

LAPORAN
PEMROGRAMAN BERORIENTASI OBJEK



Nama : Narendra Awangga
Stambuk : 13020220022
Nama Dosen : Mardiyah Hasnawi, S.Kom.,M.T.
Kelas : A1

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS MUSLIM INDONESIA
MAKASSAR
2024

1. Apakah perbedaan antara struktur kontrol percabangan if-else dan switch-case?

Dalam pemrograman, terdapat dua cara untuk mengambil keputusan berdasarkan kondisi yang ada, yaitu menggunakan switch case dan if else. Kedua metode ini memiliki perbedaan dalam hal sintaksis, kegunaan, dan performa. Dalam artikel ini, kita akan menjelajahi perbedaan antara switch case dan if else secara komprehensif.

Sintaksis

Sintaksis adalah tata bahasa yang digunakan untuk menulis kode dalam pemrograman. Switch case dan if else memiliki sintaksis yang berbeda:

- Switch case:

```
switch (expression) {  
  case value1:  
    // statement untuk value1  
    break;  
  case value2:  
    // statement untuk value2  
    break;  
  default:  
    // statement default  
}
```
- If else:

```
if (condition) {  
  // statement jika kondisi benar  
} else if (condition2) {  
  // statement jika kondisi2 benar  
} else {  
  // statement jika kondisi salah  
}
```

Perbedaan sintaksis inilah yang menjadi pembeda utama antara switch case dan if else. Switch case menggunakan kata kunci “switch”, “case”, dan “break” untuk menentukan cabang mana yang akan dieksekusi, sedangkan if else menggunakan kata kunci “if”, “else if”, dan “else”.

Kelebihan dan Kekurangan

Masing-masing metode memiliki kelebihan dan kekurangan yang perlu dipertimbangkan sebelum memilih salah satunya:

Switch Case

Kelebihan:

- Sangat berguna ketika ada banyak pilihan yang mungkin terjadi dan kode yang berbeda harus dieksekusi untuk setiap pilihan.
- Lebih mudah dibaca dan dimengerti ketika ada banyak kondisi.
- Lebih efisien dalam beberapa kasus dibandingkan if else.

Kekurangan:

- Tidak bisa mengevaluasi ekspresi yang kompleks atau mengevaluasi kondisi yang melibatkan rentang nilai.
- Tidak bisa mengevaluasi kondisi berupa variabel float atau string.
- Tidak ada kemampuan untuk melakukan kondisi yang bersarang.

If Else

Kelebihan:

- Dapat mengevaluasi kondisi yang kompleks dan berdistribusi seperti rentang nilai.
- Dapat mengevaluasi semua tipe data, termasuk float dan string.
- Bisa melakukan kondisi yang bersarang.

Kekurangan:

- Lebih sulit dibaca dan dipahami ketika ada banyak kondisi.
- Lebih tidak efisien dalam beberapa kasus, terutama ketika ada banyak kondisi yang harus dievaluasi.

Penanganan Kondisi

Selain perbedaan sintaksis dan kelebihan/kekurangan, switch case dan if else juga memiliki perbedaan dalam penanganan kondisi:

Dalam switch case, ekspresi yang diberikan akan memilih salah satu cabang yang sesuai dengan nilai ekspresi tersebut. Jika tidak ada kesesuaian dengan nilai yang ada di case-case, maka akan dieksekusi statement di dalam blok default. Jika tidak ada statement break, eksekusi akan “jatuh” ke case-case berikutnya. Switch case cocok digunakan ketika ada banyak pilihan yang mungkin terjadi.

Di sisi lain, if else mengevaluasi kondisi secara berurutan dari atas ke bawah. Jika kondisi pada if benar, maka statement di dalam blok if akan dieksekusi, dan blok else akan diabaikan. Jika kondisi pada if salah, maka kondisi pada else if akan diperiksa. Proses ini berlanjut sampai salah satu kondisi bernilai benar atau semua kondisi bernilai salah, dalam hal ini statement di dalam blok else akan dieksekusi. If else cocok digunakan ketika ada logika pemilihan yang kompleks dan banyak kondisi yang harus dievaluasi.

Kesimpulan

Dalam pemrograman, pemilihan kondisi dapat dilakukan dengan menggunakan switch case atau if else. Kedua metode ini memiliki perbedaan dalam sintaksis, kegunaan, dan cara penanganan kondisi. Memilih metode yang tepat tergantung pada kompleksitas kondisi yang ingin ditangani. Jika terdapat banyak pilihan yang mungkin terjadi, switch case merupakan pilihan yang baik. Namun, jika terdapat logika pemilihan yang kompleks dan banyak kondisi yang harus dievaluasi, if else dapat digunakan. Penting untuk memahami perbedaan antara keduanya agar bisa memilih metode yang paling sesuai dengan kebutuhan.

2. Kapan digunakan struktur kontrol if-else dan switch-case

Struktur kontrol if-else dan switch-case digunakan dalam pemrograman untuk mengatur alur program berdasarkan kondisi tertentu. Berikut adalah penjelasan singkat kapan masing-masing struktur kontrol tersebut digunakan:

1. if-else:

- **Penggunaan:** If-else digunakan ketika kita perlu membuat keputusan berdasarkan kondisi tertentu. Kondisi ini dapat berupa ekspresi boolean yang bernilai true atau false.
- **Cara Kerja:** Jika kondisi dalam if statement dievaluasi sebagai true, maka blok kode di dalam if akan dieksekusi. Jika kondisi dalam if statement dievaluasi sebagai false, maka blok kode di dalam else (jika ada) akan dieksekusi.
- **Contoh:** Misalnya, kita menggunakan if-else untuk memeriksa apakah suatu angka adalah positif atau negatif:

```
public class IfElseExample {  
    public static void main(String[] args) {  
        int angka = 10;  
  
        if (angka > 0) {  
            System.out.println("Angka adalah positif");  
        } else {  
            System.out.println("Angka adalah negatif");  
        }  
    }  
}
```

2. switch-case:

- **Penggunaan:** Switch-case digunakan ketika terdapat beberapa kondisi yang mungkin terjadi dan kita ingin mengevaluasi suatu ekspresi terhadap beberapa nilai konstan atau case.
- **Cara Kerja:** Ekspresi yang dievaluasi akan dibandingkan dengan nilai-nilai konstan yang terdaftar dalam case-case. Jika nilai ekspresi cocok dengan salah satu case, blok kode di bawah case tersebut akan

dieksekusi. Jika tidak ada case yang cocok, blok kode di dalam default (jika ada) akan dieksekusi.

- **Catatan:** Beberapa bahasa pemrograman tidak memiliki struktur switch-case atau memiliki sintaks alternatif untuk melakukan hal yang serupa.
- **Contoh:** Contoh switch-case dalam java :

```
public class SwitchCaseExample {  
    public static void main(String[] args) {  
        int bulan = 3;  
        String namaBulan;  
  
        switch (bulan) {  
            case 1:  
                namaBulan = "Januari";  
                break;  
            case 2:  
                namaBulan = "Februari";  
                break;  
            case 3:  
                namaBulan = "Maret";  
                break;  
            // dan seterusnya  
            default:  
                namaBulan = "Bulan tidak valid";  
                break;  
        }  
  
        System.out.println("Bulan: " + namaBulan);  
    }  
}
```

3. Pada program 2, tambahkan perintah untuk memilih 2 opsi menggunakan kontrol switch..case. opsi pilihah

1=inputNilai() Pilihan
2=inputNilaiBaru()

- **Kelas utama**

package hitungrata;

//Nama : Narendra Awangga

//Nim : 13020220022

//Kelas : A1 Teknik Informatika

//Tugas : 4 Evaluasi Praktikum

```
import java.util.*;

public class HitungRata {

    private double total = 0.0;

    private ArrayList nilaiBaru = new ArrayList();

    Scanner input = new Scanner(System.in);

    public void inputNilai(int nilai[]) {

        for (int i = 0; i < nilai.length; i++) {

            nilai[i] = input.nextInt();

            total += nilai[i];

        }

    }

    public double rataNilai(int Ndata) {

        return total / ((double) Ndata);

    }

    public void cetakNilai(int nilai[]) {

        for (int angka : nilai) {

            System.out.print(angka);

        }

        System.out.println();

    }

    public void inputNilaiBaru(int jumlah) {

        while (jumlah > 0) {

            nilaiBaru.add(input.nextInt());

        }

    }

}
```

```

        jumlah--;
    }
}

public void cetakNilaiBaru() {
    ListIterator i = nilaiBaru.listIterator(0);

    while (i.hasNext()) {
        Object data = i.next();

        if (data == null) {
            System.out.println("null");
        } else {
            System.out.println(data.toString());
        }
    }
}
}

```

- **Program 2**

```

package hitungrata;

import java.util.Scanner;
public class TestNilai {
    public static void main(String[] args) {
        HitungRata hitung = new HitungRata();
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan Jumlah Data : ");
        int banyakData = input.nextInt();
        int nilai[] = new int[banyakData];
        int pilihan;

        do {
            System.out.println("Menu:");
            System.out.println("1. Input Nilai");
            System.out.println("2. Input Nilai Baru");

```

```

System.out.println("3. Keluar");
System.out.print("Pilihan Anda: ");
pilihan = input.nextInt();
switch (pilihan) {
    case 1:
        System.out.print("Masukkan Nilai : ");
        hitung.inputNilai(nilai);
        System.out.print("Daftar Nilai : ");
        hitung.cetakNilai(nilai);
        System.out.println("Rata Nilai : "+
hitung.rataNilai(banyakData));
        break;
    case 2:
        System.out.print("Masukkan Nilai Baru: ");
        hitung.inputNilaiBaru(banyakData);
        System.out.print("Daftar Nilai Baru : ");
        hitung.cetakNilaiBaru();
        break;
    case 3:
        System.out.println("Keluar dari Program.");
        break;
    default:
        System.out.println("Pilihan tidak valid. Silakan pilih lagi.");
        break;
}
} while (pilihan != 3);
}
}

```

- **Output**

```

Output x
HitungRata (run) x HitungRata (run) #2 x
run:
Masukkan Jumlah Data : 3
Menu:
1. Input Nilai
2. Input Nilai Baru
3. Keluar
Pilihan Anda: 1
Masukkan Nilai : 2
2
3
Daftar Nilai : 223
Rata Nilai : 2.3333333333333335

Menu:
1. Input Nilai
2. Input Nilai Baru
3. Keluar
Pilihan Anda: 2
Masukkan Nilai Baru: 2
3
3
Daftar Nilai Baru : 2
3
3
Menu:
1. Input Nilai
2. Input Nilai Baru
3. Keluar
Pilihan Anda:

```


4. Apakah perbedaan antara struktur kontrol perulangan **while** dan **do-while**?

Struktur kontrol perulangan **while** dan **do-while** adalah dua bentuk perulangan dalam pemrograman. Meskipun keduanya digunakan untuk menjalankan serangkaian pernyataan secara berulang selama kondisi tertentu terpenuhi, ada perbedaan kunci antara keduanya:

- **while**: Pada struktur **while**, kondisi evaluasi diperiksa sebelum blok pernyataan dieksekusi. Jika kondisi benar (true) saat pertama kali dievaluasi, maka blok pernyataan akan dieksekusi. Jika kondisi salah (false) dari awal, maka blok pernyataan tidak akan pernah dieksekusi. Dengan kata lain, eksekusi blok pernyataan tergantung pada kondisi evaluasi.

Contoh:

```
int i = 0;

while (i < 5) {

    System.out.println(i);

    i++;

}
```

Pada contoh di atas, kondisi **i < 5** akan dievaluasi terlebih dahulu sebelum eksekusi blok pernyataan.

- **do-while**: Pada struktur **do-while**, blok pernyataan dieksekusi setidaknya satu kali tanpa memeriksa kondisi terlebih dahulu. Setelah eksekusi blok pernyataan, kondisi dievaluasi. Jika kondisi benar (true), maka perulangan akan dilanjutkan. Jika kondisi salah (false), perulangan akan berhenti.

Contoh:

```
int i = 0;

do {

    System.out.println(i);

    i++;

} while (i < 5);
```

Pada contoh di atas, blok pernyataan akan dieksekusi sekali bahkan jika kondisi `i < 5` salah dari awal. Setelah itu, kondisi akan diperiksa, dan jika benar, perulangan akan berlanjut.

Jadi, perbedaan utama antara keduanya adalah saat kondisi dievaluasi: pada **while**, kondisi dievaluasi sebelum eksekusi blok pernyataan, sedangkan pada **do-while**, blok pernyataan dieksekusi setidaknya satu kali sebelum kondisi dievaluasi.

5. Kapan digunakan struktur kontrol **for**?

Struktur kontrol **for** sering digunakan ketika kita tahu berapa kali perulangan harus dijalankan atau jika kita ingin mengulangi suatu blok kode dengan mengontrol variabel iterasi. Beberapa situasi di mana struktur kontrol **for** cocok digunakan meliputi:

- **Perulangan dengan jumlah iterasi yang diketahui:** Ketika Anda tahu persis berapa kali blok kode harus diulangi, Anda dapat menggunakan struktur **for** untuk mengatur iterasi berdasarkan variabel iterasi.

Contoh:

```
for (int i = 0; i < 5; i++) {  
  
    System.out.println("Iterasi ke-" + i);  
  
}
```

Pada contoh di atas, kita tahu bahwa kita ingin melakukan iterasi sebanyak lima kali.

1. **Iterasi melalui elemen koleksi:** Struktur **for** sangat berguna ketika Anda perlu melakukan iterasi melalui elemen-elemen suatu array, daftar, atau koleksi lainnya. Ini memungkinkan Anda untuk dengan mudah mengakses setiap elemen dalam struktur data tersebut.

Contoh:

```
int[] numbers = { 1, 2, 3, 4, 5 };  
  
for (int i = 0; i < numbers.length; i++) {  
  
    System.out.println(numbers[i]);  
  
}
```

2. **Perulangan dalam rentang tertentu:** Struktur **for** juga digunakan ketika Anda perlu melakukan iterasi dalam rentang tertentu, misalnya dari suatu nilai awal hingga suatu nilai akhir.

Contoh:

```
for (int i = 10; i >= 0; i--) {  
  
    System.out.println(i);  
  
}
```

Pada contoh di atas, kita melakukan iterasi dari 10 hingga 0.

Struktur kontrol **for** memberikan sintaks yang ringkas dan jelas untuk mengatur perulangan dalam berbagai situasi yang berbeda, terutama ketika jumlah iterasi atau rentang iterasi sudah diketahui.

6. Apakah perbedaan antara Array dan ArrayList ? berilah contoh masing-masing!

Array dan ArrayList adalah dua struktur data yang digunakan untuk menyimpan sekumpulan elemen dalam pemrograman Java, tetapi ada perbedaan penting antara keduanya:

1. **Array:**

- Array adalah struktur data yang dapat menyimpan sejumlah elemen dengan tipe data yang sama.
- Ukuran array ditentukan pada saat deklarasi dan tidak dapat berubah setelahnya.
- Elemen-elemen dalam array diakses menggunakan indeks.
- Array bisa memiliki tipe data primitif (seperti **int**, **double**, **char**, dsb.) atau objek (seperti **String**, **Object**, dsb.).
- Array tidak memiliki metode tambahan untuk menambah atau menghapus elemen secara langsung.

Contoh penggunaan array:

```
// Deklarasi dan inisialisasi array integer  
int[] numbers = { 1, 2, 3, 4, 5};
```

```
// Mengakses elemen array menggunakan indeks  
System.out.println("Elemen pada indeks 0: " + numbers[0]); // Output: 1
```

2. ArrayList:

- ArrayList adalah kelas dalam Java yang merupakan bagian dari paket `java.util`.
- ArrayList adalah struktur data yang menyediakan fleksibilitas dalam ukuran dan jenis elemen yang disimpan di dalamnya.
- Ukuran ArrayList dapat berubah secara dinamis saat elemen-elemen ditambahkan atau dihapus.
- Elemen-elemen dalam ArrayList diindeks dengan menggunakan metode `get(index)` atau `set(index, element)`.
- ArrayList hanya dapat menyimpan objek, bukan tipe data primitif. Untuk tipe data primitif, kita harus menggunakan wrapper class (misalnya, `Integer` untuk `int`).

Contoh penggunaan ArrayList:

```
import java.util.ArrayList;

// Deklarasi dan inisialisasi ArrayList integer
ArrayList<Integer> numberList = new ArrayList<>();

// Menambahkan elemen ke ArrayList
numberList.add(1);
numberList.add(2);
numberList.add(3);

// Mengakses elemen ArrayList menggunakan metode get
System.out.println("Elemen pada indeks 0: " + numberList.get(0)); //
Output: 1
```

Jadi, perbedaan utama antara Array dan ArrayList adalah dalam fleksibilitas ukuran dan jenis elemen yang disimpan serta kemampuan untuk mengubah ukuran dinamisnya

```
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Membuat objek Scanner untuk input dari keyboard
        Scanner scanner = new Scanner(System.in);

        // Membuat objek HashMap untuk menyimpan pasangan nilai dan kunci
        HashMap<String, Integer> hashMap = new HashMap<>();
```

```

// Memasukkan nilai dan kunci ke dalam HashMap
System.out.println("Masukkan pasangan nilai dan kunci (key):");
while (true) {
    System.out.print("Key (tekan Enter untuk berhenti): ");
    String key = scanner.nextLine();
    if (key.isEmpty()) {
        break; // Berhenti jika input kosong
    }
    System.out.print("Nilai: ");
    int value = Integer.parseInt(scanner.nextLine());

    // Memasukkan pasangan nilai dan kunci ke dalam HashMap
    hashMap.put(key, value);
}

// Menampilkan isi HashMap
System.out.println("\nIsi HashMap:");
for (String key : hashMap.keySet()) {
    int value = hashMap.get(key);
    System.out.println("Key: " + key + ", Nilai: " + value);
}

// Menutup objek Scanner
scanner.close();
}
}

```

7. **Buatlah contoh program yang mengimplementasikan HashMap dengan memasukkan nilai dan key melalui keyboard!**

Berikut adalah contoh program Java yang mengimplementasikan **HashMap** untuk memasukkan nilai dan kunci melalui keyboard:

```

import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashMap<String, Integer> hashMap = new HashMap<>();

        System.out.println("Masukkan jumlah data yang ingin dimasukkan:");
        int n = scanner.nextInt();
    }
}

```

```

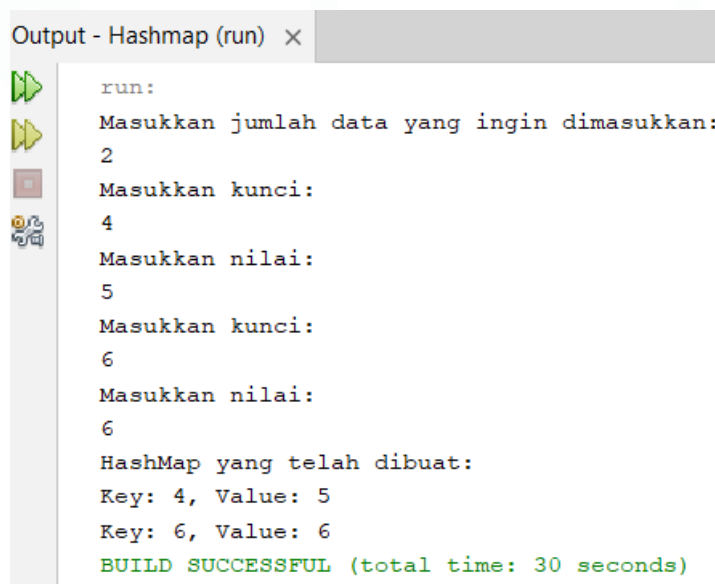
scanner.nextLine();

for (int i = 0; i < n; i++) {
    System.out.println("Masukkan kunci:");
    String key = scanner.nextLine();
    System.out.println("Masukkan nilai:");
    int value = scanner.nextInt();
    scanner.nextLine();
    hashMap.put(key, value);
}

System.out.println("HashMap yang telah dibuat:");
for (String key : hashMap.keySet()) {
    System.out.println("Key: " + key + ", Value: " + hashMap.get(key));
}
}
}

```

- Output



```

Output - Hashmap (run) x
run:
Masukkan jumlah data yang ingin dimasukkan:
2
Masukkan kunci:
4
Masukkan nilai:
5
Masukkan kunci:
6
Masukkan nilai:
6
HashMap yang telah dibuat:
Key: 4, Value: 5
Key: 6, Value: 6
BUILD SUCCESSFUL (total time: 30 seconds)

```

Dalam program ini, kita menggunakan **Scanner** untuk mengambil masukan dari pengguna melalui keyboard. Pengguna diminta untuk memasukkan jumlah data yang ingin dimasukkan ke dalam **HashMap**. Kemudian, pengguna diminta untuk memasukkan kunci dan nilai untuk setiap entri. Setelah semua data dimasukkan, program mencetak seluruh entri dalam **HashMap**.

