Materi 2 : Memahami Struktur & Aturan Penulisan Sintaks, Percabangan, Perulangan, dan Method

DAFTAR ISI

1.1 Struktur Dasar Program Java

Struktur program java secara umum dibagi menjadi 4 bagian, yaitu deklarasi package, impor library, bagian class, dan method main.

```java
package com.pub_course; // <- 1. Deklarasi Package

import java.io.*; // <- 2. Impor Library

public class Program { // <- 3. Bagian Class
   public static void main(String[] args) { // <- 4. Method Main
       System.out.println("Hello World!");
   }
}
```

1.1.1 Deklarasi Package

Package merupakan sebuah folder yang berisi sekumpulan program Java. Deklarasi `package` biasanya dilakukan saat membuat program atau aplikasi besar. Contoh deklarasi package:

```java
package com.pub_course;
```

Biasanya nama `package` mengikuti nama domain dari sebuah vendor yang mengeluarkan program tersebut. Pada contoh di atas, `com.pub_course` adalah nama domain dari pub_course.com.

Aturan penulisan package adalah nama domain nya dibalik. Misalnya youtube.com dibalik menjadi `com.youtube`.

1.1.2 Impor Library

Pada bagian ini, kita melakukan impor library yang dibutuhkan pada program. Library merupakan sekumpulan class dan fungsi yang bisa kita gunakan dalam membuat program. Contoh impor library:

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

Pada contoh tersebut, kita mengimpor class `Scanner` dari package `java.util`.

1.1.3 Bagian Class

Java merupakan bahasa pemrograman yang menggunakan paradigma OOP (*Object Oriented Programming*). Setiap program harus dibungkus di dalam class agar nanti bisa dibuat menjadi objek. Berikut adalah contohnya:

```
public class Program {
  Statement......
}
```

Di dalam tanda kurung kurawal '`{}`' kita bisa mengisinya dengan berbagai method dan juga variabel.

1.1.4 Method Main

Method `main()` merupakan blok program yang akan dieksekusi pertama kali. Ini adalah entri point dari program. Method `main()` wajib kita buat. Kalau tidak, maka programnya tidak akan bisa dieksekusi.

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

1.1.5 Gaya Penulisan Case

Gaya penulisan case (case style) yang digunakan oleh Java adalah: camelCase, PascalCase, dan ALL UPPER.

1. camelCase, digunakan pada nama variabel, nama objek, dan nama method

```
   // contoh penulisan method
   void oddEvenMethod() {

       // contoh penulisan variabel
       int bilanganGanjil;
       int bilanganGenap;

       // contoh penulisan objek
       Program programGanjilGenap = new Program();
   }
```

2. PascalCase, digunakan pada penulisan nama class

```
class MahasiwaPub {
   ...
```

```
}
```

3. UPPER_CASE, digunakan untuk membuat varibel konstana atau `final`

```
final String FIRST_NAME = "Wahyu";
```

1.2 Percabangan

1.2.1 Percabangan IF ELSE

Penggunaan `if` `else` sama seperti bahasa C atau bahasa pemrograman lain. Berikut contoh sederhananya:

```
public static void main(String[] args) {
   int a = 10;
   if (a % 2 == 0) {
       System.out.println("Ini bilangan genap");
   }
   else {
       System.out.println("Ini bilangan ganjil");
   }
}
```

1.2.2 Percabangan Ternary Operator

Percabangan ini digunakan untuk mempersingkat baris kondisi pada sebuah program. Berikut contohnya:

```
public static void main(String[] args) {
    boolean suka = true;
    String jawaban = suka ? "iya" : "tidak";
    System.out.println(jawaban);
}
```

1.2.3 Percabangan Switch Case

```
public static void main(String[] args) {
   Scanner input = new Scanner(System.in);
   String lampu;
   System.out.print("Sekarang lampu apa : ");
   lampu = input.nextLine();
   switch (lampu) {
       case "merah":
           System.out.println("Berhenti!");
           break;
```

```
        case "kuning":
            System.out.println("Bersiap!");
            break;
        case "hijau":
            System.out.println("Jalan!");
            break;
        default:
            System.out.println("Inputan salah");
    }
}
```

1.3 Perulangan / Iterasi

Iterasi/perulangan dibagi menjadi dua, yaitu counted loop dan uncounted loop

1.3.1 Counted Loop

Arti dari *counted loop* adalah perulangan yang jumlah pengulangannya terhitung atau sudah di definisikan berapa kali program dapat berulang.

1. For

   Berikut adalah contoh program perulangan menggunakan `for`

```
public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
        System.out.println(i);
    }
}
```

2. For each

```
public static void main(String[] args) {
    String[] name = {"Dimas", "Wahyu", "Anggi", "Rey"};
    for (String n : name) {
        System.out.println(n);
    }
}
```

1.3.2 Uncounted Loop

Arti dari *uncounted loop* adalah perulangan yang jumlah pengulangannya tidak terhitung atau belum di definisikan kapan perulangan itu berhenti.

1. While

   While bisa kita artikan selama. Cara kerja perulangan ini seperti percabangan, ia akan melakukan perulangan selama kondisinya bernilai `true`. Contohnya adalah sebagai berikut:

```
public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      boolean ulang = true;
      int hitung = 0;
      String jawab;
      while (ulang) {
            System.out.print("Apakah ingin mengulang (y/t) : ");
            jawab = input.nextLine();
            if (jawab.equalsIgnoreCase("t")) {
                ulang = false;
            }
            hitung++;
      }
      System.out.println("Anda telah melakukan perulangan sebanyak "
      + hitung + " kali");
}
```

2. Do while

   Cara kerja perulangan Do/While sebenarnya sama seperti perulangan While. Bedanya, Do/While melakukan satu kali perulangan dulu. Kemudian mengecek kondisinya. Berikut contoh programnya:

```
public static void main(String[] args) {
   int i = 0;
   do {
       i++;
       System.out.println("Perulangan ke-" + i);
   } while (i < 10);
}
```

1.4 Method

Apa perbedaan procedure, function, dan method di bahasa pemrograman java?

- Procedure adalah sebutan untuk fungsi yang tidak mengembalikan nilai. Fungsi ini biasanya ditandai dengan kata kunci **`void`**.
- Function adalah sebutan untuk fungsi yang mengembalikan nilai.
- Method adalah fungsi yang berada dalam suatu class. Sebutan ini biasanya digunakan pada OOP.

1.4.1 Cara Membuat Method

Untuk membuat suatu method, kita harus menulisnya di dalam sebuah class. Struktur dasar sebuah method adalah sebagai berikut:

```
static TypeDataKembalian namaFungsi(){
   // statemen atau kode fungsi
}
```

Penjelasan:

- Kata kunci **`static`**, artinya kita membuat fungsi yang dapat dipanggil tanpa harus membuat instansiasi objek. ***Penjelasan di akhir modul***
- **`TypeDataKembalian`** adalah tipe data dari nilai yang dikembalikan setelah fungsi dieksekusi.
- **`namaFungsi()`** adalah nama fungsinya. Biasanya ditulis dengan huruf kecil di awalnya. Lalu, kalau terdapat lebih dari satu suku kata, huruf awal di kata kedua ditulis kapital (camelCase).

Berikut adalah contoh programnya:

```
static void sapaan(){
   System.out.println("Hello World");
}
```

1.4.2 Cara Memanggil Method

Setelah kita membuat fungsi, selanjutnya kita akan mengeksekusi fungsinya. Fungsi dapat dipanggil dari fungsi **`main`** atau dari fungsi yang lainnya. Contoh pemanggilan fungsi dalam dalam funsgi **`main`**:

```
static void sapaan() {
   System.out.println("Hello World");
}

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

1.4.3 Menggunakan Parameter

Parameter adalah variabel yang menampung nilai untuk diproses di dalam fungsi. Parameter berperan sebagai input untuk fungsi. Berikut adalah contoh programnya:

```
static void jumlah(int x, int y) {
   System.out.println(x + x);
}
```

```java
public static void main(String[] args) {
    jumlah(5, 5);
}
```

1.4.4 Method Yang Mengembalikkan Nilai

Setelah fungsi memproses data yang diinputkan melalui parameter, selanjutnya fungsi harus mengembalikan nilai agar dapat diolah pada proses berikutnya. Pengembalian nilai pada fungsi menggunakan kata kunci **`return`**. Berikut adalah contoh programnya:

```java
static int luasPersegi(int x) {
    return x * x;
}

public static void main(String[] args) {
    System.out.println("Luas persegi adalah : " + luasPersegi(5));
}
```

1.4.5 Static & Non-static

Pada contoh-contoh diatas, kita menggunakan kata kunci **`static`** sebelum membuat fungsi. Kata kunci **`static`** akan membuat fungsi dapat dieksekusi langsung, tanpa harus membuat instansiasi objek dari class.

```java
public class Program {

    // static function
    static void minum() {
        System.out.println("Saya sedang minum kopi");
    }
    // non-static function
    void makan(){
        System.out.println("Hi!");
        System.out.println("Saya sedang makan bubur");
    }

    public static void main(String[] args) {
        // menggunakan static
        minum();

        // tidak menggunakan static
        Program saya = new Program();
        saya.makan();
    }
}
```