



Modul Praktikum **PBO**



MODUL I DASAR - DASAR JAVA

1. STRUKTUR FILE

Struktur program Java secara umum dibagi menjadi 4 bagian:

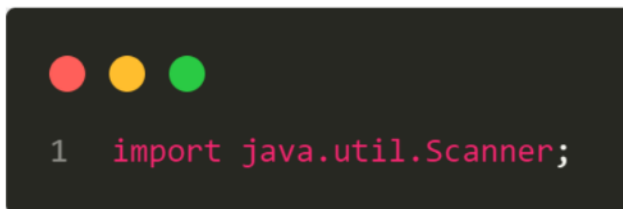
- **PACKAGE**

Package adalah sebuah cara untuk mengelompokkan class. Tujuannya menghindari bentrok nama class (jika ada yang bernama sama) serta memudahkan pengelolaan kode program, terutama untuk aplikasi besar.

Dalam prakteknya nanti, ini mirip seperti membuat folder ketika menyimpan file. Di setiap folder bisa saja terdapat file yang bernama sama, tapi karena disimpan dalam folder yang berbeda, itu tidak masalah. Begitu juga di package Java, kita bisa membuat nama class yang sama selama berada di dalam package yang berbeda. Package bahasa Java terbagi dalam 2 jenis:

- Built-in Package (package bawaan bahasa Java)

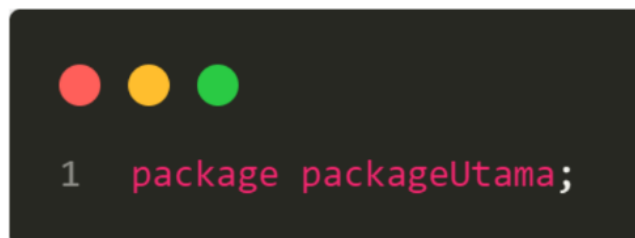
Java memiliki cukup banyak **package** bawaan dan beberapa ada yang sudah pernah kita pakai. Salah satunya package **java.util**, yang berisi **Scanner** class untuk proses input. Untuk menggunakan package, tambah perintah **import** sebelum nama package di awal kode program, seperti **import java.util.Scanner;**



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

- User-defined Package (package yang kita definisikan sendiri)

User-defined package adalah sebutan untuk package yang kita buat sendiri. Untuk membuat package, tambah perintah **package <nama_package>** di baris paling atas sebuah file Java. Dalam satu file, hanya bisa terdapat 1 perintah **package**. Contoh deklarasi package:



```
1 package packageUtama;
```



- IMPORT LIBRARY

Import Pada Java Merupakan Suatu Perintah Untuk Memasukan suatu Method atau perintah dalam Bahasa Pemrograman Java sehingga perintah tersebut dapat Aktif dan digunakan atau berfungsi. Contoh penerapan:

```
1 package packageUtama;
2 // untuk menampilkan output berbasis GUI
3 import javax.swing.JOptionPane;
4
5 public class Main {
6     public static void main(String[] args) throws Exception {
7         JOptionPane.showMessageDialog(null, "Hello World !!");
8     }
9 }
```

- Bagian Class (Akan ada di modul selanjutnya)

- METHOD MAIN

Setelah penulisan nama class, terdapat perintah **public static void main(String args[])**. Ini merupakan function atau method yang terdiri dari beberapa keyword yang sebenarnya agak kompleks jika kita bahas sekarang.

Untuk bisa memahaminya, harus menunggu sampai masuk ke materi tentang pemrograman object (OOP). Tapi sebagai gambaran dasar, saya akan jelaskan secara singkat:

- public: Berfungsi sebagai access modifier, yakni batasan akses dari sebuah kode program. Jika ditulis public, maka kode tersebut bisa diakses dari luar class. Nantinya ada beberapa access modifier lain seperti private dan protected.
- static: Menandakan bahwa ini adalah sebuah method yang bisa diakses langsung dari dalam class (tanpa harus membuat object).
- void: method ini tidak mengembalikan nilai.
- main: Ini merupakan nama dari method. Selain itu main adalah nama method khusus yang akan dibaca oleh Java compiler untuk memulai proses compile. Setiap aplikasi Java harus memiliki sebuah main method.
- (String args[]): Merupakan argument dari main method, yang dipakai untuk 'menangkap' sebuah nilai ketika kode program dijalankan dari cmd.

Jika anda tidak paham dengan penjelasan ini juga tidak masalah, karena seperti yang di singgung sebelumnya, kode ini seharusnya baru dibahas ketika masuk ke konsep OOP dari bahasa Java.



Untuk sementara, bisa dibilang bahwa perintah `public static void main(String args[])` harus ditulis di setiap file kode program Java. Dan di dalam method inilah kita menulis kode program utama.

Sama seperti class, tanda kurung kurawal juga dipakai untuk membatasi blok kode program:

```
1 package packageUtama;  
2 // untuk menampilkan output berbasis GUI  
3 import javax.swing.JOptionPane;  
4  
5 public class Main {  
6     // main method  
7     public static void main(String[] args) throws Exception {  
8         JOptionPane.showMessageDialog(null, "Hello World !!");  
9     }  
10 }
```

2. TIPE DATA

Java memiliki dua jenis tipe data yang dikategorikan menjadi dua yaitu tipe data primitif dan tipe data objek/referensi.

• TIPE DATA PRIMITIF

Java mempunyai 8 tipe dasar, yaitu boolean, char, byte, short, int, long, float, dan double.

Tipe Data	Default	Size	Range	Contoh
byte	0	8 bits	-128 .. 127	-12 ,12
short	0	16 bits	-32,768 .. 32,767	-16, 16
int	0	32 bits	-2,147,483,648 .. 2,147,483,647	-2, -1, 0, 1, 2
long	0l	64 bits	-9,223,372,036,854,775,808 9,223,372,036,854,775,807	-2L, -1L, 0L, 1L, 2L
float	0.0f	32 bits	$3.40282347 \times 10^{38}$, $1.40239846 \times 10^{-45}$	1.23e100f, - 1.23e-100f, .3f, 3.14F



double	0.0	64 bits	$1.7976931348623157 \times 10^{308}$, $4.9406564584124654 \times 10^{-324}$	1.23e300, 1.23e-300, .3, 3.14
char	'\u0000'	16 bits	\u0000 .. \uFFFF	'a', '\u0041', 67, 'B'
boolean	false	1 bit	true, false	true, false

- Integer
Integer merupakan tipe data numerik yang digunakan untuk mendefinisikan bilangan bulat. Tipe data numerik yang termasuk integer diantaranya: byte, short, int dan long.
- Floating/Double
Integer merupakan tipe data numerik yang digunakan untuk mendefinisikan bilangan bulat. Tipe data numerik yang termasuk integer diantaranya: byte, short, int dan long.
- Char
Char adalah karakter tunggal yang pendefinisiannya di awal dan akhir menggunakan tanda petik tunggal ('). Tipe char mengikuti aturan Unicode, sehingga bisa dapat menggunakan kode untuk kemudian diikuti bilangan dari 0 sampai 65535, tetapi yang biasa digunakan adalah bilangan heksadesimal dari 0000 sampai FFFF.
- Boolean
Tipe data Boolean terdiri dari dua nilai saja, yaitu true dan false. Boolean sangat penting untuk mengevaluasi suatu kondisi.

• TIPE DATA OBJEK

Kelebihan pemrograman dengan orientasi objek adalah dapat mendefinisikan tipe data baru yang merupakan objek dari class tertentu. Tipe data ini digunakan untuk mereferensikan objek atau class tertentu, seperti String.

- Variabel
Variabel merupakan container yang digunakan untuk menyimpan suatu nilai pada sebuah program dengan tipe tertentu. Untuk mendefinisikan variabel, suatu identifier dapat digunakan untuk menamai variabel tersebut.
- Identifier
Identifier adalah kumpulan karakter yang dapat digunakan untuk menamai variabel, method, class, interface, dan package. Dalam pemrograman Java identifier bisa disebut sah apabila diawali dengan:
 - Huruf /abjad



- Karakter Mata Uang
- Underscore (_)

Identifier dapat terdiri dari:

- Huruf / abjad
- Angka
- Underscore (_)

Identifier tidak boleh mengandung @, spasi atau diawali dengan angka serta tidak boleh menggunakan keyword yang telah digunakan di pemrograman java. Selain karakter, Unicode juga dapat digunakan sebagai identifier.

3. OPERATOR

Operator adalah simbol khusus yang menyajikan operasi khusus pada satu, dua, atau tiga operand dan kemudian mengembalikan hasilnya.

○ Operator Aritmatika

Operator ini digunakan pada operasi-operasi aritmatika seperti penjumlahan, pengurangan, pembagian dan lain-lain.

Operator	Keterangan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulus (sisanya)
++	Increment (menaikkan nilai dengan 1)
--	Decrement (menurunkan nilai dengan 1)

○ Operator Penugasan

Operator penugasan (Assignment Operator) fungsinya untuk memberikan suatu nilai ke sebuah variabel.

Operator	Nama Operator
=	Pengisian Nilai



+=	Pengisian dan Penambahan
-=	Pengisian dan Pengurangan
*=	Pengisian dan Perkalian
/=	Pengisian dan Pembagian
%=	Pengisian dan Sisa bagi

○ Operator Perbandingan

Untuk membandingkan 2 nilai (variabel) atau lebih dimana operator ini akan mengembalikan atau menghasilkan nilai True atau False.

Operator	Keterangan
==	Sama dengan
!=	Tidak sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan

○ Operator Logika

Operator ini menghasilkan nilai yang sama dengan operator perbandingan, hanya saja penggunaannya lebih pada operasi – operasi Boolean.

Operator	Keterangan
&&	Operasi AND
 	Operasi OR
^	Operasi XOR
!	Operasi NOT (Negasi)



- Operator Bitwise

Operator Bitwise merupakan operator yang digunakan untuk operasi bit (biner).

Operator bitwise terdiri dari:

Operator	Keterangan
&	AND
	OR
^	XOR
~	Negasi/Kebalikan
<<	Left Shift
>>	Right Shift
<<<	Left Shift (unsigned)
>>>	Right Shift (unsigned)

- Operator Ternary

Operator Ternary
kamu suka aku ? ya : tidak;
jawaban benar jawaban salah

Kondisi(?:) merupakan operator ternary, yang berarti bahwa operator ini mempunyai 3 argumen yang membentuk suatu kejadian/ekspresi. Operator ini seperti versi mini dari statement if-else. Struktur pernyataan Operator Kondisi(?:), seperti berikut ini.

ekspresi1?ekspresi2:ekspresi3;

Dimana ekspresi1 merupakan pernyataan boolean yang memiliki hasil salah satunya berupa true atau false. ekspresi2 akan di eksekusi jika nilai boolean pada ekspresi1 bernilai true dan ekspresi3 akan dijalankan jika nilai boolean pada ekspresi1 bernilai false.



4. PERCABANGAN

Percabangan hanyalah sebuah istilah yang digunakan untuk menyebut alur program yang bercabang. Percabangan juga dikenal dengan “Control Flow”, “Struktur Kondisi”, “Struktur IF”, “Decision”, dsb. Semuanya itu sama.

- IF

Bentuk If adalah yang paling sederhana, mengandung suatu pernyataan tunggal yang dieksekusi jika ekspresi bersyarat adalah benar. Sintaks dasar:

```
1  if(kondisi){
2      // lakukan sesuatu jika kondisi benar
3  }
```

- IF-Else

Untuk melakukan beberapa operasi yang berbeda jika salah satu ekspresi kondisional bernilai salah, maka digunakan statement else. Bentuk if-else memungkinkan dua alternatif operasi pemrosesan. Sintaks dasar:

```
1  if(kondisi){
2      // lakukan sesuatu kalau kondisi benar
3  }else{
4      // lakukan sesuatu kalau kondisi salah
5  }
```

- IF-Else-IF-Else

Bentuk if, else if, else memungkinkan untuk tiga atau lebih alternatif pemrosesan. Sintaks dasar:

```
1  if(kondisi_1){
2      // lakukan sesuatu jika kondisi_1 benar
3  }else if(kondisi_2){
4      // lakukan sesuatu jika kondisi_2 benar
5  }else{
6      // lakukan sesuatu jika kondisi salah
7  }
```



- Switch

Struktur pemilihan switch case sebetulnya hampir sama dengan percabangan IF ELSE IF dimana jika di percabangan IF terdapat beberapa kondisi, pada pemilihan switch akan ada beberapa case yang dapat kita buat, jika nilai yang menjadi nilai pembanding sama dengan case Struktur pemilihan switch case sebetulnya hampir sama dengan percabangan IF ELSE IF dimana jika di percabangan IF terdapat beberapa kondisi, pada pemilihan switch akan ada beberapa case yang dapat kita buat, jika nilai yang menjadi nilai pembanding sama dengan case. yang ada atau kata lainnya jika suatu case pada pemilihan switch bernilai true maka pernyataan pada case tersebut yang akan dieksekusi. Sintaks dasar:

```
1  switch(ekspresi){
2      case nilai1:
3          pernyataan1;
4          break;
5      case nilai2:
6          pernyataan2;
7          break;
8      default:
9          pernyataan_default;
10 }
```

5. INPUT

Input adalah sebuah proses memasukkan nilai/value kedalam program kita. Dalam Java memiliki beberapa input diantaranya seperti:

- Class Scanner

Sebelum Menggunakan Scanner kita wajib mengimport classnya terlebih dahulu, seperti berikut:

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

Lalu kita bisa menggunakannya seperti ini:



```
import java.util.Scanner;
public class Input{
    public static void main(String[] args) {
        String nama;
        int umur;

        Scanner inputScanner = new Scanner(System.in);
        System.out.print("Nama karyawan: ");
        nama = inputScanner.nextLine();
        System.out.print("Umur Karyawan: ");
        umur = inputScanner.nextInt();
        System.out.println("=====");
        System.out.println("Nama karyawan: " + nama);
        System.out.println("Umur Karyawan: " + umur);
    }
}
```

- Class Buffered Reader

Sama seperti scanner sebelum menggunakan inputnya kita harus mengimport classnya terlebih dahulu, seperti berikut:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

Dan dapat digunakan seperti ini:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Input{
    public static void main(String[] args) throws IOException {
        String nama;
        int umur;

        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);

        System.out.print("Nama Karyawan: ");
        nama = br.readLine();
        System.out.print("Umur Karyawan: ");
        umur = Integer.parseInt(br.readLine());

        System.out.println("=====");
        System.out.println("Nama Karyawan: " + nama);
        System.out.println("Umur Karyawan: " + umur);
    }
}
```



6. ARRAYLIST

ArrayList adalah sebuah kelas (class) pada Java yang memungkinkan kita dalam membuat sebuah objek untuk menampung data apapun yang bisa kalian gunakan untuk menampilkan daftar atau list nilai/value, yang bersifat dinamis dan juga dapat dimodifikasi, ukuran pada ArrayList dapat kita ubah dengan cara menambahkan (add) atau menghapusnya (remove).

Array	ArrayList
Ukurannya tidak dinamis	Ukurannya dinamis
Tidak mampu menyimpan data dengan tipe berbeda	Bisa menyimpan data dengan tipe berbeda
Bisa menyimpan data primitif	Tidak bisa menyimpan data primitif secara langsung, harus menggunakan <i>wrapper class</i>
Penyimpanan elemen baru harus menggunakan index	Bisa otomatis menyimpan elemen di akhir

Sebelum kita menggunakan ArrayList kita harus import dulu classnya seperti berikut:

```
1 import java.util.ArrayList;
```

Jika anda hanya ingin menyimpan satu tipe data anda juga bisa membuat objek ArrayList seperti ini:

```
4 ArrayList<String> arrayList = new ArrayList<>();
```

Berbagai operasi dapat Anda lakukan terhadap ArrayList seperti berikut:

- `size()`, untuk mencari panjang ArrayList
- `add(nilai)`, untuk menambah elemen baru
- `get(index)`, untuk mengambil elemen pada indeks tertentu
- `isEmpty()`, untuk memeriksa apakah ArrayList kosong atau tidak
- `indexOf(nilai)`, untuk mengetahui indeks dari suatu nilai
- `contains(nilai)`, untuk memeriksa apakah suatu nilai ada dalam ArrayList
- `set(index, nilai)`, untuk menimpa nilai pada indeks tertentu
- `remove(index/nilai)`, untuk menghapus elemen pada indeks tertentu
- `clear()`, untuk menghapus seluruh elemen di dalam ArrayList

Berikut adalah praktik penggunaan ArrayList:



```
1  import java.util.ArrayList;
2
3  public class Main {
4      public static void main(String[] args) {
5          ArrayList cars = new ArrayList<>();
6
7          cars.add(80);
8          cars.add(100);
9          cars.add("BMW");
10         cars.add("Ford");
11         cars.add("Daihatsu");
12
13         System.out.println("Panjang ArrayList: " + cars.size());
14         System.out.println("Isi ArrayList ");
15
16         for (int i = 0; i < cars.size(); i++) {
17             System.out.println(cars.get(i));
18         }
19
20         System.out.println("\nApakah ArrayList Kosong? " + cars.isEmpty());
21         System.out.println("Daihatsu ada di index: " + cars.indexOf("Daihatsu"));
22         System.out.println("Apakah Ada Toyota Di ArrayList? " + cars.contains("Toyota"));
23
24         cars.set(4, "Toyota");
25         cars.remove(0);
26         cars.remove("Ford");
27         System.out.println("\nSetelah Perubahan:");
28
29         for (Object car : cars) {
30             System.out.println(car);
31         }
32
33         cars.clear();
34         System.out.println("\nApakah ArrayList Kosong? " + cars.isEmpty());
35     }
36 }
```

Output Program:

```
Panjang ArrayList: 5
Isi ArrayList:
80
100
BMW
Ford
Daihatsu

Apakah ArrayList Kosong? false
Daihatsu ada di index: 4
Apakah Ada Toyota Di ArrayList? false

Setelah Perubahan:
100
BMW
Toyota

Apakah ArrayList Kosong? true
```



• PERULANGAN

Struktur perulangan adalah instruksi kode program yang bertujuan untuk mengulang beberapa baris perintah. Dalam bahasa Java, terdapat 3 buah struktur perulangan atau looping, yakni perulangan for, perulangan while dan perulangan do while.

- For

Dengan for statement kita bisa mengarahkan alur agar berulang sesuai dengan jumlah yang ditentukan. Sintaks Dasar:

```
1  for (inisialisasi; kondisi; increment atau decrement){
2      //kode program
3  }
```

Contoh:

```
3  for (int i = 1; i <= 5; i++) {
4      System.out.println("Perulangan ke - " + i);
5  }
```

- While

Dengan while statement kita bisa mengarahkan alur agar berulang selama kondisi pada while statement tersebut terpenuhi. Sintaks dasar:

```
1  while(kondisi){
2      // kode program
3  }
```

Contoh:

```
3  int x = 0;
4  while (x < 6) {
5      System.out.println("Perulangan ke - " + x);
6      x++;
7  }
```



- Do While

Do while sebenarnya hampir sama seperti while, perbedaannya adalah pernyataan do while akan mengecek kondisi di belakang, sementara while cek kondisi ada di depan. Sintaks Dasar:

```
1  do {  
2      // kode program  
3  }while(kondisi);
```

Contoh:

```
3  int x = 1;  
4  do {  
5      System.out.println("Perulangan ke - " + x);  
6      x++;  
7  } while (x <= 5);
```

- Break

Break merupakan perintah yang bisa digunakan untuk memberhentikan / keluar dari suatu perulangan walaupun kondisi atau jumlah perulangan belum selesai.

- Continue

Perintah continue mirip seperti perintah break, hanya saja jika dalam perintah break perulangan langsung berhenti, untuk perintah continue perulangan hanya melewati 1 kali proses iterasi saja.

- **METHOD/FUNGSI/PROSEDUR**

Dalam OOP, method digunakan untuk memodularisasi program melalui pemisahan tugas dalam suatu class. Pemanggilan method menspesifikasikan nama method dan menyediakan informasi (parameter) yang diperlukan untuk melaksanakan tugasnya. Sebenarnya Prosedur, Fungsi, dan Method itu sama.

- Prosedur adalah sebutan untuk fungsi yang tidak mengembalikan nilai. Fungsi ini biasanya ditandai dengan kata kunci void.
- Fungsi adalah sebutan untuk fungsi yang mengembalikan nilai.
- Method adalah fungsi yang berada di dalam Class. Sebutan ini, biasanya digunakan pada OOP.



Untuk memudahkan, mari kita sebut semuanya fungsi.

- Pembuatan Fungsi

Fungsi harus dibuat atau ditulis di dalam class. Struktur dasarnya seperti ini:

```
2 static TipeDataKembalian namaFungsi(TipeData parameter,...){
3     // kode fungsi
4 }
```

Kita gunakan keyword static agar fungsi bisa diakses langsung tanpa menginisialisasi class. Contoh fungsi tanpa nilai kembalian (Prosedur):

```
2 static void greetings(){
3     System.out.println("Hello!");
4 }
```

Contoh fungsi dengan nilai kembalian:

```
2 int tambah(int a, int b){
3     return a+b;
4 }
```

- Cara Memanggil/Mengeksekusi Fungsi

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

```
1 public class Account {
2     static int balance = 0;
3
4     static void deposit(int depositAmount) {
5         balance += depositAmount;
6         System.out.println("Rp. " + balance + " Deposited");
7     }
8
9     static int getBalance() {
10        return balance;
11    }
12
13    public static void main(String[] args) {
14        deposit(10000);
15        System.out.println(getBalance());
16    }
17 }
```

Hasil Program:

```
Rp. 10000 Deposited
10000
```




Jika fungsi tidak static maka pemanggilan fungsinya adalah seperti ini:

```
1  class Account {
2      int balance = 0;
3
4      void deposit(int depositAmount) {
5          balance += depositAmount;
6          System.out.println("Rp. " + balance + " Deposited");
7      }
8
9      int getBalance() {
10         return balance;
11     }
12
13     public static void main(String[] args) {
14         Account akun = new Account();
15         akun.deposit(1000);
16         System.out.println(akun.getBalance());
17     }
18 }
```