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 beberap 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 evironment 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 diuduh 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
==========	==========
	Apps
Apps	VM
OS	OS
Hardware	Hardware

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 sintax `//` untuk single line atau `/*  */` untuk multiple line. Komentar dugunakan unruk 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:

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

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 –> false
<	3 < 5 –> true
>=	3 >= 5 –> false
<=	3 <= 5 –> true
!=	3 != 5 –> true
==	3 == 5 –> 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	true && false => false
||	Or	true || false => false
!	Not	!true => 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 assigntment.

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

[PERTEMUAN 2]

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 dikenalinya 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".

[PERTEMUAN 3]

Class, Method dan Constructor

Class

Setiap kali kita akan membuat sebuah kode java, yang pertama harus dideklarasikan adalah class. Class di dalam java dideklarasikan menggunakan keyword class diikuti dengan nama class. Setelah nama class ada kurung kurawal buka ({) menandai awal dari class dan kurung kurawal tutup (}) yang menandai akhir dari class.

Object

Object adalah instansiasi dari class. Misal kita membuat sebuah class `public class Mahasiswa {}`, kemudian class tersebut kita instansiasi `Mahasiswa mhs = new Mahasiswa();`, maka Mahasiswa disini berperan sebagai class sedangkan mhs merupakan object.

Method

Method adalah sekumpulan kode yang diberi nama, untuk merujuk ke sekumpulan kode tersebut digunakan sebuah nama yang disebut dengan nama method. Method mempunyai parameter sebagai input dan nilai kembalian sebagai output.

Constructor

Constructor adalah method yang spesial, karena mempunyai aturan-aturan sebagai berikut:

- mempunyai nama yang sama persis dengan nama class
- tidak mempunyai tipe return

- digunakan untuk menginstansiasi object
- hanya mempunyai access modifer, tidak ada keyword lain yang diletakkan sebelum nama method pada deklarasi constructor.

```
public class Mahasiswa {
    public Mahasiswa() {
        System.out.println("Selamat datang!");
    }
    public Mahasiswa(String nama) {
        System.out.println("Halo "+nama+" selamat datang!");
    }
    public void kirimPesan(String pesan) {
        System.out.println("Pesan anda adalah "+pesan);
    }
    public Boolean hitungNilai(Integer nilai) {
        return nilai > 60 ? true : false;
    }
}
```

Challenge

Buat class main untuk menginstansiasi class Mahasiswa diatas dan panggil methodnya.

Abstract Class dan Interface

Abstract class merupakan sebuah class yang memiliki method abstract. Method abstract adalah method yang tidak memiliki definisi, dan pendefinisiannya diserahkan ke class turunannya. Sedangkan interface merupakan class abstract yang mana semua method adalah abstract dan semua variable nya akan didefinisikan sebagai static dan final (konstanta).

Sedangkan interface merupakan class yang semua methodnya bersifat abstract dan harus diimplementasikan di class yang mengimplementasikan interface tersebut, dan semua variable didalam interface akan dideklarasikan sebagai `public static final` atau dengan kata lain sebagai sebuah konstanta.

Konstanta

Konstanta dalam java juga mempunyai aturan penamaan yang diterangkan dalam Java Code Convention. Nama konstanta semuanya huruf besar dan dipisahkan dengan underscore (_) kalau terdiri dari dua kata atau lebih.

```
final Integer INI_ADALAH_KONSTANTA_INTEGER = 10;
final String INI_ADALAH_KONSTANTA_STRING = "Hello";
```

OOP

Java merupakan bahasa pemrograman berorientasi object. Ciri khas dari OOP adalah memiliki sifat-sifat atau karakter sebagai berikut: inheritance atau pewarisan, overloading, overriding, enkapsulasi dan polimorfisme.

Inheritance

Inheritance atau pewarisan adalah kemampuan untuk mewariskan method atau variable ke dalam class anak.

Overloading

Overloading adalah method dalam satu class yang sama memiliki nama yang sama namun memiliki parameter yang berbeda.

Overriding

Overriding adalah mendefinisikan ulang method dari class induk.

Enkapsulasi

Enkapsulasi atau information hiding adalah menyembunyikan method atau variable tertentu dari class lain, biasanya dengan memberi akses modifier `private`.

Polimorfisme

Polimorfisme adalah class abstract atau interface yang dapat diinstansiasi dengan berbagai macam class turunannya.

Access Modifier

Public, protected, default dan private adalah empat buah level access modifer, fungsi dari access modifer adalah mengatur bagaimana bagian-bagian kode java diakses dari bagian yang lain.

```
public namaVariable
public namaMethod()
protected namaVariable
protected namaMethod()
namaVariable //ini variabel dengan access modifier default
namaMethod() //ini method dengan access modifier default
private namaVariable
private namaMethod()
```

Access modifer public menandakan bisa diakses oleh siapapun tanpa batasan. Access modifer protected bisa diakses oleh class turunanya dan class-class lain yang berada dalam package yang sama. Access modifer default tidak memerlukan keyword, kalau tidak ada salah satu dari tiga access modifer lain maka yang digunakan adalah access modifer default.

Kalau access modifer default digunakan, maka hanya class dari package yang sama saja yang bisa mengakses, termasuk class itu sendiri. Yang terakhir adalah access modifer private yang hanya mengijinkan diakses oleh class yang sama.

Lihat source code modul tentang OOP didalam package `javabasic.session3`.

Exception

Exception sangat penting di Java, digunakan untuk menangani kesalahan, misal karena kesalahan input, ada pembagian dengan 0 atau konversi tipe data yang salah. Exception sangat berguna apabila muncul kesalahan didalam aplikasi, maka tidak akan langsung keluar atau close begitu saja.

```
try {
    //kode yang ada exception
} catch (ExceptionPertama ex){
    //handle exception dengan tipe ExceptionPertama
} catch (ExceptionKedua ex){
    //handle exception dengan tipe ExceptionKedua
} finally{
    //bersihkan resource yang dipakai, baik terjadi exception ataupun tidak
}
```

Challenge

Buat interface BangunDatar yang didalamnya memiliki method hitungLuas dengan dua buah parameter double, buat tiga buah class beri nama Segitiga, Persegi dan PersegiPanjang masing-masing dari class tersebut implement dari BangunDatar. Buat class main didalamnya terdapat pilihan untuk menghitung luas dari masing-masing bangun datar tersebut.

[PERTEMUAN 4]

JDBC

Untuk mengakses database dari Java, kita memerlukan library tambahan, atau biasa disebut sebagai JDBC driver. Driver yang akan digunakan harus disesuaikan dengan database yang digunakan. Sebagai contoh bila kita menggunakan database mysql maka driver yang digunakan adalah driver dari database mysql.

Untuk mendapatkan driver bisa dicari di repository maven seperti berikut: http://mvnrepository.com/artifact/mysql/mysql-connector-java.

Pada sesi ini kita akan mencoba membuat database menggunakan mysql dan akan kita akses dari Java.

Lihat source code modul dalam package `javabasic.session4`.

[PERTEMUAN 5]

Package dan Import

Pakcage dalam java adalah sebuah mekanisme untuk mengorganisasi penamaan class ke dalam modul-modul. Class yang mempunyai fungsionalitas serupa dan kemiripan cukup tinggi biasanya diletakkan dalam satu package yang sama.

Import digunakan untuk menyederhanakan penulisan class. Tanpa menggunakan import kita harus menuliskan nama lengkap class besarta packagenya. Dengan menggunakan import, kita deklarasikan di mana class yang digunakan tersebut berada sehingga selanjutnya tidak perlu lagi menuliskan nama package dari sebuah class. Ada dua pengecualian di mana import tidak diperlukan, pertama untuk class-class yang berada dalam package yang sama dan kedua adalah class-class yang berada dalam package java.lang.

Date, Calendar, DateFormat

Class Date adalah representasi dari waktu di Java, class ini terdapat dua jenis: java.util.Date dan java.sql.Date. Class java.util.Date adalah class yang sering digunakan dalam perhitungan tanggal, sedangkan java.sql.Date adalah representasi data tanggal di java yang setara dengan tipe data date di dalam database. Kalau tidak bekerja dengan database, sebaiknya selalu gunakan java.util.Date, sedangkan java.sql.Date hanya digunakan ketika kita bekerja dengan database. Istilah class Date di dalam bab ini merujuk ke java.util.Date.

Joda Time

Joda Time merupakan library tambahan yang sangat mempermudah dalam melakukan perhitungan yang berhubungan dengan date. Joda Time memperkenalkan enam konsep waktu, yaitu :

- Instant : sebuah instant adalah nilai dalam mili detik dari 1 januari 1970 00:00:00 hingga sekarang.
- Parsial : representasi sebagian dari waktu, misalnya tanggal saja atau waktu saja. (jarang digunakan, lebih baik menggunakan DateTime)
- Interval : interval waktu dari awal hingga akhir. Dalam interval terdapat dua buah instant yaitu awal dan akhir.
- Durasi : durasi waktu dalam mili detik, misalnya 1000 mili detik.
- Periode : periode waktu yang direpresentasikan dalam satuan-satuan waktu, misalnya 1 hari 2 jam 3 menit dan 30 detik.
- Chronology : sistem penanggalan yang bisa digunakan sebagai basis perhitungan waktu.

Joda Time dapat diunduh disitus berikut: http://www.joda.org/joda-time/.

Lihat source code modul dalam package `javabasic.session5`.