Apakah lanjut (ya / tidak?)" + " ");: Mencetak pertanyaan ke konsol tanpa pindah baris, menunggu input pengguna di baris yang sama.
13. jawab = scan.nextLine();: Menunggu dan membaca input pengguna (seluruh baris teks) dan menyimpannya ke dalam variabel jawab.

14. if (jawab.equalsIgnoreCase("tidak")) {...}: Mengecek sebuah kondisi. Jika nilai variabel jawab (tanpa memedulikan huruf besar/kecil) adalah "tidak", maka blok kode if akan dieksekusi.
15. running = false;: (Di dalam if) Mengubah nilai *flag* running dari true menjadi false. Ini adalah aksi krusial yang akan menghentikan perulangan.
16. System.out.println("Anda sudah...");: Baris ini berada di luar blok *loop* while. Baris ini hanya akan dieksekusi setelah perulangan berhenti (yaitu, ketika running menjadi false), dan akan mencetak total counter.

Output:



Gambar 2.1

Program ini mendemonstrasikan perulangan while yang dikontrol oleh *flag* boolean (*running*). Perulangan akan terus berjalan, meminta input pengguna, dan menambah counter selama variabel *running* bernilai *true*. Perulangan hanya akan berhenti ketika pengguna memasukkan "tidak" (dalam huruf besar atau kecil), yang akan mengubah nilai *flag running* menjadi *false*.

2.3.2 Program Perulangan Do-While (doWhile_2511532023)

```java
package pekan6_2511532023;

import java.util.Scanner;

public class doWhile1_2511532023 {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        String phrase;
        do {
            System.out.print("Input Password: ");
            phrase = console.next();
        } while (!phrase.equals("abcd"));
    }

}
```

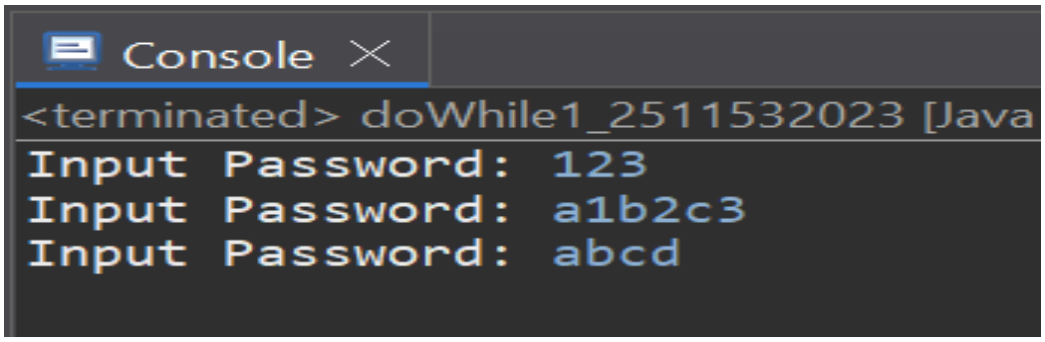
Kode Program 2.2

Struktur dan Langkah Program:

1. package pekan6_2511532023;: Mendeklarasikan *class* sebagai bagian dari *package* pekan6_2511532023.
2. import java.util.Scanner;: Mengimpor *class* Scanner untuk membaca input.
3. public class doWhile1_2511532023 {...}: Mendefinisikan *class* publik.
4. public static void main(String[] args) {...}: *Method* utama, titik awal eksekusi.
5. Scanner console = new Scanner(System.in);: Membuat objek Scanner baru bernama console.
6. String phrase;: Mendeklarasikan variabel phrase (tipe String) untuk menyimpan input *password*.
7. do {...}: Memulai blok do dari struktur do-while. Kode di dalam blok ini dijamin akan dieksekusi setidaknya satu kali.
8. System.out.print("Input Password: ");: (Di dalam do) Mencetak *prompt* ke pengguna untuk memasukkan *password*.
9. phrase = console.next();: (Di dalam do) Membaca input pengguna (hanya kata pertama, next()) dan menyimpannya ke phrase.

10. while (!phrase.equals("abcd"));: Ini adalah bagian kondisi dari do-while. Kondisi ini dievaluasi setelah blok do dijalankan.
    a. !phrase.equals("abcd"): Kondisi ini berarti "selama phrase TIDAK (!) sama dengan 'abcd'".
    b. Jika true (misal pengguna input "123"), program akan melompat kembali ke do dan meminta *password* lagi.
    c. Jika false (pengguna input "abcd"), perulangan berhenti dan program selesai.

Output:



Gambar 2.2

Program ini menunjukkan keunggulan do-while untuk validasi input. Program harus meminta password terlebih dahulu (aksi do) sebelum bisa mengecek apakah password itu benar (kondisi while). Blok do akan terus diulang selama pengguna belum memasukkan input yang benar ("abcd").

2.3.3 Program Perulangan While (SentinelLoop_2511532023)

```java
package pekan6_2511532023;

import java.util.Scanner;

public class SentinelLoop_2511532023 {
    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);
    }

}
```

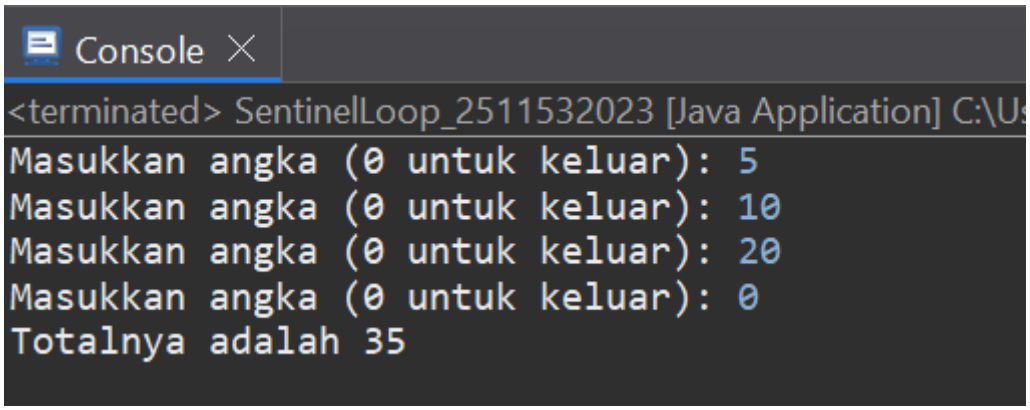
Kode Program 2.3

Struktur dan Langkah Program:

1. package pekan6_2511532023;: Mendeklarasikan *package*.
2. import java.util.Scanner;: Mengimpor *class* Scanner.
3. public class SentinelLoop_2511532023 {...}: Mendefinisikan *class* publik.
4. public static void main(String[] args) {...}: *Method* utama.
5. Scanner console = new Scanner(System.in);: Membuat objek Scanner.
6. int sum = 0;: Mendeklarasikan dan menginisialisasi variabel sum ke 0, yang akan digunakan untuk akumulasi (penjumlahan).
7. int number = 12;: Mendeklarasikan variabel number. Variabel ini diberi "nilai dummy" (12, atau nilai apa pun selain 0) agar kondisi while pada pengecekan pertama bernilai true.
8. while (number != 0) {...}: Memulai perulangan while. Kondisi ini berarti "selama number tidak sama dengan 0". Nilai 0 di sini bertindak sebagai *sentinel value* (nilai penanda berhenti).
9. System.out.print("Masukkan angka (0 untuk keluar): ");: (Di dalam *while*) Menampilkan *prompt* ke pengguna.

10. number = console.nextInt();: Membaca input int dari pengguna dan memperbarui nilai number.
11. sum = sum + number;: Menambahkan nilai number yang baru diinput ke sum.
12. System.out.println("Totalnya adalah " + sum);: Dieksekusi setelah perulangan while berhenti (yaitu, ketika pengguna memasukkan 0).

Output:



Gambar 2.3

Program ini menggunakan nilai 0 sebagai sentinel untuk menghentikan perulangan. Program terus meminta angka dan menjumlahkannya (termasuk nilai 0 terakhir saat loop berhenti) sampai pengguna memasukkan 0. Pengecekan while (number != 0) di awal memastikan perulangan berhenti setelah nilai sentinel dimasukkan dan dievaluasi.

2.3.4 Program Perulangan While (Lempardadu_2511532023)

```java
package pekan6_2511532023;

import java.util.Random;

public class Lempardadu_2511532023 {
    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!");
    }

}
```

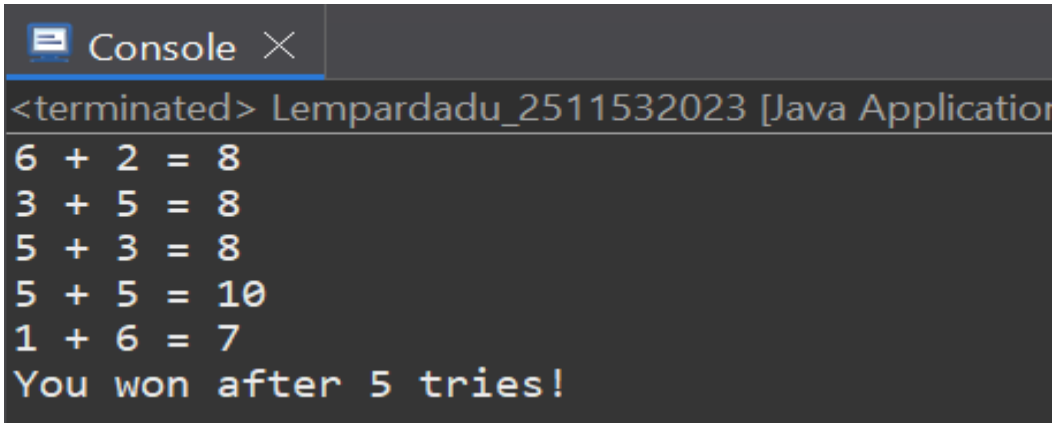
Kode Program 2.4

Struktur dan Langkah Program:

1. package pekan6_2511532023;: Mendeklarasikan *package*.
2. import java.util.Random;: Mengimpor *class* Random dari java.util, yang diperlukan untuk menghasilkan angka acak.
3. public class Lempardadu_2511532023 {...}: Mendefinisikan *class* publik.
4. public static void main(String[] args) {...}: *Method* utama.
5. Random rand = new Random();: Membuat objek baru dari *class* Random bernama rand.
6. int tries = 0;: Mendeklarasikan dan menginisialisasi tries (jumlah percobaan) ke 0.
7. int sum = 0;: Mendeklarasikan dan menginisialisasi sum (jumlah total dadu) ke 0. Nilai awal ini (bukan 7) penting agar perulangan while bisa dimulai.
8. while (sum != 7) {...}: Memulai perulangan while. Kondisi ini berarti "selama sum tidak sama dengan 7".

9. int dadu1 = rand.nextInt(6) + 1;: (Di dalam *while*) Menghasilkan angka acak. rand.nextInt(6) menghasilkan angka 0-5, kemudian + 1 mengubah rentangnya menjadi 1-6.
10. int dadu2 = rand.nextInt(6) + 1;: Menghasilkan angka acak 1-6 untuk dadu kedua.
11. sum = dadu1 + dadu2;: Menjumlahkan kedua dadu dan memperbarui nilai sum. Nilai sum baru ini akan dicek di pengulangan berikutnya.
12. System.out.println(dadu1 + " + " + dadu2 + " = " + sum);: Mencetak hasil lemparan dadu saat ini.
13. tries++;: Menambah (increment) jumlah percobaan.
14. System.out.println("You won after " + tries + " tries!");: Dieksekusi setelah perulangan while berhenti (yaitu, ketika sum akhirnya bernilai 7).

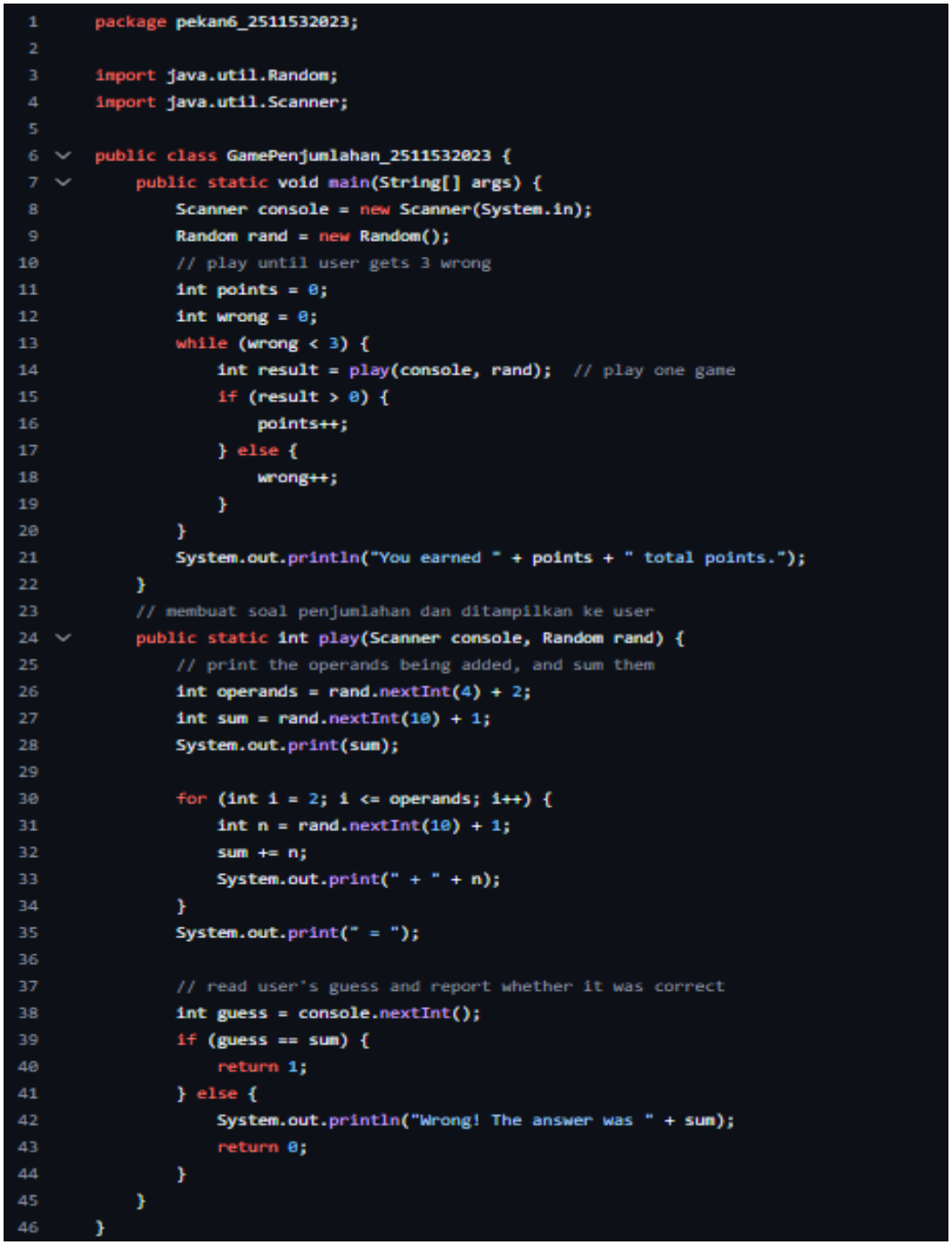
Output:



Gambar 2.4

Program ini adalah contoh sempurna dari *uncounted loop* di mana jumlah pengulangan tidak dapat diprediksi. Program akan terus "melempar dadu" (menghasilkan sum acak) dan berhenti hanya ketika sum yang dihasilkan adalah 7.

2.3.5 Program Perulangan While (GamePenjumlahan_2511532023)



```java
package pekan6_2511532023;

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

public class GamePenjumlahan_2511532023 {
    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.");
    }
    // membuat 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 whether it was correct
        int guess = console.nextInt();
        if (guess == sum) {
            return 1;
        } else {
            System.out.println("Wrong! The answer was " + sum);
            return 0;
        }
    }
}
```

Kode Program 2.5

Struktur dan Langkah Program:

1. package pekan6_2511532023;: Mendeklarasikan bahwa *class* ini berada dalam *package* (folder) pekan6_2511532023.

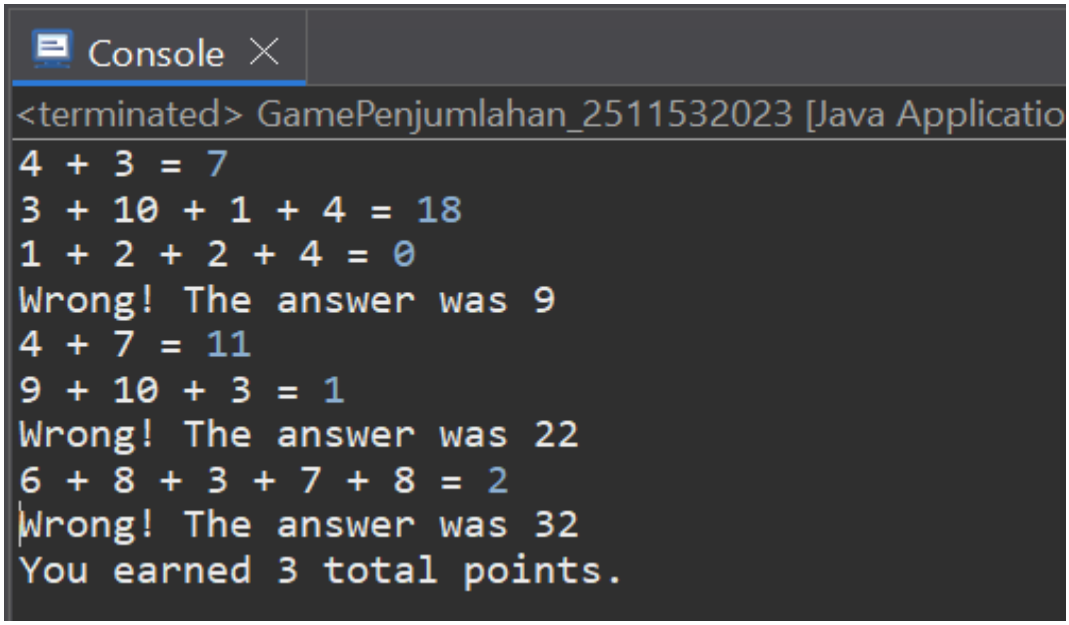
2. import java.util.Random;: Mengimpor *class* Random dari *library* java.util untuk fungsionalitas penghasil angka acak.
3. import java.util.Scanner;: Mengimpor *class* Scanner dari *library* java.util untuk membaca input pengguna.
4. public class GamePenjumlahan_2511532023 {...}: Mendefinisikan sebuah *class* publik.
5. public static void main(String[] args) {...}: *Method* utama, titik awal eksekusi program.
6. Scanner console = new Scanner(System.in);: Membuat objek Scanner baru bernama console untuk membaca input keyboard.
7. Random rand = new Random();: Membuat objek Random baru bernama rand untuk menghasilkan angka acak.
8. int points = 0;: Mendeklarasikan variabel points (skor) dan menginisialisasinya ke 0.
9. int wrong = 0;: Mendeklarasikan variabel wrong (jumlah kesalahan) dan menginisialisasinya ke 0. Variabel ini akan bertindak sebagai pengontrol perulangan while.
10. while (wrong < 3) {...}: Memulai struktur perulangan while. Ini adalah *uncounted loop* yang akan terus berjalan selama nilai variabel wrong masih kurang dari 3.
11. int result = play(console, rand);: (Di dalam *while*) Memanggil *method* play (yang didefinisikan di bawah) dan meneruskan objek console dan rand sebagai argumen. Nilai yang dikembalikan (1 jika benar, 0 jika salah) disimpan dalam variabel result.
12. if (result > 0) {...}: Mengecek nilai result. Jika result lebih dari 0 (yaitu, bernilai 1), berarti pengguna menjawab dengan benar.
13. points++;: (Di dalam if) Menambahkan 1 ke total skor points.
14. else {...}: Jika kondisi if tidak terpenuhi (yaitu, result adalah 0), berarti pengguna menjawab salah.
15. wrong++;: (Di dalam else) Menambahkan 1 ke total kesalahan wrong. Ini adalah baris krusial yang memperbarui kondisi while. Setelah 3 kali salah,

wrong akan bernilai 3, dan kondisi while (wrong < 3) akan menjadi false, sehingga *loop* berhenti.

16. System.out.println("You earned " + points + " total points.");: Dieksekusi setelah *loop* while selesai. Mencetak total skor yang diperoleh.
17. public static int play(Scanner console, Random rand) {...}: Mendefinisikan *method* statis baru bernama play yang menerima Scanner dan Random sebagai parameter dan mengembalikan (return) sebuah nilai int.
18. int operands = rand.nextInt(4) + 2;: (Di dalam play) Menghasilkan angka acak antara 2 hingga 5. rand.nextInt(4) menghasilkan (0-3), ditambah 2 menjadi (2-5). Ini menentukan jumlah bilangan yang akan dijumlahkan.
19. int sum = rand.nextInt(10) + 1;: Menghasilkan bilangan acak pertama (1-10) dan menyimpannya sebagai nilai awal sum.
20. System.out.print(sum);: Mencetak bilangan pertama ke konsol.
21. for (int i = 2; i <= operands; i++) {...}: (Di dalam play) Memulai perulangan for untuk menangani bilangan kedua dan seterusnya, sampai jumlah operands tercapai.
22. int n = rand.nextInt(10) + 1;: (Di dalam for) Menghasilkan bilangan acak baru (1-10) untuk ditambahkan.
23. sum += n;: Menambahkan bilangan baru n ke total sum. (sum += n adalah singkatan dari sum = sum + n).
24. System.out.print(" + " + n);: Mencetak " + " diikuti dengan bilangan baru n.
25. System.out.print(" = ");: Dieksekusi setelah *loop* for selesai, untuk mencetak tanda " = ".
26. int guess = console.nextInt();: Membaca jawaban (tebakan) integer dari pengguna.
27. if (guess == sum) {...}: Mengecek apakah tebakan pengguna sama dengan total sum yang benar.
28. return 1;: (Di dalam if) Jika tebakan benar, *method* play berhenti dan mengembalikan nilai 1 ke *method* main.
29. else {...}: Jika tebakan salah.

30. System.out.println("Wrong! The answer was " + sum);: Memberi tahu pengguna jawaban yang benar.
31. return 0;: (Di dalam else) *Method* play berhenti dan mengembalikan nilai 0 ke *method* main.

Output:



Gambar 2.5

Program ini adalah sebuah permainan kuis penjumlahan yang menggabungkan dua jenis perulangan. Perulangan while di *method main* bertindak sebagai *game loop* utama, yang mengontrol sesi permainan. Perulangan ini akan terus berjalan (*uncounted loop*) hingga kondisi "salah kurang dari 3" tidak lagi terpenuhi. Di dalam *game loop* tersebut, perulangan for di dalam *method* play digunakan sebagai *counted loop* untuk membangun soal penjumlahan dengan jumlah bilangan (*operands*) yang acak (antara 2 sampai 5 bilangan).

BAB III

PENUTUP

3.1 Kesimpulan

Berdasarkan analisis praktikum, dapat disimpulkan bahwa struktur while dan do-while merupakan mekanisme kontrol yang esensial dalam Java untuk menangani situasi pengulangan yang jumlahnya belum ditentukan sebelumnya. Perbedaan mendasar di antara keduanya terletak pada waktu pengecekan kondisi. Struktur while melakukan pengecekan kondisi di awal (*pre-test*), yang berarti blok kode hanya akan dieksekusi jika kondisi sudah terpenuhi; jika tidak, blok kode tersebut bisa saja tidak dieksekusi sama sekali. Sebaliknya, do-while melakukan pengecekan di akhir (*post-test*), sebuah mekanisme yang menjamin blok kode di dalamnya dieksekusi setidaknya satu kali. Secara praktis, while sangat efektif untuk pengulangan yang dikontrol oleh sebuah penanda kondisi (*flag*) atau nilai penentu berhenti (*sentinel value*). Sementara itu, do-while terbukti ideal untuk skenario seperti validasi input, di mana sebuah aksi (seperti meminta masukan) harus dilakukan terlebih dahulu sebelum kondisinya dapat diperiksa.

3.2 Saran

Untuk pengembangan pemahaman lebih lanjut, disarankan bagi mahasiswa agar mendalami perbedaan fungsional antara while dan do-while sehingga dapat memilih struktur mana yang paling efisien untuk suatu masalah. Selain itu, sangat krusial untuk selalu memastikan bahwa di dalam blok perulangan terdapat sebuah aksi yang pada akhirnya akan memperbarui variabel dan mengubah kondisi berhenti. Kelalaian dalam hal ini, seperti lupa menambahkan *increment* (counter++) atau tidak mengambil input baru dari pengguna, akan mengakibatkan terjadinya *infinite loop* atau perulangan tak terhingga. Kondisi ini berisiko membuat program berhenti bekerja (*crash*) atau tidak merespons, oleh karena itu, praktikan disarankan untuk selalu memeriksa logika pembaruan kondisi ini dengan seksama.

DAFTAR PUSTAKA

[1] B. Hartono, *Pemrograman Java untuk Pemula*. Semarang: Yayasan Prima Agus Teknik, 2022. [Daring]. Tersedia pada: https://penerbit.stekom.ac.id/index.php/yayasanpat/article/view/351. [Diakses: 05-Okt-2025].

[2] T. Suryana, "Bab 2 Perulangan dalam JavaScript," Universitas Komputer Indonesia, 2019. [Daring]. Tersedia pada: http://kuliahonline.unikom.ac.id/?listmateri/&detail=46017. [Diakses: 06-Okt-2025].

[3] Departemen Informatika Universitas Andalas, *Pedoman Penulisan Laporan Praktikum Java*. Padang: FTI Unand, 2025.

[4] Dunia Ilkom, "Tutorial Belajar Java: Perulangan Do-While Bahasa Java." [Daring]. Tersedia pada: https://www.duniailkom.com/tutorial-belajar-java-perulangan-do-while-bahasa-java/. [Diakses: 06-Okt-2025].

[5] Minarsih, "Belajar Bahasa Pemrogaman Java (35) Perulangan Do-While Bahasa Java," 2023. [Daring]. Tersedia pada: https://www.minarsih.com/artikel/belajar-bahasa-pemrogaman-java-35-perulangan-do-while-bahasa-java. [Diakses: 05-Okt-2025].

[6] Studocu, "Laporan Praktikum Pemrograman 1 Modul 4." [Daring]. Tersedia pada: https://www.studocu.com/id/document/universitas-tadulako/praktikum-pemograman-1/laporan-praktikum-pemrograman-1-modul-4/46849985. [Diakses: 05-Okt-2025].