**POKOK BAHASAN 5**

**EXCEPTION (EKSEPSI)**

**PENDAHULUAN**

Setelah mempelajari bab ini diharapkan mahasiswa mampu :

1. Memahami perbedaan antara kode program java yang memanfaatkan Exception Handling dan yang tidak memanfatkan.
2. Mahasiswa mampu mengimplementasikan kata kunci try, catch, dan throw dan throws.

**PENYAJIAN (TUTORIAL)**

Eksepsi adalah mekanisme yang memungkinkan program untuk menangkap kondisi-kondisi tertentu yang membutuhkan penanganan khusus. Eksepsi dapat berupa kesalahan yang terjadi dalam program pada saat run time. Format eksepsi adalah :

Try {

// blok program

. . . . . .

} catch (EksepsiTipe1 e) {

// blok program untuk EksepsiTipe1

. . . . . .

} catch (EksepsiTipe2 e) {

// blok program untuk EksepsiTipe2

. . . . . .

} catch (EksepsiTipe3 e) {

// blok program untuk EksepsiTipe3

. . . . . .

} finally {

// blok program

. . . . . . }

**try** menyatakan bahwa dalam blok try dapat terjadi suatu ekspresi dan bila ekspresi terjadi, maka aktifkan program yang berada dalam kelompok (blok) **catch** sesuai dengan tipe ekspresinya.

Object **e** dari tipe ekspresi dapat digunakan sebagai referensi penanganan eksepsi aktual.

finally menyatakan bahwa apapun jenis tipe ekspresinya, maka program dalam blok ini harus tetap dijalankan.

Dengan menangkap eksepsi program tidak langsung berhenti dan melaporkan kesalahan tersebut, melainkan membelokkan dengan program pengaman tertentu.

Pada saat run time, jika java menemukan kesalahan dan melempar pesan ke program. Dengan pesan ini, program akan melakukan suatu aksi tertentu. Mekanisme melempar ini disebut juga sebagai **throw**, alur program akan dihentikan, kemudian program pengendali (exception handling) akan diaktifkan.

**throw**

Throw digunakan secara eksplisit untuk membuang sebuah eksepsi. Contoh berikutnya, TestThrow mendefinisikan Obyek Eksepsi “NullPointerException” pada sebuah variabel t.

Dengan instruksi throw, program Eksepsi untuk kemudian ditangkap oleh instruksi catch. Namun tentunya yang ditangkap harus sesuai degan tipe eksepsi yang dilemparkan. Dalam hal ini program

**throws**

Eksepsi selalu ditangkap dengan **catch** pada blok dimana eksepsi tersebut dilempar. Tanpa **catch**, program tidak akan dikompilasi. Program dibawah ini menghilangkan **try** dan **catch** pada subroutine demo( ), namun dengan menggunakan tambahan kata kunci **throws** Exception.

Tabel 6.1 Common Jaxa Exceptions

|  |  |
| --- | --- |
| ***Exceptions*** | ***Description*** |
| Arithmeticexception | Disebabkan oleh error secara matematika seperti pembagian dengan nol |
| ArrayIndexOutOfBouns Exception | Disebabkan oleh array index yang salah |
| ArrayStoreException | Disebabkan ketika program mencoba menyimpan file data yang salah |
| FileNotFoundException | Disebabkan oleh usaha mengakses file yang tidak ada |
| IOException | Disebabkan oleh general I/O failures, seperti kegagalan membaca file |
| NullPointException | Disebabkan oleh mereferensi obyek null |
| NumberFormatException | Disebabkan ketika konversi antara string dan bilangan gagal |
| OutOfMemoryException | Disebabkan ketika tidak cukup memory untuk alokasi obyek |
| SecurityException | Disebabkan ketika sebuah applet mencoba melakukan aksi yang tidak diperbolehkan oleh setting security browser |
| StackOverflowException | Disebabkan ketika sistem run melebihi kapasitas stack |
| StringIndexOutOfBoundsException | Disebabkan ketika program mencoba mengakses posisi karakter yang tidak ada di dalam string |

**LEMBAR KERJA DAN TUGAS**

1. Menggunakan exception

public class arg01 {

public static void main (String argv []) {

String str="??????";

try {

str = argv[0];

}

catch(ArrayIndexOutOfBoundsException e) {

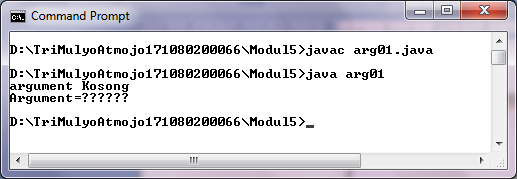
System.out.println("argument Kosong");

}

System.out.println("Argument="+str);

}

}



1. Menggunakan Throw

public class CobaThrow {

static void demo() {

try{

NullPointerException t;

t = new NullPointerException("Test Throw");

throw t;

//Barisinitakpernahdijalankan

}

catch (ArithmeticException e) {

System.out.println("Auuuu");

}

}

public static void main (String argv[]) {

try{

demo();

System.out.println("Selesai");

}

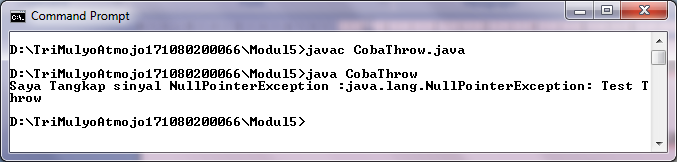
catch (NullPointerException e) {

System.out.println("SayaTangkapsinyalNullPointerException :" + e);

}

}

}



1. Menggunakan Throws

public class ThrowDemo {

static void demo()throws IllegalAccessException {

System.out.println("xxxxxx");

throw new IllegalAccessException ("ThrowDemo");

}

public static void main (String argv[] ) {

try{

demo();

System.out.println("Selesai");

}

catch (IllegalAccessException e) {

System.out.println("Sayatangkap " + e);

}

}

}



**TUGAS**

1. Buatlah program yang hanya bisa menerima input berupa angka saja. Bila

diinputkan salin angka (numeric) maka akan ditangkap oleh blok catch

importjava.util.\*;

public class Tugas1{

static Scanner input = new Scanner(System.in);

public static void main(String[] args){

Int angka;

try{

System.out.print(" MasukkanAngka = ");

angka = input.nextInt();

}

catch(Exception e){

System.out.println(" MAAF !! TolongInputkanAngka . . . BukanHuruf . . .");

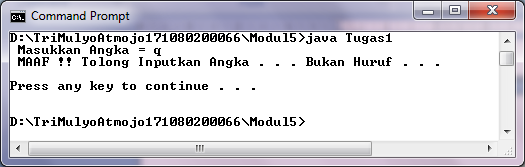
}

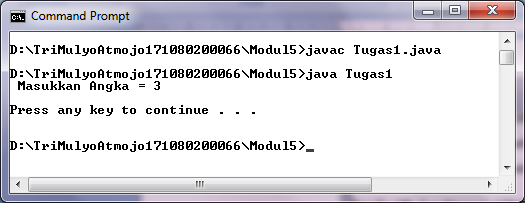
System.out.println(" \nPress any key to continue . . . ");

System.out.println();

}

}





1. Berdasarkan program diatas tambahkan blok finally untuk menyatakan bahwa

telah dilakukan pengecekan terhadap data yang diinputkan.

importjava.util.\*;

public class Tugas2{

static Scanner input = new Scanner(System.in);

public static void main(String[] args){

int angka;

try{

System.out.print(" MasukkanAngka = ");

angka = input.nextInt();

}

catch(Exception e){

System.out.println(" MAAF !! TolongInputkanAngka . . . BukanHuruf . . .");

}

finally{

System.out.println("Sudahdilakukanpengecekan");

}

System.out.println(" \nPress any key to continue . . . ");

System.out.println();

}

}

