

**LAPORAN PRATIKUM**  
**ALGORITMA DAN PEMROGRAMAN**  
**“STRUKTUR KONTROL DAN PERCABANGAN**  
**PADA BAHASA PEMROGRAMAN JAVA”**

**disusun Oleh:**  
**RAZIF AL FARISI**  
**NIM 2511532028**  
**Dosen Pengampu: Dr.Wahyudi S.T, M.T**  
**Asisten Laboratorium: Rahmad Dwirizki Olders**



**DEPARTEMEN INFORMATIKA**  
**FAKULTAS TEKNOLOGI INFORMASI**  
**UNIVERSITAS ANDALAS**  
**PADANG**  
**2025**

# KATA PENGANTAR

Segala puji bagi Allah SWT, Tuhan Yang Maha Esa, karena atas limpahan rahmat, taufik, dan hidayah-Nya, laporan praktikum dengan judul “**Struktur Kontrol atau Percabangan pada Bahasa Pemrograman Java**” dapat terselesaikan dengan baik sesuai waktu yang telah ditentukan. Tanpa pertolongan dan izin-Nya, niscaya penyusunan laporan ini tidak akan berjalan dengan lancar.

Laporan ini disusun sebagai salah satu bentuk tanggung jawab akademik dalam mengikuti kegiatan praktikum Algoritma dan Pemrograman. Di samping itu, laporan ini juga dimaksudkan sebagai sarana untuk mendalami serta mengasah keterampilan dalam memahami konsep dasar pemrograman, khususnya mengenai *control structure* atau struktur kontrol dengan fokus pada percabangan (*branching*) dalam bahasa pemrograman *Java*. Melalui laporan ini diharapkan pemahaman tentang logika pemrograman tidak hanya sebatas teori, tetapi juga dapat diaplikasikan dalam bentuk program nyata yang bermanfaat.

Ucapan terima kasih ditujukan kepada semua pihak yang telah memberikan bantuan, dukungan, dan dorongan selama proses penyusunan laporan ini berlangsung. Dukungan tersebut dapat berupa bimbingan, saran, maupun motivasi, baik dari dosen, asisten laboratorium, maupun rekan seperjuangan. Kehadiran mereka memberikan kontribusi yang sangat berarti dalam keberhasilan penyelesaian laporan ini

Harapan yang besar semoga laporan ini dapat memberikan manfaat, baik sebagai bahan kajian tambahan, referensi pembelajaran, maupun sumber pengetahuan yang berguna bagi siapa saja yang ingin mendalami struktur kontrol dalam bahasa pemrograman *Java*. Dengan demikian, pemahaman terhadap logika pemrograman dapat berkembang lebih komprehensif dan dapat diterapkan secara luas dalam berbagai bidang ilmu maupun praktik pemrograman.

Padang, 03 Oktober 2025

Penulis

# Daftar Isi

<b>KATA PENGANTAR .....</b>	i
<b>BAB I .....</b>	1
<b>PENDAHULUAN .....</b>	1
1.1    Latar Belakang .....	1
1.2    Tujuan .....	2
1.3    Manfaat .....	2
<b>BAB II .....</b>	3
<b>PEMBAHASAN.....</b>	3
2.1    Struktur Kontrol Percabangan.....	3
2.1.1    If – Else .....	3
2.1.2    Switch – Case .....	4
2.2    Langkah Pengerjaan .....	5
2.2.1    If – else .....	5
2.2.2    Multi If.....	6
2.2.3    Switch – Case .....	7
<b>BAB III.....</b>	10
<b>PENUTUP .....</b>	10
3.1    Kesimpulan .....	10
3.2    Saran.....	10
<b>DAFTAR PUSTAKA.....</b>	11

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Perkembangan teknologi informasi pada era modern telah memberikan dampak signifikan terhadap berbagai aspek kehidupan, baik dalam bidang pendidikan, ekonomi, industri, maupun pemerintahan. Salah satu faktor utama yang mendorong perkembangan teknologi tersebut adalah bahasa pemrograman. Bahasa pemrograman berfungsi sebagai media komunikasi antara manusia dan komputer, sehingga instruksi yang dibuat dapat dipahami serta dijalankan oleh mesin.

Salah satu konsep fundamental dalam pemrograman adalah *control structure* atau struktur kontrol. Struktur kontrol merupakan mekanisme untuk mengatur alur eksekusi perintah dalam sebuah program. Secara umum, struktur kontrol terbagi menjadi tiga bentuk, yaitu:

1. Struktur urutan (*sequence*), yaitu perintah yang dieksekusi secara berurutan dari atas ke bawah.
2. Struktur percabangan (*selection* atau *branching*), yaitu perintah yang dijalankan berdasarkan kondisi tertentu.
3. Struktur perulangan (*looping*), yaitu perintah yang dijalankan secara berulang selama memenuhi kondisi tertentu.

Fokus kajian dalam laporan ini adalah struktur percabangan (*selection*). Percabangan memungkinkan sebuah program untuk mengambil keputusan, sehingga alur eksekusi dapat disesuaikan dengan kondisi yang terjadi. Misalnya, sebuah program dapat menentukan apakah seorang siswa dinyatakan lulus atau tidak lulus berdasarkan nilai yang dimasukkan. Dengan demikian, percabangan memberikan fleksibilitas pada program dan menjadikannya lebih dinamis.

Pemahaman tentang percabangan sangat penting bagi mahasiswa, karena logika pengambilan keputusan merupakan dasar dari hampir semua aplikasi perangkat lunak. Tanpa pemahaman ini, penyusunan program hanya akan berjalan secara linear, sehingga tidak mampu mengakomodasi variasi kondisi yang lebih kompleks.

## 1.2 Tujuan

Tujuan dari penyusunan laporan praktikum ini adalah sebagai berikut:

1. Menjelaskan konsep struktur kontrol, khususnya percabangan dalam bahasa pemrograman *Java*.
2. Menguraikan jenis-jenis percabangan yang terdapat dalam bahasa *Java*, seperti *if statement*, *if-else statement*, *else-if ladder*, *nested if*, dan *switch-case*.
3. Memberikan contoh implementasi percabangan dalam program sederhana berbasis *Java*.
4. Menganalisis hasil keluaran program berdasarkan kondisi yang diberikan.

## 1.3 Manfaat

1. Sebagai bahan pembelajaran bagi mahasiswa untuk memahami konsep percabangan dalam bahasa pemrograman *Java*.
2. Sebagai referensi tambahan bagi dosen maupun asisten praktikum dalam memberikan pengajaran terkait struktur kontrol pada pemrograman.
3. Sebagai acuan bagi pembaca umum yang ingin memperdalam pengetahuan mengenai logika percabangan pada *Java*.

# BAB II

## PEMBAHASAN

### 2.1 Struktur Kontrol Percabangan

Dalam pemrograman, struktur kontrol percabangan adalah mekanisme untuk menentukan alur eksekusi program berdasarkan suatu kondisi tertentu. Dengan percabangan, program dapat mengambil keputusan yang berbeda tergantung nilai atau keadaan yang diperiksa. Secara umum, struktur kontrol percabangan dibagi menjadi dua tipe utama :

1. *If – else*
2. *Switch - case*

#### 2.1.1 If – Else

Struktur *if – else* digunakan untuk mengeksekusi perintah tertentu jika kondisi yang diberikan bernilai benar (*true*). Jika kondisi bernilai salah (*false*), maka blok *else* akan dijalankan.

1. *If*
  - Digunakan untuk memeriksa suatu kondisi
  - Sintaks (*Java*) : `if (kondisi) { /* perintah */ }`
  - Catatan : (*kondisi*) harus menghasilkan *true/false*.
2. *Else*
  - Digunakan jika nilai *if* bernilai *false*
  - Sintaks: `if (...) { ... } else { ... }`
  - Catatan: *else* berpasangan dengan *if* terdekat yang belum punya *else* (masalah *dangling else*).
3. *Else – If*
  - Digunakan untuk memeriksa kondisi tambahan setelah *if* pertama *false*
  - Struktur: `else if (kondisi) { ... }`
  - hanya **satu** cabang dalam rantai *if–else if...–else* yang akan dieksekusi yang *pertama* bernilai *true*.

#### 4. Nested If

- *If* di dalam *if*. Berguna bila kondisi kedua relevan hanya jika kondisi pertama terpenuhi. Hati-hati kedalaman bersarang bila terlalu dalam, refactor ke fungsi.

#### 5. Multi If

- Semua *if* diperiksa secara *independent* : **lebih dari satu** blok bisa dieksekusi bila beberapa kondisi benar. Gunakan ini saat kondisi tidak saling eksklusif.

### 2.1.2 Switch – Case

Struktur *switch* membandingkan satu ekspresi dengan beberapa *case* bernilai konstanta dan mengeksekusi blok yang cocok. Cocok saat memilih berdasarkan nilai tetap (konstan).

#### 1. Switch

- Fungsi: mengeksekusi blok berdasarkan hasil evaluasi satu ekspresi.
- Sintaks dasar :

```
switch (ekspresi) {  
    case konstanta1:  
        // perintah  
        break;  
    case konstanta2:  
        // perintah  
        break;  
    default:  
        // perintah bila tidak ada yang cocok  
}
```

- Ekspresi dievaluasi sekali, lalu dibandingkan dengan setiap *case*.

#### 2. Case

- Menyatakan nilai konstanta yang dibandingkan dengan ekspresi *switch*.
- Bisa menumpuk *case* untuk melakukan *grouping* (contoh: case 6: case 7: untuk weekend).

#### 3. Break

- Menghentikan eksekusi di dalam *switch* dan keluar dari struktur.
- Jika dihilangkan, eksekusi akan **fall-through** ke *case* berikutnya (sifat yang sering dimanfaatkan tapi juga sumber *bug*).

#### 4. Default

- Cabang fallback jika tidak ada *case* yang cocok.
- Opsional tapi direkomendasikan untuk menangani keadaan tak terduga.

### 5. *Fall-through*

- ketika satu *case* tidak punya break, eksekusi berlanjut ke *case* selanjutnya.
- Digunakan untuk menggabungkan hasil.

## 2.2 Langkah Pengerjaan

### 2.2.1 If – else

1. **Inisialisasi variable** : int nilai ; digunakan untuk menyimpan input angka dari pengguna.
2. **Input data**: pengguna diminta memasukkan angka nilai melalui Scanner.
3. **Proses percabangan**: program memeriksa kondisi if secara berurutan:
  - Jika nilai  $\geq 81$ , maka hasil "A".
  - Jika tidak, periksa nilai  $> 70$ , hasil "B".
  - Jika tidak, periksa nilai  $> 60$ , hasil "C".
  - Jika tidak, periksa nilai  $> 50$ , hasil "D".
  - Jika semua kondisi salah, maka otomatis "E".
4. **Output hasil**: program menampilkan huruf nilai sesuai rentang angka.
5. **Kode Program** :

```
import java.util.Scanner;

public class Nilai {
    public static void main(String[] args) {
        int nilai;
        Scanner input = new Scanner(System.in);
        System.out.print("Inputkan Nilai Angka = ");
        nilai = input.nextInt();
        input.close();

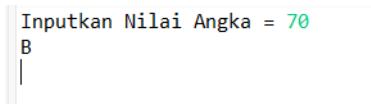
        if(nilai >= 81) {
            System.out.println("A");
        } else if (nilai > 70) {
            System.out.println("B");
        } else if (nilai > 60) {
            System.out.println("C");
        }
    }
}
```

```

        } else if (nilai > 50) {
            System.out.println("D");
        } else {
            System.out.println("E");
        }
    }
}

```

#### 6. Hasil Program :



```

Inputkan Nilai Angka =
B
|

```

#### 2.2.2 Multi If

1. **Deklarasi variabel:** int umur dan char sim digunakan untuk menyimpan input usia dan status kepemilikan SIM.
2. **Input data:** pengguna diminta memasukkan umur (angka) dan status SIM (y untuk ya, t untuk tidak).
3. **Proses percabangan:** beberapa pernyataan if digunakan **tanpa else**, sehingga lebih dari satu kondisi bisa benar sekaligus.
  - Jika  $umur \geq 17$  dan punya SIM → "Sudah Dewasa dan Boleh Bawa Motor".
  - Jika  $umur \geq 17$  dan belum punya SIM → "Sudah Dewasa tetapi Belum Boleh Bawa Motor".
  - Jika  $umur < 17$  dan tidak punya SIM → "Belum Cukup Umur Untuk Membawa Motor".
  - Jika  $umur < 17$  dan punya SIM → "Belum Cukup Umur Untuk Mempunyai SIM".
4. **Output hasil:** menampilkan pesan sesuai kondisi yang terpenuhi.

#### 5. Kode Program :

```

import java.util.Scanner;

public class MultiIf {
    public static void main(String[] args) {
        int umur;
        char sim;
        Scanner a = new Scanner(System.in);

        System.out.print("Input umur anda : ");
        umur = a.nextInt();
    }
}

```

```

        System.out.print("Apakah Anda Sudah Punya SIM
: ");
        sim = a.next().charAt(0);
        a.close();

        if ((umur >= 17) && (sim == 'y')) {
            System.out.println("Anda Sudah Dewasa dan
Boleh Bawa Motor");
        }
        if ((umur >= 17) && (sim != 'y')) {
            System.out.println("Anda Sudah Dewasa
Tetapi Belum Boleh Bawa Motor");
        }
        if ((umur < 17) && (sim != 'y')) {
            System.out.println("Anda Belum Cukup Umur
Untuk Membawa Motor");
        }
        if ((umur < 17) && (sim == 'y')) {
            System.out.println("Anda Belum Cukup Umur
Untuk Mempunyai SIM");
        }
    }
}

```

## 6. Hasil Program :

```

Input umur anda : 16
Apakah Anda Sudah Punya SIM : y
Anda Belum Cukup Umur Untuk Mempunyai SIM

```

### 2.2.3 Switch – Case

- Deklarasi objek Scanner** “Scanner scanner = new Scanner(System.in);”.  
Digunakan untuk menerima input angka dari pengguna.
- Input data** : program meminta pengguna memasukkan angka bulan dari 1–12.
- Proses percabangan dengan *switch-case*** :
  - Nilai variabel bulan diperiksa melalui struktur *switch*.
  - Tiap *case* mewakili angka tertentu (1 sampai 12).
  - Pada tiap *case*, program mencetak nama bulan sesuai nomor.
  - Kata kunci *break* ; dipakai untuk menghentikan eksekusi agar tidak jatuh (*fall-through*) ke *case* berikutnya.

4. **Default case** : Jika nilai bulan tidak berada dalam rentang 1–12, maka *default* akan dieksekusi dan menampilkan "Angka Tidak Valid".

5. **Menutup Scanner** "scanner.close();".

6. **Kode Program :**

```
import java.util.Scanner;

public class NamaBulan {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan Angka Bulan (1-12) : ");
        int bulan = scanner.nextInt();

        switch (bulan) {
            case 1: System.out.println("Januari");
            break;
            case 2: System.out.println("Februari");
            break;
            case 3: System.out.println("Maret");
            break;
            case 4: System.out.println("April");
            break;
            case 5: System.out.println("Mei");
            break;
            case 6: System.out.println("Juni");
            break;
            case 7: System.out.println("Juli");
            break;
            case 8: System.out.println("Agustus");
            break;
            case 9: System.out.println("September");
            break;
            case 10: System.out.println("Oktober");
            break;
            case 11: System.out.println("November");
            break;
            case 12: System.out.println("Desember");
            break;
            default: System.out.println("Angka Tidak Valid");
        }
        scanner.close();
    }
}
```

}

7. Hasil Program :

```
Masukkan Angka Bulan (1-12) : 2
Februari
```

## **BAB III**

## **PENUTUP**

### **3.1 Kesimpulan**

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa struktur kontrol percabangan berfungsi untuk menentukan alur program sesuai kondisi yang diberikan. Melalui penggunaan *if*, *if – else*, maupun *switch*, program dapat mengambil Keputusan dengan benar sehingga menghasilkan output yang sesuai.

### **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**

Oracle. (2024). *The Java™ Tutorials: Control Flow Statements*. Oracle Official Documentation.

Wahana Komputer. (2022). *Pemrograman Java untuk Pemula*. Yogyakarta: Andi Offset.

Raharjo, B. (2017). *Belajar Java untuk Pemula*. Bandung: Informatika.

Nugroho, B. (2019). *Dasar Pemrograman Java*. Yogyakarta: Andi.

Supardi. (2020). *Algoritma dan Pemrograman dengan Java*. Jakarta: PT Elex Media Komputindo.