MODUL PRAKTIKUM 8
EXCEPTION HANDLING

A. Dasar Teori

Exception Handling

Exception adalah event yang terjadi ketika program menemui kesalahan pada saat instruksi program dijalankan. Banyak hal yang dapat menimbulkan event ini, misalnya crash, harddisk rusak dengan tiba-tiba, sehingga program-program tidak bisa mengakses file-file tertentu. Programmer pun dapat menimbulkan event ini, misalnya dengan melakukan pembagian dengan bilangan nol, atau pengisian elemen array melebihi jumlah elemen array yang dialokasikan dan sebagainya.

Program Java, ketika terjadi kesalahan, otomatis akan dilemparkan sebuah objek yang disebut *exception*, yang kemudian dapat diproses lebih lanjut oleh fungsi-fungsi yang siap menangani kesalahan tersebut. Proses pelemparan *exception* tersebut sering dikenal dengan istilah *throwing exception*, sedangkan proses penerimaan exception yang bersangkutan dikenal dengan istilah *catch exception*.
Ada Beberapa subkelas yang diturunkan dari kelas *exception*, yaitu :

1. ***ClassNotFoundException***
   Terjadi bila ingin menggunakan kelas yang tidak ada atau belum dibuat.
2. ***CloneNotSupportedException***
   Terjadi bila ingin meng-clone atau menggandakan suatu kelas yang tidak didukung oleh method clone
3. ***RuntimeException***
   Disebabkan oleh proses inisialisasi program yang tidak sempurna
4. ***NullPointerException***
   Disebabkan karena pada saat runtime, terjadi pemanggilan atribut atau metode terhadap sebuah objek yang belum diinisialisasi, alias masih null.
5. ***ArrayIndexOutOfBoundsException***
   Disebabkan akses array yang melebihi kapasitas array yang ada
6. ***ArithmeticException***
   Khusus untuk operasi aritmatika integer. Seperti pembagian suatu bilangan integer dengan 0
7. **IOException**
   Terjadi bila ada I/O error, seperti gagal menemukan dan membuka file. User memasukkan input yang tidak valid. Subkelas ini memiliki beberapa subkelas lain, seperti InterruptedIOException, EOFException, serta FileNotFoundException.

Error dan Exception Classes

Seluruh exception adalah subclasses, baik secara langsung maupun tidak langsung, dari sebuah root class Throwable. Kemudian, dalam class ini terdapat dua kategori umum: Error class dan Exception class.
Exception class menunjukkan kondisi yang dapat diterima oleh user program. Umumnya hal tersebut disebabkan oleh beberapa kesalahan pada kode program.

Contoh dari exceptions adalah pembagian oleh 0 dan error di luar jangkauan array. Error class digunakan oleh Java run-time untuk menangani error yang muncul pada saat dijalankan. Secara umum hal ini di luar control user karena kemunculannya disebabkan oleh run-time environment. Sebagai contoh adalah out of memory dan harddisk crash.

Keyword pada *Exception Handling*

Ada 5 keyword penting dalam java dalam hal exception handling:

- ***Try***

Keyword ini biasanya digunakan dalam suatu block program. Keyword ini digunakan untuk mencoba menjalankan block program kemudian mengenai dimana munculnya kesalahan yang ingin diproses. Keyword ini juga harus dipasangkan dengan keyword catch atau keyword finally.

- ***Catch***

Jika kita sudah melihat penjelasan try maka secara tidak langsung kita sudah memahami kegunaan dari keyword ini. Dalam java, keyword catch harus dipasangkan dengan try. Kegunaan keyword ini adalah menangkap kesalahan atau bug yang terjadi dalam block try. Setelah menangkap kesalahan yang terjadi maka developer dapat melakukan hal apapun pada block catch sesuai keinginan developer. Keyword catch juga dapat diletakan berulang-ulang sesuai dengan kebutuhan.

- ***Finally***

Keyword ini merupakan keyword yang menunjukan bahwa block program tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada.

- ***Throw***

Keyword ini digunakan untuk melemparkan suatu bug yang dibuat secara manual.

- ***Throws***

Keyword throws digunakan dalam suatu method atau kelas yang mungkin menghasilkan suatu kesalahan sehingga perlu ditangkap errornya. Cara mendefinisikannya dalam method adalah
sebagai berikut :

```
<method modifier> type method-name throws exception-list1,
exceptio-list2, … {}
```

Mekanisme Mengantisipasi Exception

Berikut adalah kemungkinan skenario exception :

1. Tidak terjadi exception

   Program dijalankan, seluruh statement dalam blok try telah dieksekusi dan tidak terjadi exception sama sekali, maka blok catch tidak akan dieksekusi oleh interpreter.

2. Terjadi exception pada blok method tunggal

   Blok method tunggal disini adalah bila method itu tidak memanggil statement dari method yang lain. Bila saat interpreter mengeksekusi block try ada statement yang menyebabkan exception, maka interpreter akan langsung keluar dari blok try, kemudian mencari blok catch yang bersesuaian dengan exception yang terjadi. Jika interpreter menemukan catch yang sesuai maka interpreter akan mengeksekusi blok tersebut. Bila tidak ada catch yang sesuai, maka interpreter akan menghentikan program dan menampilkan pesan exception yang terjadi.

3. Terjadi exception pada blok method tersarang

   Blok method tersarang maksudnya adalah method yang mendeklarasikan exception (misal method A) memanggil method lainya (method B)

Method A memanggil Method B. Pada saat interpreter mengeksekusi statement dari method B ini, terjadi exception. Interpreter akan menghentikan eksekusi statement, selanjutnya mencari catch yang sesuai dalam method B. Bila catch ini tidak ditemukan (dalam method B), maka interpreter akan pergi ke meyhod A. Bila di temukan, maka interpreter akan mengeksekusi blok tersebut, namun bila tidak ada catch yang sesuai, interpreter akan menghentikan program dan menampilkan pesan exception.

Jenis Exception Handling

Try Catch

Untuk penanganan exception, dalam Java digunakan blok try dan catch. Blok try digunakan untuk menempatkan kode-kode program Java yang mengandung kode program yang mungkin melemparkan exception. Blok catch digunakan untuk menempatkan kode-kode program Java yang digunakan untuk menangani sebuah exception tertentu. Setelah kita tambahkan blok try – catch untuk mengatasi error yang terjadi, maka program akan menampilkan pesan error bahwa ada error yang terjadi pada konsol. Sintaks blok try – catch adalah sebagai berikut :

```
Try
{
... kode program yang mungkin menghasilkan
exception
}
Catch {
Exception xx}{...}
Catch {
exception xx}{...}
```

> ***Petunjuk Penulisan Program :***
> *Blok catch dimulai setelah kurung kurawal dari kode try atau catch terkait. Penulisan kode dalam blok mengikuti identasi.*

Untuk multiple catch akan kita gunakan jika blok try mungkin menimbulkan lebih dari satu exception. Sintaks blok multiple catch adalah sebagai berikut :

```
try { instruksi-1
 instruksi-2 instruksi-3 ……. diabaikan
 instruksi-n
}
catch(tipe_eksepsi_1 e1)
{
}
catch(tipe_eksepsi_2 e2)
{
}
catch(tipe_eksepsi_3 e3)
{
}
catch(tipe_eksepsi_4 e4)
{
}
....
catch(tipe_eksepsi_n en)
{
}

instruksi-lain
......

try {
instruksi-1
instruksi-2
instruksi-3
…….
instruksi-n
}
catch(tipe_eksepsi_1 e1)
        {
        }
        catch(tipe_eksepsi_2 e2)
```

```
{
}
catch(tipe_eksepsi_3 e3)
{
}
catch(tipe_eksepsi_4 e4)
{
}
....
catch(tipe_eksepsi_n en)
{
}

instruksi-lain
......
```

Try-Catch-Finally

Selain try – catch, kita dapat mendefinisikan blok try – catch dan finally yang memiliki proses yang lebih lengkap, karena pada finally kita dapat mendefinisikan kode program yang selalu dieksekusi, baik ada exception yang terjadi maupun bila tidak terjadi exception sama sekali. Bentuk umum dari blok try-catch-finally adalah sebagai berikut:

```
try{
//tulis pernyataan yang dapat mengakibatkan
exception //dalam blok ini
}
catch( <exceptionType1> <varName1> ){
//tulis aksi apa dari program Anda yang dijalankan
jika ada //exception tipe tertentu terjadi
}
. . .
catch( <exceptionTypen> <varNamen> ){
//tulis aksi apa dari program Anda yang dijalankan jika ada
//exception tipe tertentu terjadi
}
finally{
//tambahkan kode terakhir di sini
}
```

Exception dilemparkan selama eksekusi dari blok *try* dapat ditangkap dan ditangani dalam blok *catch*. Kode dalam blok *finally* selalu di-eksekusi. Berikut ini adalah aspek kunci tentang sintak dari konstruksi *try-catch-finally*:

- Notasi blok bersifat perintah

- Setiap blok *try*, terdapat satu atau lebih blok *catch*, tetapi hanya satu blok *finally.*
- Blok *catch* dan blok *finally* harus selalu muncul dalam konjungsi dengan blok *try*, dan diatas urutan.
- Blok *try* harus diikuti oleh **paling sedikit** satu blok *catch* atau satu blok *finally,* atau keduanya.

Keyword Finally

Try-catch yang menggunakan finally maka keyword finally tersebut fungsinya hampir sama dengan keyword default pada switch-case.
Ada 3 sekenario pemrosesan dengan keyword finally yaitu :

1. Bila tidak terjadi exception,maka blok finally akan di eksekusi . setelah selesai, interpreter akan mengeksekusi statement selanjutnya.
2. Bila terjadi exception, interpreter akan berhenti mengeksekusi statement dalam blok try berikutnya. Kemudian, interpreter akan mencari catch yang sesuai. Bila ditemukan, interpreter akan mengeksekusi catch dan finally.
3. Bila exception terjadi, namun tidak ada catch yang sesuai, maka statement – statemenet try berikutnya yang masih tersisa tidak akan di eksekusi. Selanjutnya, interpreter akan mengeksekusi blok finally.

Keyword throw

Kata kunci throw digunakan di program untuk melempar (throw) exception secara eksplisit. Bentuk umum kalimat adalah :

Throw ThrowableInstance;

Instan Throwable harus merupakan obyek dengan tipe Throwable atau subkelas dari Throwable. Terdapat dua cara untuk memperoleh obyek Throwable, yaitu :

- ✓ Menggunakan parameter di klausa catch
- ✓ Menciptakan salah satu dengan menggunakan operator new()

Eksekusi program akan dihentikan segera setelah kalimat throw. Java akan melakukan inspeksi blok try terdekat untuk menemukan catch yang cocok dengan dengan tipe exception yang dilempar. Jika Java menemukan, maka kendali program ditransfer ke kalimat itu. Jika tidak ditemukan, maka Java akan melakukan penelusuran ke blok berikutnya dan bila tetap tidak ditemukan, maka penanganan exception secara default akan menghentikan programnya.

Klausa throws mendaftarkan tipe-tipe exception yang dapat dilempar method. Hal ini diperlukan agar diketahui semua exception yang mungkin dilempar method atau subkelasnya, kecuali tipe Error atau RunTimeException yang dilakukan sistem Java secara otomatis bila menemui pelangggaran aturan-aturan Java. Semua exception yang hendak dilempar method harus dideklarasikan di klausa throws. Jika method melemparkan exception yang tidak dideklarasikan di deklarasi method, maka kompilator akan memberi pesan kesalahan.

Disamping menangkap exception, Java juga mengijinkan seorang user untuk melempar sebuah exception. Sintax pelemparan exception cukup sederhana. Dapat kita lihat sebagai berikut:

```
throw <exception object>;
```

Contoh :

```
/* Melempar exception jika terjadi
kesalahan input */ class ThrowDemo {

     public static void main(String args[]){
          String input = "invalid input";
          try {
               if (input.equals("invalid input"))
               {
                    throw new
               RuntimeException("throw
               demo"); } else {
                    System.out.println(input);
               }
          System.out.println("After throwing"); }
                    catch (RuntimeException e) {
               System.out.println("Exception
               caught here.");
               System.out.println(e);
          }
     }
}
```

Keyword Throws

Jika sebuah method dapat menyebabkan sebuah exception namun tidak menangkapnya, maka digunakan keyword throws. Berikut ini penulisan syntax menggunakan keyword throws :

```
<type> <methodName> (<parameterList>) throws
       <exceptionList> { <methodBody>
 }
```

Sebuah method perlu untuk menangkap ataupun mendaftar seluruh exceptions yang mungkin terjadi, namun hal itu dapat menghilangkan tipe Error, RuntimeException, ataupun subclass-nya. Contoh berikut ini menunjukkan bahwa method *myMethod* tidak menangani.

ClassNotFoundException :

```
class ThrowingClass {
static void myMethod() throws
     ClassNotFoundException { throw
     new ClassNotFoundException
     ("just a demo");
}
```

```
}
class ThrowsDemo {
public static void main(String args[]) {
     try {
          ThrowingClass.myMethod();
     } catch (ClassNotFoundException e) {
          System.out.println(e);
     }
}
}
```

Semua contoh program ada di file yang sudah diupload di classroom, silahkan didownload dan dipelajari.