**Remedial Pemrograman Berorientasi Objek**

**Rangkuman Materi Semester Dua**

**M.Urffan Fadillah**

**XI – RPL 2**

**Interface**

Interface adalah jenis khusus dari blok yang hanya berisi method signature atau constant. Interface mendefinisikan sebuah signature dari sebuah kumpulan method tanpa tubuh. Interface mendefinisikan sebuah cara standar dan umum dalam menetapkan sifat-sifat dari class-class. Mereka menyediakan class-class tanpa memperhatikan lokasinya dalam hirarki class untuk mengimplementasikan sifat-sifat yang umum.

**Contoh pendeklarasian Interface**

public interface InterfaceA{

String atributA = “Ini konstanta dari interface A”;

void methodSatuA();

String methodSatuA();

}

**Contoh Program Interface**

Perhatikan kode interface CetakLaporan:

public interface CetakLaporan {

    public void cetakA4();

    public void cetakA3();

}

Interface CetakLaporan diimplementasikan pada class LaporanTahunan

public class LaporanTahunan implements CetakLaporan{

    public void cetakA4(){

        System.out.println("Cetak Laporan di A4");

    }

    public void cetakA3(){

        System.out.println("Cetak Laporan di A3");

    }

    public static void main(String[] args) {

        // TODO Auto-generated method stub

        LaporanTahunan lt = new LaporanTahunan();

        lt.cetakA3();

        lt.cetakA4();

    }

}

**Multiple Inheritance**

public interface TampilLaporan {

    public void tampilWeb();

    public void tampilMobile();

}

public class LaporanTahunan implements CetakLaporan, TampilLaporan{

    public void cetakA4(){

        System.out.println("Cetak Laporan di A4");

    }

    public void cetakA3(){

        System.out.println("Cetak Laporan di A3");

    }

    public void tampilWeb(){

        System.out.println("Tampil di Monitor");

    }

    public void tampilMobile(){

        System.out.println("Tampil di Handphone");

    }

    public static void main(String[] args) {

        // TODO Auto-generated method stub

        LaporanTahunan lt = new LaporanTahunan();

        lt.cetakA3();

        lt.cetakA4();

        lt.tampilMobile();

        lt.tampilWeb();

    }

}

**Class built-in (math)**

1) Class Math Java juga menyediakan konstanta dan method untuk menunjukkan perbedaan operasi matematika seperti fungsi trigonometri dan logaritma. Selama methodmethod ini semua static, kita dapat menggunakannya tanpa memerlukan sebuah objek Math.

2) Public Static double abs (double a) Menghasilkan nilai mutlak. Sebuah method yang dioverload dapat juga menggunakan nilai float atau integer atau juga long integer sebagai parameter, dengan kondisi tipe kembaliannya juga menggunakan float atau integer atau long.

3) Public Static double random() Method ini digunakan untuk membangkitkan suatu nilai double acak dengan rentang lebih besar atau sama dengan nol (0) dan lebih rendah dari 1 (0 <=Math.random() < 1.0). Method ini digunakan untuk menuliskan suatu ekspresi sederhana untuk membangkitkan angka-angka acak dengan sembarang rentang.

4) Public Static double max (double a, double b) Menghasilkan nilai maksimum, diantara dua nilai double, a dan b. Sebuah method yang di-overload dapat juga menggunakan nilai float atau integer atau juga long integer sebagai parameter, dengan kondisi tipe kembalinya juga menggunakan float atau integer atau longinteger, secara berturut-turut.

5) Public Static double min (double a, double b) Menghasilkan nilai minimum diantara dua nilai double ,a dan b. Sebuah method yang di-overload dapat juga menggunakan nilai float atau integer atau juga longinteger sebagai parameter, dengan kondisi tipe kembaliannya juga menggunakan float atau integer atau longinteger, secara berturut-turut.

6) Public Static double log (double a) Menghasilkan logaritma natural dari a. Mengembalikan logaritma natural (base e) dari nilai ganda. Kasus khusus : Jika argumen adalah NAN atau kurang dari nol, maka hasilnya adalah NAN. Jika argumen adalah infinity positif, maka hasilnya adalah tak terhingga positif. Jika argumen positif nol atau negatif nol, maka hasilnya adalah tak terhingga negatif. Hasilnya dihitung harus berada dalam 1 ulp hasil yang tepat. Hasil harus semi-monoton. Parameter : a – nilai. Pengembalian: nilai ln a, logaritma natural dari.

**Class Built-in (String dan Wrapper)**

1) Class String

Class String disediakan oleh Java SDK dengan menggunakan kombinasi character literals. Tidak seperti bahasa pemrograman lainnya, seperti C atau C++, strings dapat digunakan menggunakan array dari character atau disederhanakan dengan menggunakan class String. Sebagai catatan, bahwa sebuah objek String berbeda dari sebuah array dari character

2) Class StringBuilder/StringBuffer

Ketika objek String diciptakan, objek String tidak bisa lagi dimodifikasi. Objek StringBuffer serupa dengan objek String, kecuali kenyataan bahwa objek StringBuffer bersifat dapat berubah atau dapat dimodifikasi, sedangkan pada object String bersifat konstan. Dapat dikatakan bahwa class StringBuffer ini lebih fleksibel dibanding class String

3) Class Wrapper

tipe data primitive seperti int, char and long bukanlah sebuah objek. Sehingga, variabel-variabel tipe data ini tidak dapat mengakses methodmethod dari classObject. Hanya objek-objek nyata, yang dideklarasikan menjadi referensi tipe data, dapat mengakses method-method dari classObject.

**Exception**

Exception adalah sebuah event yang menjalankan alur proses normal pada program. Event ini biasanya berupa kesalahan error dari beberapa bentuk, disebabkan program kita berakhir tidak normal. Ketika terjadi kesalahan, otomatis akan dilemparkan sebuah objek yang disebut exception. Kelas-kelas exception yang digunakan dalam java adalah ArithmeticException, FileNotFoundException, InputMisMatchException, NullPointerException, ClassNotFoundException, IOException, RunTimeException, dan masih banyak lagi lainnya. Kelas-kelas exception dapat diklarifikasikan menjadi 3 tipe utama yaitu error system, exception, dan exception runtime. Contoh sub kelas error adalah LinkageError, dan VirtualMachineError. Contoh sub kelas exception adalah ClassNotFoundException dan IOException. Dan contoh sub kelas RunTime.

**Exception Handling**

Java memberikan akses kepada developer untuk mengambil object bug yang terjadi dengan mekanisme yang dikenal dengan ExceptionHandling. ExceptionHandling merupakan fasilitas di java yang memberikan flexibiitas kepada developer untuk menangkap bug atau kesalahan yang terjadi ketika program berjalan. Dalam exception Handling ada 5 keyword penting yang digunakan, yaitu Try yang digunakan untuk menjalankan block program kemudian mengenai dimana kesalahan yang akan diproses. Catch digunakan berpasangan dengan keyword try, keyword ini digunakan untuk menangkap kesalahan yang terjadi dalam block try. Finally digunakan untuk menunjukkan bahwa block program akan selalu dieksekusi meskipun adanya kesalahan yang muncul ataupun tidak ada. Keyword throw digunakan untuk melemparkan bug yang dibuat secara manual, keyword ini digunakan dalam satu method atau kelas yang menghasilkan kesalahan sehingga perlu ditangkap errornya.

**Contoh Program Exception Handling**

1. ArithmeticException

|  |
| --- |
| public class cobaException  {  public static void main(String args[])  {  int bil = 10;  System.out.println(bil/0);  }  } |

Program diatas akan memunculkan outpur error sebagai berikut :

|  |
| --- |
| Exception in thread "main" java.lang.ArithmaticException: / by zero  At daspro.contoh\_try.main(contoh\_try.java:18)  Java Result : 1 |

Disana muncul adanya kesalahan aritmetika, pesan kesalahan tersebut tampil karena adanya proses perhitungan yang salah. karena di java secara otomatis akan menangkap exception ketika ada kode program yang salah.

**Thread**

Thread adalah rangkaian eksekusi dari sebuah aplikasi java dan setiap program java minimal memiliki satu buah thread. Sebuah thread bisa berada di salah satu dari 4 status, yaitu new, runnable, blocked, dan dead.

Status Thread

• New, Thread yang berada di status ini adalah objek dari kelas Thread yang baru dibuat, yaitu saat instansiasi objek dengan statement new. Saat thread berada di status new,belum ada sumber daya yang dialokasikan, sehingga thread belum bisa menjalankan perintah apapun.

• Runnable. Agar thread bisa menjalankan tugasnya, method start() dari kelas Thread harus dipanggil. Ada dua hal yang terjadi saat pemanggilan method start(), yaitu alokasi memori untuk thread yang dibuat dan pemanggilan method run(). Saat method run() dipanggil, status thread berubah menjadi runnable, artinya thread tersebut sudah memenuhi syarat untuk dijalankan oleh JVM. Thread yang sedang berjalan juga berada di status runnable.

• Blocked. Sebuah thread dikatakan berstatus blocked atau terhalang jika terjadi blocking statement, misalnya pemanggilan method sleep(). sleep() adalah suatu method yang menerima argumen bertipe integer dalam bentuk milisekon. Argumen tersebut menunjukkan seberapa lama thread akan “tidur”. Selain sleep(), dulunya dikenal method suspend(), tetapi sudah disarankan untuk tidak digunakan lagi karena mengakibatkan terjadinya deadlock. Thread akan menjadi runnable kembali jika interval method sleep()-nya sudah berakhir, atau pemanggilan method resume() jika untuk menghalangi thread tadi digunakan method suspend()

• Dead. Sebuah thread berada di status dead bila telah keluar dari method run(). Hal ini bisa terjadi karena thread tersebut memang telah menyelesaikan pekerjaannya di method run(), maupun karena adanya pembatalan thread. Status jelas dari sebuah thread tidak dapat diketahui, tetapi method isAlive() mengembalikan nilai boolean untuk mengetahui apakah thread tersebut dead atau tidak.

Pembentukan Thread dalam java

Untuk membuat thread dalam java terdapat dua cara ( sejauh yang saya tahu 😀 )

• Extends class Thread

Untuk menjalankan thread, dapat dilakukan dengan memanggil method start(). Saat start() dijalankan, maka sebenarnya method run() dari class akan dijalankan. Jadi untuk membuat thread, harus mendefinisikan method run() pada definisi class. Konstruktor dari cara ini adalah :

SubThread namaObject = new SubThread();

namaObject.start();

• Implements interface Runnable

Cara ini merupakan cara yang paling sederhana dalam membuat thread. Runnable merupakan unit abstrak, yaitu kelas yang mengimplementasikan interface ini hanya cukup mengimplementasikan fungsi run(). Dalam mengimplementasi fungsi run(), kita akan mendefinisikan instruksi yang membangun sebuah thread. Konstruktor dari cara ini adalah :

MyThread myObject= new MyThread();

Thread namaObject = new Thread(myObject);

Atau dengan cara singkat seperti :

New Thread(new ObjekRunnable());

MyThread merupakan class yang mengimplementasikan interface dari Runnable, dan object dari class MyThread akan menjadi parameter instansi object class Thread.

**Contoh Program Thread**

Pada contoh yang pertama ini, kita akan membuat program sederhana yang menampilkan output setiap 1 detik sekali menggunakan *thread*dan *for loops*, sebelumnya kita sudah belajar mengenai penggunaan for loop pada java, jadi disarankan kalian sudah memahaminya.

public class latihan\_thread {

public static void main(String[] args) {

int jumlah = 10;

Thread thread = new Thread(){

public void run(){

try{

for(int w=1; w<=jumlah; w++){

System.out.println("Nomor: "+w);

sleep(1000); //Waktu Pending

}

}catch(InterruptedException ex){

ex.printStackTrace();

}

}

};

thread.start();

}

}

Jika kita jalankan program tersebut akan melooping dan mengeluarkan output sebanyak 10 kali, program akan mengeluarkan output 1 kali per detiknya, itulah yang membedakan program yang menggunakan thread dan yang tidak, jika kita tidak menggunakan thread atau hanya for loop saja, maka program akan mengeluarkan semua output tersebut sekaligus, **sleep(1000)**artinya kita mengset waktu pending selama 1000 millisecond/1detik, *try* dan *catch*digunakan untuk mengkap jenis error *InterruptedException*agar program tidak crash saat error itu terjadi.  
  
InterruptedException terjadi karena thread berhenti sementara, jadi kalian bisa melakukan apa saja jika thread berhenti sementara.

**Package**

Pengertian Package

Package adalah sarana/cara pengelompokkan dan pengorganisasian kelas-kelas dan interface yang sekelompok menjadi suatu unit tunggal dalam library.

Package juga mempengaruhi mekanisme hak akses ke kelas-kelas di dalamnya.

**Pengaruh Package terhadap Method main()**

Kelas yang mengandung method main() memilki syarat tidak berada dalam suatu package, dan hirarki posisi foldernya di atas package yang diimport.

**Membuat Package**

Ada tiga langkah untuk membuat package :

1. Mendeklarasikan dan memberi nama package.
2. Membuat struktur dan nama direktori yang sesuai dengan struktur dan nama package.
3. Mengkompilasi kelas-kelas sesuai dengan packagenya masing-masing.

**Mendeklarasikan dan Memberi Nama Package**

Deklarasi package harus diletakkan pada bagian paling awal (sebelum deklarasi import) dari source code setiap kelas yang dibungkus package tersebut.

Bentuk umum deklarasi package :

*package namaPackage;*

Deklarasi tersebut akan memberitahukan kompilator, ke library manakah suatu kelas dikompilasi dan dirujuk.

Syarat nama package :

1. Diawali huruf kecil
2. Menggambarkan kelas-kelas yang dibungkusnya
3. Harus unik (berbeda dengan nama package standard)
4. Merepresentasikan path dari package tersebut.
5. Harus sama dengan nama direktorinya.

Contoh package standard :

*java.lang* (berisi kelas-kelas fundamental yang sering digunakan).

*java.awt* dan *javax.swing* (berisi kelas-kelas untuk membangun aplikasi GUI)

*java.io* (berisi kelas-kelas untuk proses input output)

**Membuat Struktur Direktori**

Pada langkah ini, buatlah direktori menggunakan file manager (di windows menggunakan explorer) sesuai struktur package dari langkah sebelumnya. Kemudian tempatkan kelas-kelas tersebut ke direktori yang bersesuaian (mirip seperti  menyimpan file-file ke dalam folder).

Package dapat bersarang di package lain, sehingga dapat dibuat hirarki package.

Bentuk umum pernyataan package multilevel :

*package namaPackage1[.namaPackage2[.namaPackage3]];*

Contoh hirarki package di JDK :

*package java.awt.image;*

Compile dan Run Kelas dari suatu Package

Selanjutnya masing-masing kelas tersebut dalam package tersebut dikompilasi menjadi byte code (\*.class). Artinya package tersebut siap digunakan.

**Menggunakan Package**

Ada dua cara menggunakan suatu package yaitu :

1. Kelas yang menggunakan berada dalam direktori (package) yang sama dengan kelas-kelas yang digunakan. Maka tidak diperlukan import.
2. Kelas yang menggunakan berada dalam direktori (package) yang berbeda dengan kelas-kelas yang digunakan. Maka pada awal source code di kelas pengguna harus mencantumkan :

*import namaPackage.NamaKelas;*atau

*import namaPackage.\*;*

Contoh :

*import java.text.DecimalFormat;*

*import javax.swing.\*;*

**Setting Classpath**

Path hirarki package, didaftarkan sebagai salah satu nilai variabel lingkungan yang bernama Classpath.

Classpath diset dengan aturan : berawal dari drive (C:\ atau D:\) sampai dengan satu tingkat sebelum kita mendeklarasikan package.

**Contoh Program Package**

package ContohPkg;

public class Lingkaran

{

private final double PI = 3.14;

private double r;

public Lingkaran(double r)

{

this.r = r;

}

public double luas()

{

return (PI\*r\*r);

}

public double keliling()

{

return (2\*PI\*r);

}

}

Setelah itu buat kembali  file baru dengan nama  Segitiga.javadan simpan ke dalam direktori  
ContohPkg.  Selanjutnya isikan kode program seperti berikut :  
  
  
package ContohPkg;  
public class Segitiga  
{  
private double tinggi;  
private double alas;  
public Segitiga(double tinggi, double alas)  
{  
this.tinggi = tinggi;  
this.alas = alas;  
}  
public double luas()  
{  
return ((tinggi\*alas)/2);  
}  
}  
  
  
  
Selanjutnya lakukan kompilasi terhadap kedua file di atas dengan menggunakan perintah  
Lingkaran.java dan SEgitiga.java dengan cara seperti ini akan tercipta fileLingkaran.class dan  
Segitiga.class di dalam direktori ContohPkg. Setelah selesei, lakukan pengimportan paket dengan  
cara :  
Pertama buat file baru dengan nama PanggilPaket.java simpan di direktori bin j2sdk1.4.1.  
selanjutnya, isi kode sebagai berikut :

import ContohPkg.\*;  
class PanggilPaket  
{  
public static void main(String[] args)  
{  
Lingkaran obLingkaran = new Lingkaran(7);  
Segitiga obSegitiga = new Segitiga(4, 3);  
System.out.println("Luas Lingkaran : "+obLingkaran.luas());  
System.out.println("Keliling Segitiga : "+obLingkaran.keliling());  
System.out.println("Luas Segitiga : "+obSegitiga.luas());  
}  
}