LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN

PERULANGAN WHILE DAN DO-WHILE

TUGAS ALPRO PEKAN 6

Disusun Oleh :

KINAYA NOVRYA MANDA

(2511531016)

Kelas B Informatika

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, November 2025

Penulis

DAFTAR ISI

DAFTAR LAMPIRAN

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

```java
package Pekan6_2511531016;
import java.util.Random;
import java.util.Scanner;
public class tugasAlproPekan6_2511531016 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Random rand = new Random();
        String jawab;
        int percobaan = 0;
        boolean menang = false;

        do {
            int dadu1 = rand.nextInt(6) + 1; // angka dari 1 sampai 6
            int dadu2 = rand.nextInt(6) + 1;
            int jumlah = dadu1 + dadu2;

            percobaan++;

            System.out.println(dadu1 + " + " + dadu2 + " = " + jumlah);

            if (jumlah == 7) {
                System.out.println("Tebakan Anda Benar");
                System.out.println("Anda menang setelah " + percobaan + " percobaan!");
                menang = true;
                break;
            } else {
                System.out.println("Tebakan Anda Salah");
                System.out.print("Apakah mau lempar dadu (ya / tidak?) ");
                jawab = input.next();


                if (jawab.equalsIgnoreCase("tidak")) {
                    System.out.println("Anda gagal menang");
                    break;
                }
            }
        } while (true);

        input.close();
    }
}
```

Kode Program 2.1 (Kode Program tugasAlproPekan6)

2.1.1 Uraian Kode Program

1. *package Pekan6_2511531016;*

   → Menunjukkan program ini berada dalam paket bernama *Pekan6_2511531016*.

2. *import java.util.Random;* dan *import java.util.Scanner;*

   → Mengimpor *library Random* untuk angka acak dan *Scanner* untuk input pengguna dari *keyboard*.

3. *public class tugasAlproPekan6_2511531016 {* dan *public static void main(String[] args) {*

   → Mendefinisikan kelas utama beserta fungsi utama yang dijalankan saat program dieksekusi.

4. *Scanner input = new Scanner(System.in);*, *Random rand = new Random();*, *String jawab;*, *int percobaan = 0;*, dan *boolean menang = false;*

   → Inisialisasi dan deklarasi objek *Scanner* untuk input, objek *Random* untuk acak angka, serta *variable* lokal untuk status permainan, input *user*, dan jumlah percobaan.

5. *do { ... } while (true);*

   → Struktur perulangan yang akan berjalan terus hingga pemain menang atau memilih berhenti.

6. *int dadu1 = rand.nextInt(6) + 1;* dan *int dadu2 = rand.nextInt(6) + 1;*

   → Menghasilkan angka acak dari 1 sampai 6 untuk dua buah dadu.

7. *int jumlah = dadu1 + dadu2;*, *percobaan++;*, dan *System.out.println(dadu1 + " + " + dadu2 + " = " + jumlah);*

   → Menjumlahkan hasil kedua dadu, menambah percobaan, dan menampilkan hasil ke layar.

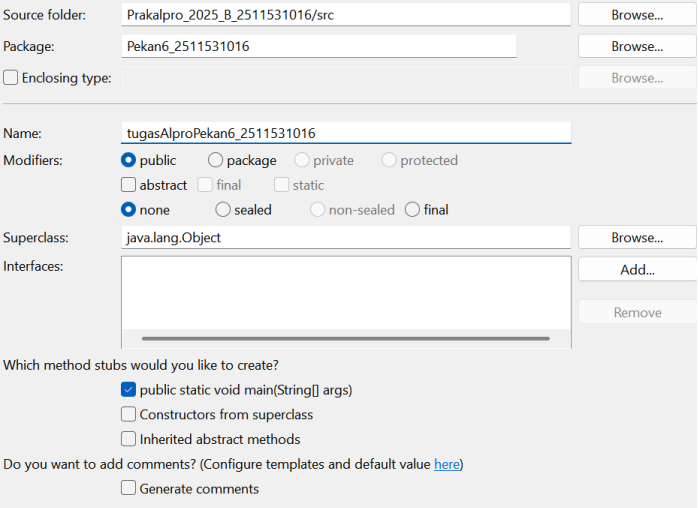
8. *Blok* kondisi
   - *if (jumlah == 7):* Jika hasil kedua dadu adalah 7, program menampilkan pesan kemenangan dan jumlah percobaan, mengubah status menang menjadi *true*, lalu keluar dari perulangan dengan *break*.
   - *else:* Jika bukan 7, menampilkan pesan salah dan meminta input pengguna apakah ingin mencoba lagi.
   - *if (jawab.equalsIgnoreCase("tidak")):* Jika pengguna memilih “tidak”, program menampilkan pesan gagal dan keluar dari perulangan dengan *break*.

9. *input.close();*

→ Menutup *Scanner* setelah program selesai dieksekusi.

2.1.2 Langkah Kerja

1. Membuat kelas baru bernama *tugasAlproPekan6* dalam *package Pekan6_2511531016.*



Gambar 2.1 (Langkah 1)

2. *Import* 2 *library* yaitu *Random* dan *Scanner*. Lalu, definisikan kelas utama *tugasAlproPekan6_2511531016*. Program dijalankan dan masuk ke *main.*

```
package Pekan6_2511531016;
import java.util.Random;
import java.util.Scanner;
public class tugasAlproPekan6_2511531016 {

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

Gambar 2.2 (Langkah 2)

3. Buat input untuk membaca jawaban dari pengguna. Lalu, deklarasikan variabel yang diperlukan.

```
Scanner input = new Scanner(System.in);
Random rand = new Random();
String jawab;
int percobaan = 0;
boolean menang = false;
```

Gambar 2.3 (Langkah 3)

4. Gunakan *do-while* untuk melempar dadu pertama kali, sebelum menanyakan apakah pengguna ingin lanjut atau tidak.

```
do {    } while (true);
```

Gambar 2.4 (Langkah 4)

5. Lempar dua dadu dan jumlahkan kedua nilai. Setiap kali perulangan berjalan, variabel percobaan bertambah 1. Hasilnya akan menampilkan dua dadu ke layar.

```
int dadu1 = rand.nextInt(6) + 1; // angka dari 1 sampai 6
int dadu2 = rand.nextInt(6) + 1;
int jumlah = dadu1 + dadu2;

percobaan++;

System.out.println(dadu1 + " + " + dadu2 + " = " + jumlah);
```

Gambar 2.5 (Langkah 5)

6. Jika jumlah dua dadu sama dengan 7, maka pengguna menang. Program menampilkan pesan kemenangan dan menghentikan perulangan dengan break.

```
if (jumlah == 7) {
    System.out.println("Tebakan Anda Benar");
    System.out.println("Anda menang setelah " + percobaan + " percobaan!");
    menang = true;
```

Gambar 2.6 (Langkah 6)

7. Jika hasil bukan 7, program menampilkan pesan “Tebakan Anda Salah”. Lalu, menanyakan apakah pengguna ingin melempar dadu lagi. Jika menjawab “tidak”, program akan berhenti dan menampilkan pesan “Anda gagal menang”.

```
} else {
    System.out.println("Tebakan Anda Salah");
    System.out.print("Apakah mau lempar dadu (ya / tidak?) ");
    jawab = input.next();

    if (jawab.equalsIgnoreCase("tidak")) {
        System.out.println("Anda gagal menang");
        break;
```

Gambar 2.7 (Langkah 7)

8. Tutup objek *Scanner* untuk menandakan bahwa input dari pengguna sudah selesai.

```
        input.close();
    }
}
```

Gambar 2.8 (Langkah 8)

2.1.3 Analisis Hasil

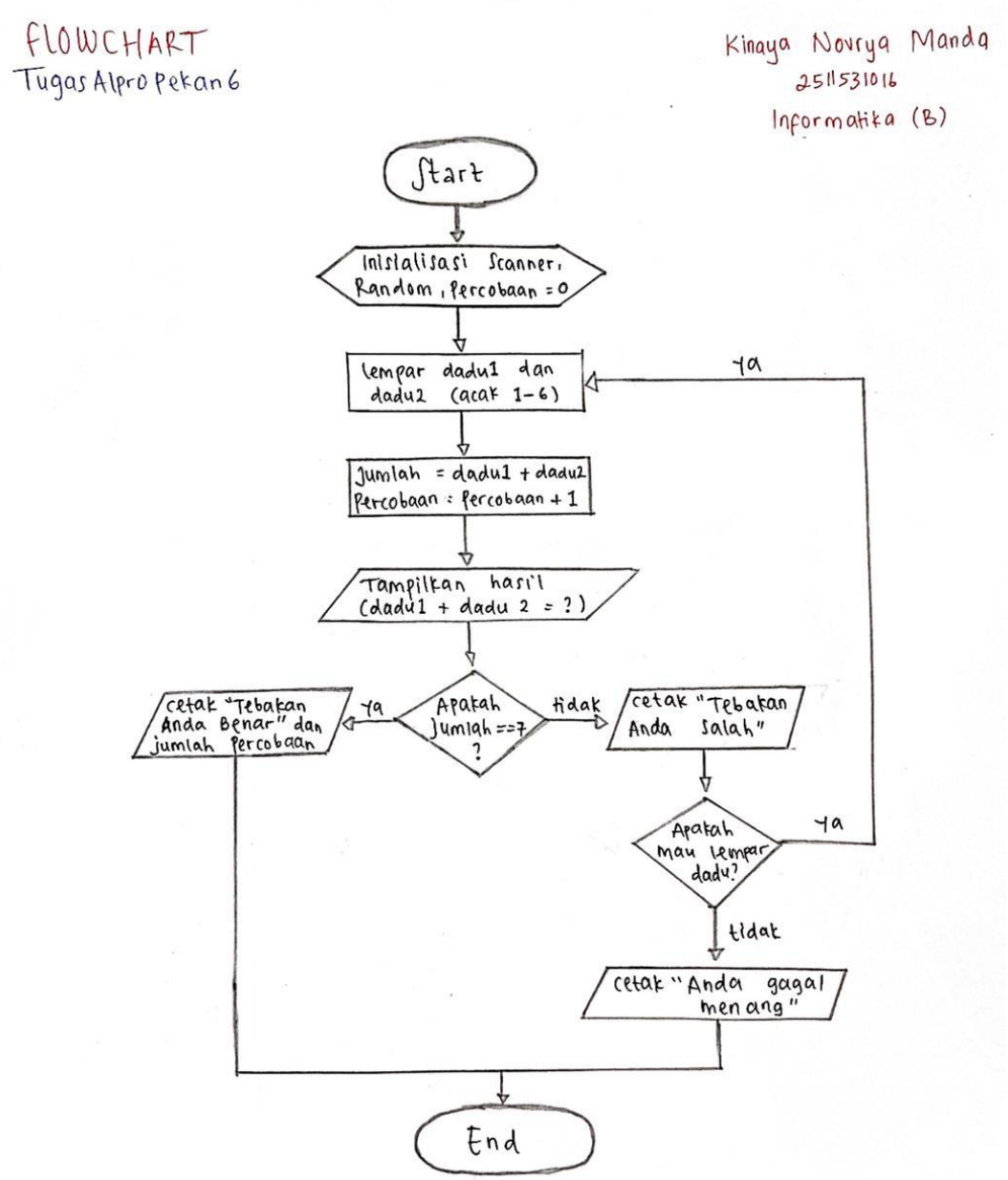
Program *tugasAlproPekan6_2511531016* merupakan simulasi permainan lempar dadu acak berbasis Java, di mana pemain harus menebak hingga hasil penjumlahan dua dadu menghasilkan angka 7. Pada program ini, penggunaan objek *Random* mencerminkan prinsip pembangkitan angka acak, Struktur perulangan *do-while* dipilih agar setidaknya satu kali proses lempar dadu pasti terjadi sebelum ada keputusan untuk berhenti, dan pemain diberi kebebasan untuk melanjutkan atau menghentikan permainan kapan saja.

Setiap kali dadu dilempar, hasilnya dicetak dan dibandingkan dengan angka target. Jika jumlah kedua dadu tepat 7, program menampilkan pesan kemenangan beserta jumlah percobaan yang telah dilakukan. Jika tidak, pengguna diberi pilihan melalui input untuk melanjutkan atau berhenti. Penggunaan variabel untuk menyimpan status permainan, jumlah percobaan, dan input user merupakan praktik standar dalam logika pemrograman terstruktur, memastikan setiap keputusan dan keluaran program dapat dikendalikan dengan jelas.

Berdasarkan teori pemrograman, konsep *random* pada Java mendorong terciptanya aplikasi yang dapat meniru fenomena probabilitas, sedangkan *loop* menjadi kunci dalam menangani proses berulang yang bergantung pada kondisi tertentu atau input pengguna.

Referensi akademik dan praktis menyatakan bahwa kombinasi kelas, objek, kontrol perulangan, pengambilan keputusan dengan *if-else*, dan operasi *input/output* merupakan fondasi pembelajaran logika algoritma serta pemrograman interaktif dalam bahasa Java. Dengan demikian, program ini tidak hanya menerapkan teori *random* dan *looping* secara efektif, melainkan juga memperlihatkan praktik pengembangan aplikasi yang responsif dan mendidik pemahaman dasar pemrograman Java secara terstruktur.

2.2 Flowchart



Gambar 2.9 (Flowchart)

Flowchart di atas menjelaskan alur program permainan sederhana yaitu lempar dua buah dadu. Program diawali dengan inisialisasi t input, pengacak angka, dan variabel percobaan. Kemudian, dua dadu dilempar secara acak dengan nilai antara 1 sampai 6, hasilnya dijumlahkan, dan jumlah percobaan bertambah satu. Program menampilkan hasil penjumlahan kedua dadu tersebut kepada pengguna.

Selanjutnya, program memeriksa apakah jumlah kedua dadu sama dengan tujuh. Jika iya, maka program menampilkan pesan “Tebakan Anda benar” beserta jumlah percobaan, lalu program berakhir. Jika tidak, program menampilkan pesan “Tebakan Anda salah” dan menanyakan apakah pengguna ingin mencoba lagi. Jika pengguna memilih berhenti, maka program menampilkan pesan “Anda gagal menang” dan selesai. *Flowchart* ini menggambarkan penggunaan logika percabangan dan perulangan dalam permainan sederhana berbasis lempar dadu.

2.3 Pseudocode

Judul: Program tugasAlproPekan6 {Program untuk menghitung total dari banyaknya percobaan melempar dadu}
Deklarasi: input : Scanner; rand : Random; jawab : String; dadu1, dadu2, jumlah, percobaan : Integer; menang : Boolean;

Pseudocode:

```
1. Start
2.    input ← new Scanner(System.in)
3.    rand ← new Random()
4.    percobaan ← 0
5.    menang ← false
6.    do
7.       dadu1 ← rand.nextInt(6) + 1
8.       dadu2 ← rand.nextInt(6) + 1
9.       jumlah ← dadu1 + dadu2
10.        percobaan ← percobaan + 1
11.       print(dadu1 + " + " + dadu2 + " = " + jumlah)
12.       if (jumlah == 7) then
13.          print("Tebakan Anda Benar")
14.          print("Anda menang setelah " + percobaan + " percobaan!")
15.          menang ← true
16.          break
17.       else
18.          print("Tebakan Anda Salah")
19.          print("Apakah mau lempar dadu (ya / tidak?)")
20.          jawab ← input.nextLine()
21.          if (jawab.equalsIgnoreCase("tidak")) then
22.             print("Anda gagal menang")
23.             break
24.          endif
25.       endif
26.    while (true)
27.    input.close()
28. End
```

Tabel 2.1 (Pseudocode)

Pseudocode ini menggunakan struktur *if* dan *do-while* untuk mengatur logika permainan. Setiap kali dadu dilempar, program memeriksa apakah hasilnya sama dengan 7. Jika belum, program akan menanyakan kepada pengguna apakah ingin mencoba lagi. Kelebihannya adalah alur logika mudah dipahami dan menyerupai kode Java sebenarnya.

BAB III
KESIMPULAN

3.1 Hasil Praktikum

3.1.1 Output Menang

```
<terminated> tugasAlproPekan6_2511531016 [Java Applic
1 + 1 = 2
Tebakan Anda Salah
Apakah mau lempar dadu (ya / tidak?) ya
1 + 3 = 4
Tebakan Anda Salah
Apakah mau lempar dadu (ya / tidak?) ya
2 + 6 = 8
Tebakan Anda Salah
Apakah mau lempar dadu (ya / tidak?) ya
2 + 2 = 4
Tebakan Anda Salah
Apakah mau lempar dadu (ya / tidak?) ya
2 + 2 = 4
Tebakan Anda Salah
Apakah mau lempar dadu (ya / tidak?) ya
6 + 1 = 7
Tebakan Anda Benar
Anda menang setelah 6 percobaan!
```

Gambar 3.1 (Hasil Praktikum Output Menang)

3.1.2 Output Gagal Menang

```
<terminated> tugasAlproPekan6_2511531016 [Java Applicatio
3 + 3 = 6
Tebakan Anda Salah
Apakah mau lempar dadu (ya / tidak?) tidak
Anda gagal menang
```

Gambar 3.2 (Hasil Praktikum Output Gagal Menang)

Output di atas memperlihatkan proses interaktif dari program permainan lempar dadu berbasis Java. Pada awal setiap percobaan, dua angka acak mewakili lemparan dua dadu ditampilkan beserta hasil penjumlahannya, misalnya “1 + 1 = 2” atau “1 + 3 = 4”. Jika hasil penjumlahan belum tepat 7, program menampilkan pesan “Tebakan Anda Salah” dan memberikan pertanyaan kepada pengguna: “Apakah mau lempar dadu (ya / tidak?)”. Jawaban “ya” dari pengguna menyebabkan perulangan proses dan lemparan dadu berikutnya, yang terjadi beberapa kali hingga akhirnya hasil dadu mencapai angka 7 (“6 + 1 = 7”).

Ketika penjumlahan dua dadu tepat 7, program memberikan respon “Tebakan Anda Benar” dan menampilkan pesan kemenangan lengkap dengan jumlah percobaan yang sudah dilakukan, misalnya “Anda menang setelah 6 percobaan!”. Namun jika pengguna langsung menjawab “tidak” pada percobaan pertama, maka langsung menampilkan pesan “Anda gagal menang”.

Output ini secara langsung menggambarkan bagaimana program menggunakan perulangan *(loop)*, pengkondisian *(if–else)*, serta interaksi input/output untuk mengontrol alur permainan berbasis logika dan keputusan pengguna. Setiap langkah dieksekusi secara sistematis dan hasilnya transparan.

3.2 Saran Pengembangan

Saran pengembangan untuk program ini adalah dapat difokuskan pada hal-hal sederhana seperti menambahkan tampilan teks yang lebih menarik, misalnya menggunakan garis pemisah atau pesan motivasi setiap kali pemain menang atau kalah agar permainan terasa lebih interaktif.

Selain itu, program juga dapat dibuat menampilkan jumlah total kemenangan dan kekalahan setelah permainan selesai, sehingga pemain bisa mengetahui hasil akhirnya. Penggunaan validasi input juga penting agar program tidak *error* ketika pengguna memasukkan huruf atau simbol selain angka. Saran lain adalah menambahkan batas percobaan, misalnya maksimal lima kali lemparan sebelum permainan berakhir otomatis, agar lebih menantang.

Dengan pengembangan sederhana seperti ini, program akan tetap mudah dipahami tetapi juga dapat menjadi contoh penerapan logika tampilan teks yang lebih menarik, interaktif untuk dimainkan dan dinamis dalam bahasa pemrograman Java.

DAFTAR PUSTAKA

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

[2] J. Bloch, *Effective Java*, 3rd ed. Boston, MA: Addison-Wesley, 2018.

[3] Oracle, *"Class Random (Java Platform SE 8),"* Oracle Documentation, 2015. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

[4] W. Stallings, *Foundations of Computer Science: From Data Manipulation to Theory of Computation,* 4th ed. New York, NY: Pearson, 2015.

[5] D. Parsons, *Introduction to Programming with Java: A Problem Solving* Approach, 2nd ed. Boston, MA: Cengage Learning, 2017.

[6] R. Lafore, *Object-Oriented Programming in Java*, 5th ed. Indianapolis, IN: Sams Publishing, 2018.