LAPORAN PRATIKUM ALGORITMA PEMROGRAMAN

PERULANGAN PADA JAVA

DI SUSUN OLEH :
DIGO YUANDRA
NIM 2511533017
DOSEN PENGAMPU : Dr.WAHYUDI, S.T, M.T
ASISTEN LABORATORIUM: JOVANTRI IMMANUEL GULO

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

PADANG

2025

KATA PENGANTAR

Sebelumnya saya Panjatkan Puji syukur atas kehadiran Tuhan Yang Maha Esa, berkat rahmat dan izinnya juga laporan praktikum “Perulangan Pada Java” ini dapat diselesaikan dengan baik.Saya ucapkan terima kasih yang sebesar besarnya kepada

Dr.Wahyudi, S.T, M.T selaku dosen pengampu yang telah membimbing Mata kuliah Algoritma dan Pemrograman. Dan tidak lupa saya ucapkan terima kasih kepada Uda Aufan Tafiqurrahman selaku asisten labor yang telah membimbing praktikum Perulangan Pada Java

Laporan ini disusun sebagai Bentuk hasil kegiatan praktikum yang bertujuan untuk memahami konsep dasar Perulangan Pada java dalam pemrograman. Penulis menyadari bahwa pemahaman mendalam terhadap kedua topik ini merupakan fondasi essential bagi pengembangan keterampilan pemrograman yang lebih advanced.

Saya menyadari bahwa penulisan laporan praktikum ini masih jauh dari kata sempurna baik dari segi pembahasan dan penulisannya. namun dengan demikian saya telah berupaya dengan segala kemampuan dan pengetahuan yang dimiliki supaya laporan ini dapat selesai dengan baik dan oleh karenanya saya dengan rendah hati menerima saran,masukan dan kritikan guna penyempurnaan laporan ini.

Padang, 30 Oktober2025

Penulis

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Perulangan adalah melakukan perintah yang ada di dalam blok perulangan tersebut secara berulang-ulang sesuai dengan nilai yang ditentukan atau sampai mencapai batas yang diinginkan. Dalam bahasa pemrograman JAVA terdapat beberapa statement perulangan yang dapat digunakan salah satunya yaitu "*while*".

Teknik perulangan *while* adalah kontruksi pemrograman yang mengeksukusi blok kode berulang kali selama kondisi logis yang ditentukan bernilai *true* . ini sangat berguna ketika jumlah literasi tidak diketahui di awal, misalnya saat menunggu input yang valid dari pengguna. Perulangan *while* akan terus berjalan sampai kondisi menjadi false, dan penting untuk memastikan ada mekanisme di dalam blok kode untuk mengahiri perulangan(misalnya dengan mengubah nilai variabel yang digunakan dalam kondisi) untuk mencegah *infinite loop* atau perulangan tak terbatas

1.2 Tujuan

1. Menjelaskan cara mengulang kode eksekusi secara efisien
2. Dapat mengontrol alur eksekusi program dengan lebih baik.
3. Mengaplikasikan perulangan *while* dalam bahasa java

1.3 Manfaat

1. Menambah pemahaman tentang struktur perulangan *while* dalam bahasa Java.
2. Menambah wawasan serta bekal mahasiswa mengenai pemrograman

.

BAB II PEMBAHASAN

2.1 Perulangan Pada Java

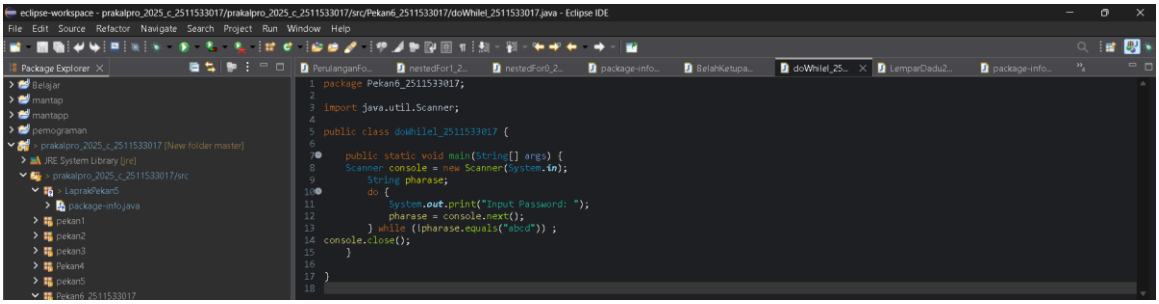
- Perulangan
  Adalah struktur kontrol yang digunakan untuk mengulang eksekusi blok kode dengan banyak kondisi tertentu
- Jenis perulangan

- For : digunakan untuk mengulang eksekusi kode berdasarkan kondisi tertentu
- While : digunakan untuk mengulang eksekusi kode selama kondisi tertentu terpenuhi
- Do-while : digunakan untuk mengulang eksekusi kode minimal sekali sebelum kondisi diperiksa

2.2 Langkah pengerjaan

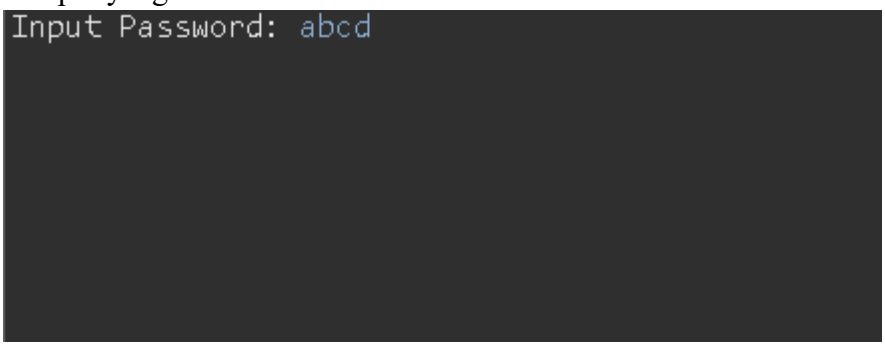
1. Program
   a) Inisialisasi Scanner dan Variabel
      - Objek Scanner dibuat untuk membaca input dari user
      - Variabel phrase dideklarasikan untuk menyimpan input password
   b) Eksekusi Awal Do-Block
      - Program langsung mengeksekusi badan loop tanpa pemeriksaan kondisi awal
      - Mencetak prompt "Input pasword:" ke konsol
      - Membaca input dari user dan menyimpannya ke variabel phrase
   c) Pemeriksaan Kondisi While
      - Setelah eksekusi do-block, program memeriksa kondisi
      - Program memeriksa apakah nilai phrase tidak sama dengan string "abcd"
      - Jika kondisi benar (password salah), program kembali ke langkah b
      - Jika kondisi salah (password benar "abcd"), program keluar dari loop
   d) Pengulangan Proses
      - Program mengulangi langkah b-c terus-menerus selama kondisi while bernilai true
      - Setiap iterasi meminta input password baru dari user
      - Loop hanya berhenti ketika user memasukkan password "abcd"
   e) Penutupan Scanner
      - Setelah loop berhenti, program menutup objek Scanner
      - Program selesai dieksekusi

Kode program



Gambar 2.1

Output yang dihasilkan



Gambar 2.2

2. Program GamePenjumlahan_2511533017
   a) Inisialisasi Variabel dan Objek
      - Objek Scanner dibuat untuk membaca input dari user
      - Objek Random dibuat untuk menghasilkan angka acak
      - Variabel points diinisialisasi dengan 0 untuk menyimpan jumlah jawaban benar
      - Variabel wrong diinisialisasi dengan 0 untuk menyimpan jumlah jawaban salah
   b) Pemeriksaan Kondisi While Loop Utama
      - Program memeriksa apakah nilai wrong kurang dari 3
      - Jika kondisi benar (kesalahan < 3), program masuk ke badan loop
      - Jika kondisi salah (kesalahan = 3), program keluar dari loop
   c) Pemanggilan Method play() dan Penanganan Hasil
      - · Memanggil method play() yang mengembalikan nilai integer
      - · Jika result > 0 (jawaban benar), increment points
      - · Jika result = 0 (jawaban salah), increment wrong
   d) Update dan Pengulangan Proses
      - · Program kembali ke langkah b (pemeriksaan kondisi) dan mengulangi langkah b-c
      - · Loop berjalan selama jumlah kesalahan kurang dari 3

- · Saat wrong menjadi 3, kondisi wrong < 3 bernilai false, loop berhenti

Method play() - Bagian 1: Persiapan Soal

a) Inisialisasi Operands dan Sum
- operands diberi nilai acak antara 2-5 (jumlah operand)
- sum diberi nilai acak antara 1-10 (angka pertama)
- Mencetak angka pertama

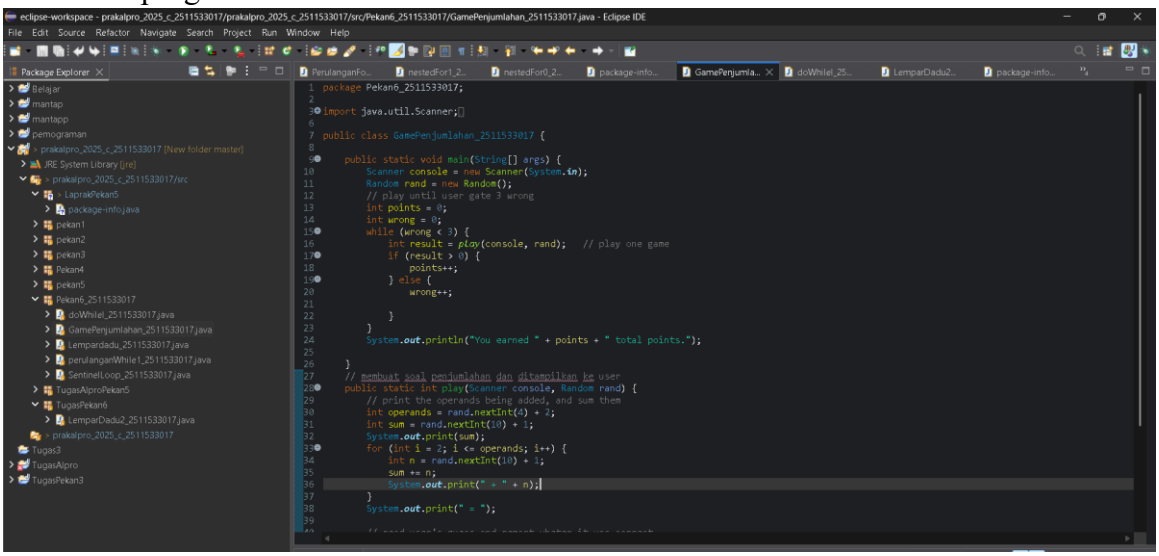
b) Perulangan For untuk Operand Tambahan
- Inisialisasi: int i = 2
- Kondisi: i <= operands
- Increment: i++
- Loop berjalan untuk menambahkan operand kedua dan seterusnya
- Setiap iterasi: menghasilkan angka acak, menambahkannya ke sum, dan mencetak tanda "+" dengan angka

Method play() - Bagian 2: Input dan Evaluasi Jawaban

a) Input dari User -
Mencetak tanda "="
- Membaca tebakan user dari input

b) Perulangan For untuk Operand Tambahan
- · Jika tebakan sama dengan jumlah sebenarnya: cetak pujian dan return 1
- · Jika tebakan salah: cetak jawaban benar dan return 0

Kode program:



```
37          }
38          System.out.print(" = ");
39
40          // read user's guess and report wheter it was correct
41          int guess = console.nextInt();
42          if (guess == sum) {
43              return 1;
44          } else {
45              System.out.println("Wrong! The answer was " + sum);
46              return 0;
47          }
48      }
49  }
50
```

Gambar 2.3

Output yang dihasilkan:

```
8 + 9 + 4 + 7 = 2
Wrong! The answer was 28
7 + 1 + 6 + 10 = 2
Wrong! The answer was 24
2 + 4 = 6
5 + 8 + 2 = 15
10 + 1 + 4 + 8 + 3 = 26
4 + 1 = 5
3 + 10 + 6 + 10 + 3 = 32
2 + 2 = 4
1 + 9 + 4 + 10 + 1 =
```

Gambar 2.4

3. Program Lempardadu_2511533017
   a) Inisialisasi Variabel dan Objek
      - Objek Random dibuat untuk menghasilkan angka acak
      - Variabel tries diinisialisasi dengan 0 untuk menyimpan jumlah percobaan
      - Variabel sum diinisialisasi dengan 0 untuk menyimpan jumlah dadu
   b) Pemeriksaan kondisi while loop
      - Program memeriksa apakah nilai sum tidak sama dengan 7
      - Jika kondisi benar (jumlah ≠ 7), program masuk ke badan loop
      - Jika kondisi salah (jumlah = 7), program keluar dari loop
   c) Badan Loop - Pelemparan Dadu
      - dadu1 diberi nilai acak antara 1-6 (hasil lemparan dadu pertama)
      - dadu2 diberi nilai acak antara 1-6 (hasil lemparan dadu kedua)
      - sum dihitung dari penjumlahan dadu1 + dadu2
   d) Badan Loop - Output dan Increment

- Mencetak hasil lemparan dadu dan jumlah percobaan
- Variabel tries ditambah 1 (menghitung jumlah percobaan)

e) Update dan Pengulangan proses
- Program kembali ke langkah b (pemeriksaan kondisi) dan mengulangi langkah b-d
- Loop berjalan selama jumlah dadu tidak sama dengan 7
- Saat sum menjadi 7, kondisi sum != 7 bernilai false, loop berhenti

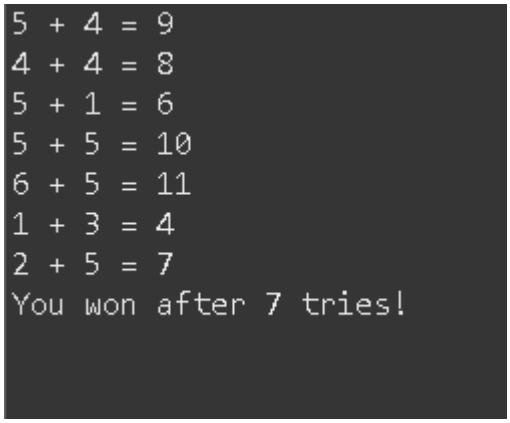
f) Output akhir program
- Setelah loop selesai (ketika jumlah dadu = 7), program mencetak pesan kemenangan dengan jumlah percobaan

Kode program:

```java
package Pekan6_2511533017;

import java.util.Random;

public class Lempardadu_2511533017 {

    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 2.5

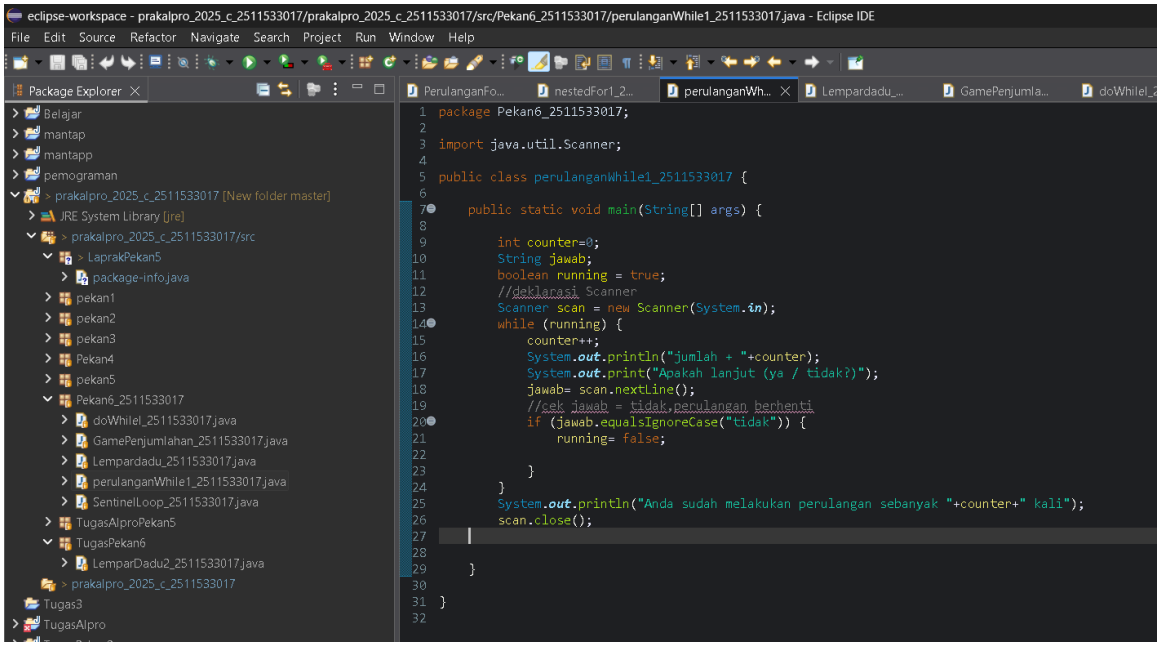
Output yang dihasilkan:



Gambar 2.6

4. Program perulanganWhile1_2511533017:
   a) Inisialisasi Variabel dan Objek
      - · Variabel counter diinisialisasi dengan 0 untuk menyimpan jumlah perulangan
      - · Variabel jawab dideklarasikan untuk menyimpan input user
      - · Variabel running diinisialisasi dengan true sebagai kondisi perulangan
      - · Objek Scanner dibuat untuk membaca input dari user
   b) Pemeriksaan Kondisi While Loop
      - · Program memeriksa apakah nilai running sama dengan true
      - · Jika kondisi benar (running = true), program masuk ke badan loop
      - · Jika kondisi salah (running = false), program keluar dari loop
   c) Badan Loop - Increment Counter dan Output
      - Variabel counter ditambah 1 (menghitung jumlah perulangan)
      - Mencetak jumlah perulangan saat ini
   d) Badan Loop - Input dari User
      - Mencetak prompt untuk konfirmasi kelanjutan
      - Membaca input user dan menyimpannya ke variabel jawab
   e) Badan Loop - Pengecekan Kondisi Berhenti
      - Program memeriksa apakah input user sama dengan "tidak" (tidak case sensitive)
      - Jika kondisi benar, variabel running diubah menjadi false
      - Jika kondisi salah, variabel running tetap true
   f) Update dan Pengulangan Proses
      - Program kembali ke langkah 2 (pemeriksaan kondisi) dan mengulangi langkah 2–4 terus-menerus.
      - Karena nilai awal i = 1 dan setiap iterasi i bertambah 1, maka loop akan berjalan untuk nilai i = 1, 2, 3, ..., 10 (total 10 iterasi). Saat i menjadi 11, kondisi i <= 10 bernilai false, sehingga loop berhenti dan program selesai
   g) Output Akhir Program
      - · Setelah loop selesai, program mencetak total perulangan yang dilakukan

- · Menutup objek Scanner



Kode program:

Gambar 2.7

Output yang dihasilkan:

```
jumlah + 1
Apakah lanjut (ya / tidak?)ya
jumlah + 2
Apakah lanjut (ya / tidak?)ya
jumlah + 3
Apakah lanjut (ya / tidak?)tidak
Anda sudah melakukan perulangan sebanyak 3 kali
```

Gambar 2.8

5. Program SentinelLoop_2511533017
   a) Inisialisasi Variabel dan Objek
      - Objek Scanner dibuat untuk membaca input dari user
      - Variabel sum diinisialisasi dengan 0 untuk menyimpan total penjumlahan
      - Variabel number diinisialisasi dengan 12 sebagai nilai awal
   b) Pemeriksaan Kondisi While Loop
      - Program memeriksa apakah nilai number tidak sama dengan 0 - Jika kondisi benar (number ≠ 0), program masuk ke badan loop
      - Jika kondisi salah (number = 0), program keluar dari loop
   c) Badan Loop - Input dari User
      - Mencetak prompt untuk memasukkan angka
      - Membaca input angka dari user dan menyimpannya ke variabel number
   d) Badan Loop - Akumulasi Penjumlahan
      - · Menambahkan nilai number ke variabel sum
      - · Melakukan akumulasi penjumlahan semua angka yang dimasukkan user
   e) Update dan Pengulangan proses
      - Program kembali ke langkah b (pemeriksaan kondisi) dan mengulangi langkah b-d
      -
      - Loop berjalan selama nilai number tidak sama dengan 0
      - Saat user memasukkan angka 0, kondisi number != 0 bernilai false dan loop berhenti
   f) Output Akhir Program
      - Setelah loop selesai, program mencetak total penjumlahan semua angka
      - Menutup objek Scanner

   Kode program:

```java
package Pekan6_2511533017;

import java.util.Scanner;

public class SentinelLoop_2511533017 {

    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("Masukan angka (0 untuk keluar): ");
            number = console.nextInt();
            sum = sum + number;
        }
        System.out.println("totalnya adalah " + sum);
        console.close();
    }


}
```

Gambar 2.9

Hasil output

```
Console  Git Staging  Install Java 25 Su
<terminated> SentinelLoop_2511533017 [Java Applica
Masukan angka (0 untuk keluar): 22
Masukan angka (0 untuk keluar): 33
Masukan angka (0 untuk keluar): 44
Masukan angka (0 untuk keluar): 55
Masukan angka (0 untuk keluar): 0
totalnya adalah 154
```

BAB III KESIMPULAN

3.1 Kesimpulan

Berdasarkan serangkaian praktikum perulangan dalam Java, dapat disimpulkan bahwa konsep perulangan merupakan fondasi penting dalam pemrograman untuk menangani tugas-tugas yang berulang. Praktikum ini berhasil menunjukkan berbagai jenis perulangan seperti for, while, dan do-while, masing-masing dengan karakteristik dan penggunaan yang spesifik. Perulangan for cocok untuk situasi dimana jumlah iterasi sudah diketahui sebelumnya, sementara while dan do-while lebih fleksibel untuk kondisi yang tidak pasti. Yang menarik, praktikum juga mengimplementasikan nested loop yang memungkinkan pembuatan pola matriks dan tabel penjumlahan, menunjukkan bagaimana perulangan dapat dikombinasikan untuk menyelesaikan masalah yang lebih kompleks.

Praktikum ini tidak hanya mengajarkan sintaks perulangan, tetapi juga konsep penting seperti sentinel value yang digunakan sebagai penanda berhenti, accumulator variable untuk menyimpan total nilai, dan flag variable untuk mengontrol eksekusi program. Implementasi dalam game penjumlahan dan simulasi lempar dadu membuktikan bahwa perulangan sangat aplikatif dalam pengembangan program interaktif. Melalui berbagai contoh ini, menjadi jelas bahwa pemahaman mendalam tentang perulangan memungkinkan programmer untuk menulis kode yang lebih efisien, mudah dibaca, dan mampu menangani berbagai skenario pemrograman yang melibatkan proses berulang.

3.2 Saran

1. Akan lebih baik bila dosen menyelenggarakan sesi pra-praktikum di kelas agar mahasiswa dapat memperoleh pemahaman awal yang lebih memadai, sehingga dapat menghindari kepanikan atau kesalahan saat praktikum

2. Sebaiknya dosen membagikan materi praktikum terlebih dahulu melalui iLearn agar mahasiswa bisa mempersiapkan diri sebelum pelaksanaan praktikum

DAFTAR PUSTAKA

[1] “The while and do-while Statements (The Java™ Tutorials > Learning the Java Language)”, Oracle, 2024. Available: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html