PERTEMUAN KE – 11
Exception Handling

A. TUJUAN

Dapat mengenal berbagai macam exception

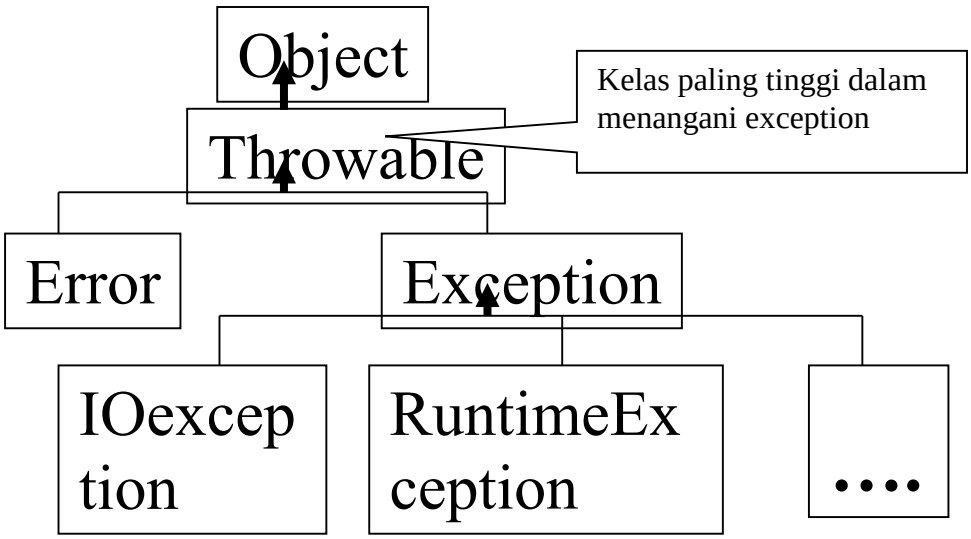
Dapat membuat exception handling

B. TEORI SINGKAT

Exception adalah sebuah masalah yang terjadi ketika program sudah pada tahap eksekusi, bukan saat tahap *compile.* Sebuah *exception* terjadi karena beberapa penyebab, misalnya: user memasukkan data yang tidak valid, file yang akan dibuka tidak ditemukan, koneksi yang belum tersambung atau tiba-tiba putus ditengah proses. Terdapat tiga macam exception:

1. Checked Exception : adalah exception yang biasanya terjadi karena kesalahan pengguna atau suatu masalah yang terjadi di luar perkiraan programmer. Contohnya sebuah file yang akan dibuka tapi file tersebut tidak dapat ditemukan maka exception terjadi. Kejadian seperti ini tidak terdeteksi pada saat tahap kompilasi.
2. Runtime Exception : adalah exception yang mungkin bisa dihindari oleh programmer. Kebalikan dari Checked Exception, bahwa Runtime Exception diabaikan pada tahap kompilasi.
3. Errors : adalah masalah yang terjadi diluar kontrol user ataupun programmer. Error sering diabaikan di dalam program karena anda tidak bisa menanganinya secara keseluruhan. Contohnya: jika kapasitas di memori penuh. Error seperti ini tidak terdeteksi pada tahap kompilasi

Hirarki kelas yang menangani exception:



Struktur penulisan exception adalah :

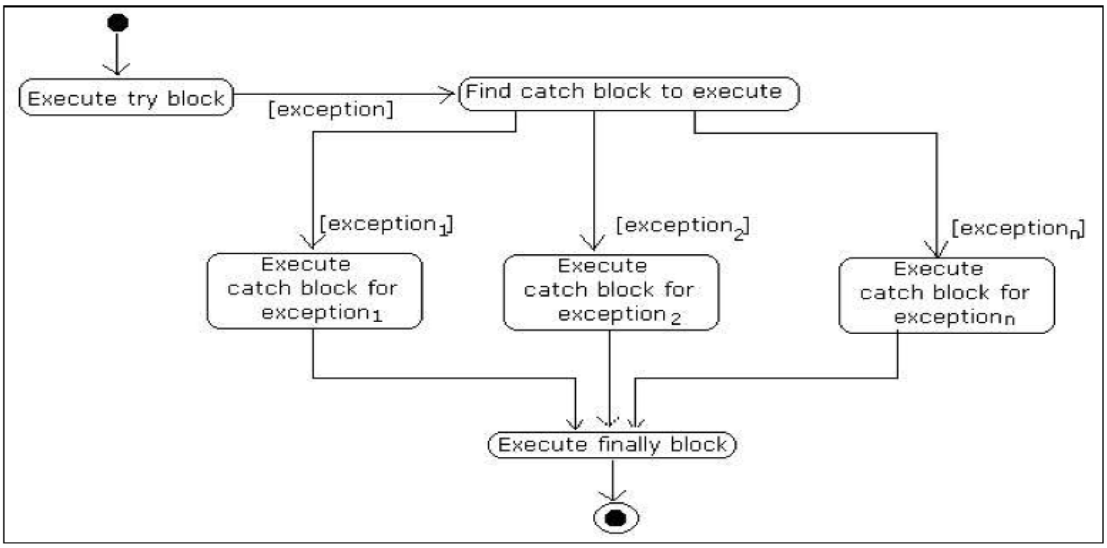
```
try {
   // blok program tempat menuliskan statement yang akan di uji apakan ada
kesalahan atau tidak
}
catch ( <tipe exception1> <nama variabel>) {
   // blok program tempat menuliskan aksi program bila sebuah exception
terjadi
}
. . .
catch ( <tipe exception2> <nama variabel>) {
   // blok program tempat menuliskan aksi program bila sebuah exception
terjadi
}
finally {
   // blok program lanjutan
}
```

Ketentuan dalam membuat exception adalah:

a. Wajib membuat notasi blok
b. Setiap blok try boleh memiliki lebih dari satu blok catch dan hanya boleh memiliki satu blok finally
c. Blok catch dan blok finally harus muncul bersama blok try
d. Blok try harus diikuti minimal satu blok catch, atau satu blok finally, atau kedua blok catch dan finally

e. Setiap blok catch mendefinisikan penanganan exception. Di dalam header blok catch terdapat satu argumen yang akan ditangani oleh blok exception. Exception harus berasal dari class Throwable atau dari class turunannya.

Alur kerja dari exception :



C. PRAKTIK

Praktik 1 . Membuat kelas tanpa exception

```
1. public class BagiNol {
2. public static void main(String[] args) {
3.     System.out.println("Sebelum pembagian");
4.     System.out.println(5/0);
5.     System.out.println("Sesudah pembagian");
6.  }
7. }
```

Baris ke-5 tidak akan dieksekusi karena ada kesalahan pembagian dengan bilangan nol pada baris ke-4

Praktik 2 . Membuat kelas menggunakan exception

```
1.  public class BagiNol2 {
2.  public static void main(String[] args) {
3.   System.out.println("Sebelum pembagian");
4.   try { System.out.println(5/0); }
5.      catch (Exception t) {
6.       System.out.print("Pesan kesalahan: ");
```

```
7.       System.out.println(t.getMessage());
8.     }
9.     System.out.println("Sesudah pembagian");
10.  }
11.  }
```

Program tidak berhenti di baris ke -4 , dibuktikan dengan munculnya hasil dari baris ke-9

Praktik 3 . Menggunakan lebih dari satu statement catch

```
public class MultiCatchException{
public static void main(String args[]){
   try{
    System.out.println(5/0);
   }
   catch(ArithmeticException e){
    System.out.println(0);
   }
   catch(ArrayIndexOutOfBoundsException e){
    System.out.println(1);
   }
   catch(Exception e){
     System.out.println(2);
   }
}
}
```

Output adalah angka nol yang artinya pada blok try terdapat kesalahan *arithmetic* maka kesalahan tersebut ditangkap oleh blok catch dari kelas ArithmeticException.

Praktik 4 . Menggunakan statement finally

```
public class MultiCatchException{
public static void main(String args[]){
   try{
    System.out.println(5/0);
   }
   catch(ArithmeticException e){
    System.out.println(0);
   }
   catch(ArrayIndexOutOfBoundsException e){
    System.out.println(1);
   }
   catch(Exception e){
     System.out.println(2);
   }
   finally{
     System.out.println(3);
   }
}
}
```

Bagian blok *finally* adalah bagian blok yang selalu dikerjakan. Output program adalah angka nol dan tiga, artinya setelah salah satu blok *catch* dieksekusi maka alur program selanjutnya adalah mengeksekusi bagian blok *finally.*

D. LATIHAN

- *Latihan diberikan oleh dosen pengampu pada saat praktikum.*
- *Dikerjakan di laboratorium pada jam praktikum.*

E. TUGAS

- *Tugas diberikan oleh dosen pengampu pada akhir praktikum.*
- *Dikerjakan di rumah dan dilampirkan pada laporan.*