

## Sinau Java Desktop 1 - Fundamental [PERTEMUAN 1]

Java merupakan bahasa berorientasi object yang dibuat oleh SUN Microsystem (sebelum diakuisisi oleh Oracle). Java merupakan bahasa pemrograman yang lengkap, yang mampu membuat aplikasi mobile, desktop sampai web hanya dengan satu bahasa saja. Teknologi Java dibagi menjadi tiga bagian: JavaSE, JavaME (sekarang sudah jarang dipakai dan diganti dengan Android) dan JavaEE. Yang akan kita pelajari pada modul kali ini adalah JavaSE, yaitu dasar dari aplikasi Java yang dapat digunakan untuk membangun aplikasi console maupun desktop.

### Persiapan Persenjataan

Sebelum memulai coding dengan Java ada beberapa tools yang perlu kita siapkan.

1. Runtime dan Compiler; yang kita gunakan untuk meng-compile dan menjalankan aplikasi Java
2. IDE (Integrated Development Environment) atau tools yang memudahkan kita untuk membuat dan mengedit program Java yang akan kita buat

### Java SDK

Di Java runtime dan compiler ini dikenal dengan nama JDK (Java Development Kit). Didalam JDK ini terdapat compiler dan runtime yang biasa disebut dengan JRE (Java Runtime Environment). Kita bisa menginstall nya secara terpisah, jadi kalau hanya membutuhkan untuk menjalankan aplikasi Java saja cukup install JRE. Tapi pada modul kali ini kita membutuhkan JDK karena untuk development membutuhkan compiler tidak hanya runtime saja.

JDK bisa diunduh secara gratis [disitus resmi Oracle](#). Ada banyak pilihan download, unduh saja yang ada keterangannya JDK.

Setelah selesai mengunduh JDK bisa diinstall dikomputer yang akan kita pakai untuk memprogram Java. Proses instalasi bisa dilakukan seperti biasa, sama seperti menginstall aplikasi biasa pada umumnya. Namun bila anda menggunakan linux cukup ekstrak saja JDK yang telah diunduh.

Supaya JDK yang telah diinstall dikenali oleh OS yang kita pakai, maka kita harus melakukan setting di environment variable OS.

- JAVA\_HOME = {dir instalasi JDK}
- PATH = {dir instalasi JDK}/bin

Bila anda menggunakan linux (debian based) caranya agak sedikit berbeda, masuk ke console dan ketikkan perintah ini `sudo gedit /etc/bash.bashrc` enter. Tambahkan baris kode ini dibagian paling bawah file:

```
export JAVA_HOME={dir instalasi JDK}
export PATH=$PATH:$JAVA_HOME/bin
```

Untuk menguji apakah setting yang kita lakukan sudah benar, buka command prompt atau console, kemudian ketikkan perintah ini:

```
java -version
```

Enter, maka akan keluar output seperti ini:

```
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b15)
Java HotSpot(TM) Server VM (build 23.25-b01, mixed mode)
```

Apabila telah muncul output seperti diatas, maka konfigurasi JDK kita telah berhasil dikenali. Namun bila tidak muncul seperti diatas bisa dipastikan setting kita belum berhasil, biasanya karena direktori instalasi yang kita masukkan belum benar.

## IDE

Selanjutnya kita membutuhkan IDE. Ada banyak IDE yang bisa kita gunakan, yang gratis (Eclipse, Netbeans) maupun yang berbayar (IntelliJIDEA). Pada modul ini kita akan menggunakan Eclipse, karena lebih ringan, plugin banyak dan support komunitas yang besar.

Eclipse bisa diunduh secara gratis [disitus resminya](#). Ada banyak sekali varian dari Eclipse yang bisa kita gunakan, masing-masing memiliki fungsi dan tujuannya sendiri. Saran saya unduh Eclipse Standard saja, namun bila memiliki bandwidth yang sedikit melimpah bisa mengunduh Eclipse for JavaEE Developers (32 / 64 bit sesuaikan dengan OS yang kita pakai).

Setelah selesai diunduh, cukup ekstrak saja dan letakkan didirektori mana saja yang kita inginkan. Kemudian double-click file eclipse.exe atau eclipse (bila menggunakan linux). Ketika pertama kali dijalankan biasanya Eclipse akan meminta didirektori mana project yang akan kita buat disimpan. Dan Eclipse pun siap untuk digunakan.

## Command Line Programming

Pertama kita tidak akan menggunakan Eclipse terlebih dahulu. Ini sangat berguna untuk memahami bagaimana sesungguhnya Java meng-compile dan menjalankan program kita (no magic trick).

Semua program Java akan dicompile menjadi file bytecode dan dapat dijalankan dikomputer manapun yang sudah terinstall JDK / JVM, inilah yang membuat Java platform independent.

Native Apps	Non Native Apps
C, C++, VB, C#, Objective-C	Java, Scala, JRuby, Clojure
=====	=====



Buka aplikasi teks editor apa saja (notepad, textpad, notepad++, sublime, dll), ketikkan kode berikut dan simpan dengan nama HelloWorld.java. Ingat penulisan di Java case sensitif, artinya penulisan huruf besar dan kecil dianggap berbeda.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

Buka command prompt atau console, masuk ke direktori tempat menyimpan file tersebut, dan ketikkan perintah berikut.

```
javac HelloWorld.java
```

Ini adalah perintah untuk melakukan compile di java. Bila tidak ada error maka tidak akan muncul pesan apapun di command prompt. Setelah compile berhasil compiler akan membuatkan 1 file baru dengan nama HelloWorld.class. Ini adalah file bytecode hasil compilasi tadi. File inilah yang nantinya akan dijalankan oleh java runtime.

Untuk menjalankannya gunakan perintah berikut.

```
java HelloWorld
```

## Eclipse IDE

Eclipse merupakan tools gratis yang memudahkan kita dalam membuat aplikasi java. Banyak kemudahan yang akan kita dapatkan bila membuat program menggunakan IDE dari pada text editor biasa seperti notepad, seperti: code completion, compile dan running yang tinggal klik, code highlighting, debugging yang mudah, dll.

Untuk memulai pertama harus membuat project dulu. New Project >> Java Project. Kemudian buat class baru didalam project tersebut.

## Challenge

Buat program untuk menerima input berupa nama dan usia. Gunakan class Scanner untuk membaca input dari keyboard.

Solusi: lihat source code modul InputNamaDanUsia.java.

## Java Coding Standard

Ada beberapa aturan dasar yang sebaiknya kita ikuti untuk memudahkan pembuatan program. Ini bukanlah hal wajib yang ketika tidak diikuti akan mengakibatkan error diprogram yang kita buat, sama sekali bukan. Akan tetapi lebih kepada supaya baris program yang kita buat lebih mudah dipahami dan lebih mudah dibaca dan dimengerti.

Berikut adalah beberapa aturan yang sebaiknya kita ikuti dalam membuat aplikasi java:

- Nama class : awal kata selalu diawali dengan huruf kapital (biasa disebut PascalCase), seperti contoh berikut; IniNamaClass
- Nama method dan variable : paling depan diawali dengan huruf kecil kata selanjutnya menggunakan huruf kapital (biasa disebut camelCase), seperti contoh berikut; iniNamaMethod , iniNamaVariable
- Nama package : menggunakan huruf kecil semua, seperti contoh berikut; ininamapackage
- Nama variable absolute : varibale dengan tipe final (yang berarti nilainya tidak dapat diubah-ubah lagi) penamaannya menggunakan huruf kapital semua dan setiap pemisah antar kata digunakan tanda \_ (underscore), seperti contoh berikut; INI\_NAMA\_ABSOLUTE\_VARIABLE
- Comment / komentar di Java menggunakan syntax `//` untuk single line atau `/* */` untuk multiple line. Komentar digunakan untuk memberi keterangan pada program, dan semua yang ada didalam komentar akan diabaikan oleh compiler

Selain itu di Java ada beberapa nama yang tidak boleh dipakai sebagai nama variabel karena merupakan keyword dari bahasa Java, sebagai berikut:

<b>abstract</b>	<b>continue</b>	<b>for</b>	<b>new</b>	<b>switch</b>
<b>assert</b>	<b>default</b>	<b>if</b>	<b>package</b>	<b>synchronized</b>
<b>boolean</b>	<b>do</b>	<b>goto</b>	<b>private</b>	<b>this</b>
<b>break</b>	<b>double</b>	<b>implements</b>	<b>protected</b>	<b>throw</b>
<b>byte</b>	<b>else</b>	<b>import</b>	<b>public</b>	<b>throws</b>
<b>case</b>	<b>enum</b>	<b>instanceof</b>	<b>return</b>	<b>transient</b>
<b>catch</b>	<b>extends</b>	<b>int</b>	<b>short</b>	<b>try</b>
<b>char</b>	<b>final</b>	<b>interface</b>	<b>static</b>	<b>void</b>
<b>class</b>	<b>finally</b>	<b>long</b>	<b>strictfp</b>	<b>volatile</b>
<b>const</b>	<b>float</b>	<b>native</b>	<b>super</b>	<b>while</b>

## Tipe Data Java

Lihat source code CobaTipeData.java.

```
package javabasic.session1;

public class CobaTipeData {
    public static void main(String[] args) {
        int primitifInteger = 10;
        long primitifLong = 1178676550;
    }
}
```

```

double primitifDouble = 12d;
char primitifChar = 'c';
boolean primitifBoolean = true;

Integer objectInteger = 13;
Long objectLong = 14996088971;
Double objectDouble = 15d;
Character objectChar = 'c';
Boolean objectBoolean = false;
String objectString = "ini string";

System.out.println(primitifInteger);
System.out.println(primitifLong);
System.out.println(primitifDouble);
System.out.println(primitifChar);
System.out.println(primitifBoolean);

System.out.println(objectInteger);
System.out.println(objectLong);
System.out.println(objectDouble);
System.out.println(objectChar);
System.out.println(objectBoolean);
System.out.println(objectString);
}
}

```

Di Java ada tipe data primitif dan wrapper class. Pada tipe data primitif, value akan disimpan langsung ke dalam memori. Sedangkan pada wrapper class menyimpan referensi dari alamat yang sebenarnya.

## Operator

Operator adalah hal yang umum dalam bahasa pemrograman. Berdasarkan penggunaannya, kita dapat klasifikasikan operator menjadi:

- Aritmatika
- Relasional
- Logika
- Assignment

### Aritmatika

Operator ini digunakan untuk melakukan operasi matematika. Berikut adalah daftar operator aritmatika.

Operator	Kegunaan	Contoh
+	Penambahan	$x + y$
-	Pengurangan	$x - y$

*	Perkalian	$x * y$
/	Pembagian	$x / y$
%	Nilai sisa dari pembagian (modulus)	$x \% y$

### Challenge

Buat class untuk melakukan penjumlahan, pengurangan, perkalian, pembagian dan modulus untuk dua buah bilangan dan tampilkan hasilnya dilayar.

Solusi: lihat source code modul `OperatorAritmatika.java`.

### Relasional

Operator relasional membandingkan operand dan menentukan hasil perbandingan tersebut. Operator relasional dalam Java adalah sebagai berikut.

Operator	Contoh
>	$3 > 5 \rightarrow \text{false}$
<	$3 < 5 \rightarrow \text{true}$
>=	$3 \geq 5 \rightarrow \text{false}$
<=	$3 \leq 5 \rightarrow \text{true}$
!=	$3 \neq 5 \rightarrow \text{true}$
==	$3 == 5 \rightarrow \text{false}$

### Challenge

Buat class untuk membandingkan dua buah variabel menggunakan operator relasional.

Solusi: lihat source code modul `OperatorRelasional.java`.

### Logika

Operator logika berhubungan erat dengan nilai true atau false. Operator ini biasanya digunakan dalam blok if-else atau while. Berikut adalah operator kondisional yang tersedia di Java.

Operator	Nama	Contoh
&&	And	$\text{true} \ \&\& \ \text{false} \Rightarrow \text{false}$
	Or	$\text{true} \    \ \text{false} \Rightarrow \text{false}$
!	Not	$!\text{true} \Rightarrow \text{false}$

### Challenge

Buat class untuk membandingkan dua buah variabel menggunakan operator logika.

Solusi: lihat source code modul `OperatorLogika.java`.

## Assignment

NO	Operator	Deskripsi
1	=	Operator assignment sederhana, untuk meng-assign suatu value atau perintah ke dalam suatu variabel. Contoh: $C = A + B$ akan memberikan hasil penjumlahan dari $A + B$ ke dalam variable $C$
2	+=	Merupakan operator tambah dan assign. Contoh: $C += A$ sama dengan $C = C + A$
3	-=	Merupakan operator kurangi dan assign. Contoh: $C -= A$ sama dengan $C = C - A$
4	*=	Merupakan operator kalikan dan assign. Contoh: $C *= A$ sama dengan $C = C * A$
5	/=	Merupakan operator bagi dan assign. Contoh: $C /= A$ sama dengan $C = C / A$
6	%=	Merupakan operator modulus dan assign. Contoh: $C \% = A$ sama dengan $C = C \% A$
7	++	Merupakan operator unary increment. Contoh: $C++$ sama dengan $C = C + 1$
8	--	Merupakan operator unary decrement. Contoh: $C--$ sama dengan $C = C - 1$

## Challenge

Buat class untuk membandingkan dua buah variabel menggunakan operator assignment.

Solusi: lihat source code modul `OperatorAssignment.java`.

## Decisions

Didalam pemrograman decision atau aliran kontrol biasanya digunakan untuk menentukan jalannya program.

### If Dan If-Else

Percabangan **if** berguna untuk mengambil keputusan terhadap suatu kemungkinan.

```
if(kondisi) {  
    // perintah yang akan dijalankan  
}
```

Sedangkan percabangan **if else** merupakan percabangan yang sama dengan **if** tapi memiliki kondisi **false**, yang kalau kondisi **if** tidak terpenuhi maka akan menjalankan perintah yang ada didalam kondisi **else**.

```
if(kondisi) {  
    // perintah yang akan dijalankan didalam if  
} else {  
    // perintah yang akan dijalankan didalam else  
}
```

```
if(kondisi1) {  
    // perintah yang akan dijalankan didalam kondisi1  
} else if(kondisi2) {  
    // perintah yang akan dijalankan didalam kondisi2  
} else if(kondisi3) {  
    // perintah yang akan dijalankan didalam kondisi3  
} else {  
    // perintah yang akan dijalankan didalam else  
}
```

Lihat source code modul: CobaIfElse.java.

## Switch

Merupakan bentuk lain dari percabangan.

```
switch (month) {  
    case 1:  
        System.out.println("January");  
        break;  
    case 2:  
        System.out.println("February");  
        break;  
    case 3:  
        System.out.println("March");  
        break;  
    case 4:  
        System.out.println("April");  
        break;  
    case 5:  
        System.out.println("May");  
        break;  
    case 6:  
        System.out.println("June");  
        break;  
    case 7:  
        System.out.println("July");  
        break;  
    case 8:  
        System.out.println("August");  
        break;  
    case 9:  
        System.out.println("September");  
        break;  
    case 10:  
        System.out.println("October");  
        break;  
    case 11:  
        System.out.println("November");  
        break;  
}
```



```

    case 12:
        System.out.println("December");
        break;
    default:
        System.out.println("Invalid month.");
        break;
}

```

Perintah break digunakan untuk menghentikan perulangan. Break juga bisa dipakai untuk perulangan yang lain seperti for, while, dll.

Lihat source code modul: CobaSwitch.java.

## For

Merupakan perulangan yang akan menjalankan instruksi dengan rentang tertentu.

```

for (inisial value; batasan value; increment) {
    // jalankan perintah
}

```

Lihat source code modul: CobaFor.java.

## While

Merupakan sebuah perulangan yang akan menjalankan instruksi selama masih bernilai **true**.

```

while (true){
    // jalankan instruksi
}

```

Bentuk lain dari **while** adalah do-while.

```

do {
    // jalankan perintah
} while (expression);

```

Lihat source code modul: CobaWhile.java.

## Continue

Merupakan perintah untuk melewati perulangan tertentu.

Lihat source code modul: CobaContinue.java.

## Code Block

Code block merupakan pembatas dikenalnya suatu variabel. Ketika suatu variabel didefinisikan didalam blok tertentu maka tidak akan dikenali didalam blok yang lain.

Lihat source code modul CobaCodeBlock.java.

## Debugging

Debugging merupakan suatu teknik untuk mencari titik permasalahan didalam program. Biasanya debugging dilakukan ketika output dari program yang dibuat tidak sesuai dengan yang diharapkan.

## Tugas

Buat perangkingan nilai A-E, jika nilai lebih dari 80 = A, 70-80 = B, 50-60 = C, 30-50 = D, kurang dari 30 = E.

- Input 1: Nama Mahasiswa.
- Input 2: Nilai Mahasiswa.
- Output: Halo "Nama Mahasiswa", nilai anda adalah: "Nilai".