Alfan F. Wicaksono

FASILKOM UI

Exception

Exception adalah penanda bahwa **kondisi/kejadian yang tidak diinginkan** telah terjadi pada program kita. Ada 2 jenis exception (+1 Error):

Ingat, kategori ini tidak sempurna !

1. **Checked Exception**
    a. Exception yang di-check saat di-compile oleh compiler
    b. Exception jenis ini **tidak dapat** dihindari oleh programmer. **Tetapi**, programmer **wajib** aware terhadap exception jenis ini dengan cara**:**
        i. menuliskan kata kunci **throws** yang diletakkan pada method header (ini artinya exception **tidak ditangani**, dan program akan berhenti)
        ii. maupun **menanganinya** dengan blok **try-catch** (program tidak akan berhenti)

        Contoh: ketika membuat program untuk membaca file, terjadi kondisi dimana nama file yang dimasukkan oleh user tidak ada di direktori -> **bukan salah programmer**, tapi programmer harus aware akan hal ini.

    c. Sebagian besar merupakan exception dalam hal input, output, atau dalam hal baca-tulis file. Contoh: **IOException** dan **turunannya** seperti **FileNotFoundException**, **EOFException**, dll.
2. **Unchecked Exception**
    a. Exception yang **tidak akan** di-check saat di-compile oleh compiler
    b. Exception ini **seharusnya dapat dihindari** oleh programmer. Jadi, kalau exception jenis ini terjadi, artinya **programmer yang bersalah**. Terserah Anda sebagai programmer mau menangani exception ini atau tidak.
        i. Contoh: ketika anda membuat array of integer dengan ukuran 3. Kemudian, anda mengakses indeks ke-5 dan mengakibatkan exception **IndexArrayOutOfBoundsException** dilemparkan -> **kesalahan programmer**.
    c. Yang masuk kategori ini adalah exception **RuntimeException** beserta turunannya seperti **ArithmeticException**, **ClassCastException**, **IllegalArgumentException**, **IndexArrayOutOfBoundsException, dll.**
3. **Error (tidak bisa di-recover)**
    a. Dalam hirarki kelas exception, Error tidak termasuk turunan kelas Exception.
    b. Error adalah kesalahan yang tidak bisa **di-recover**.
    c. Contoh: **OutOfMemoryError**

Gambar Hirarki dari Kelas Exception di Java. Tandai yang merupakan **Checked Exception** dan tandai yang merupakan **Unchecked Exception** !

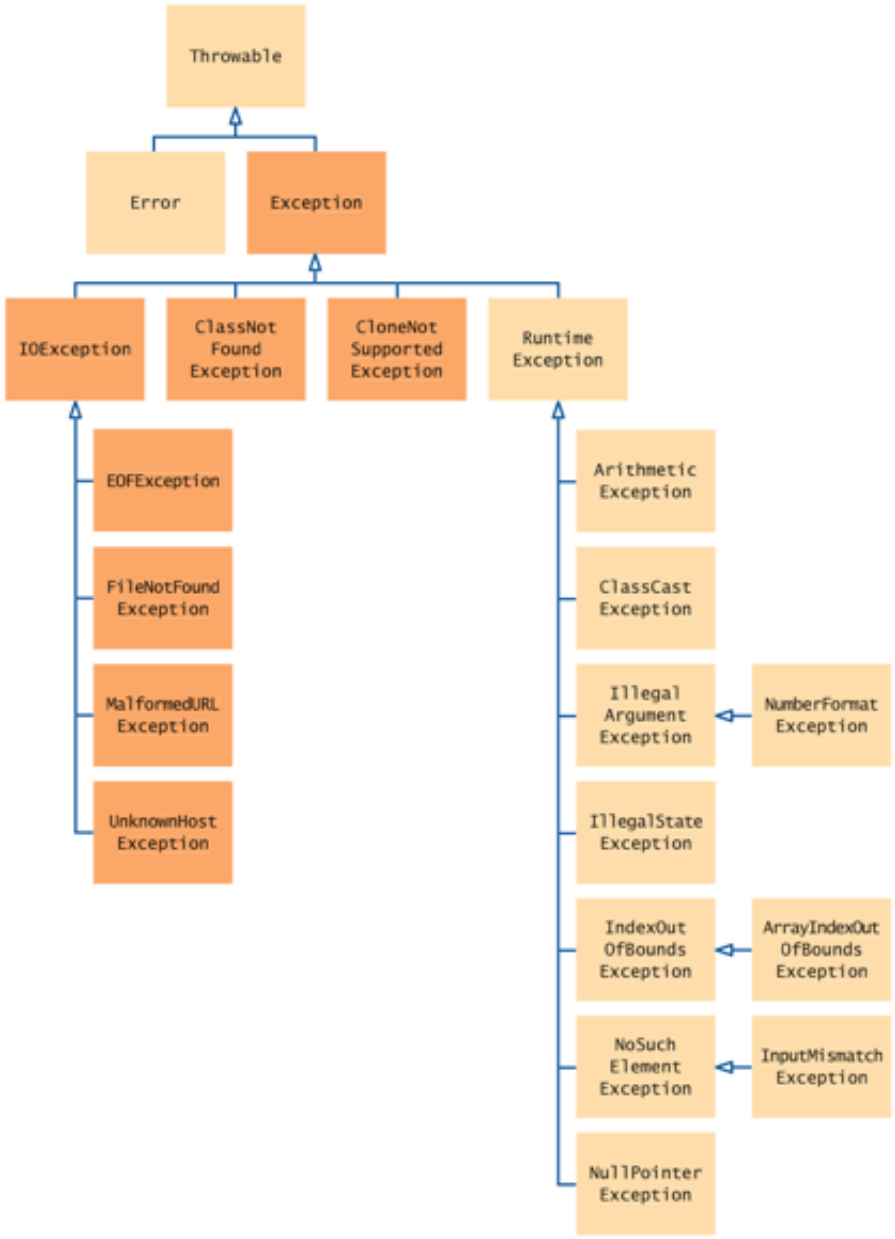

Figure 1 The Hierarchy of Exception Classes

Kode #1

```
public class Question1 {

   public static void main(String[] args)
   {
     int[]x = {0,1,2,3,4};

     try
     {

        /*   line 10   */

     }
     catch (IndexOutOfBoundsException e)
     {
        System.out.println("1");
        /* line 16 */
     }
     /* line 18  */
     catch (ArrayIndexOutOfBoundsException e)
     {
        System.out.println("2");
     }
     /* line 23 */
     finally
     {
        System.out.println("3");
     }
     System.out.println("x[0]"+x[0]);
   }
 }
```

Kode #2

```
public class Question2 {

   public static void main(String[] args)
   {
     String stri = "inner";
     String stro = "outer";

     try
     {
       throw new Exception();
     }
     catch (Exception eo)
     {
       try
       {
         throw new Exception();
       }
       catch (Exception ei)
       {
         System.out.println(stri);
       }
       finally
       {
         System.out.println("finally");
       }
     }
     finally
     {
       System.out.println(stro);
     }
     System.out.println(stro);
   }
 }
```

Soal terkait kode #1:

1. apakah kode #1 lulus kompilasi ? mengapa ?

2. bagaimana jika "`IndexOutOfBoundsException`" dan "`ArrayIndexOutOfBoundsException`" ditukar posisinya ? apakah masih tetap tidak lulus kompilasi ?

3. Jika pada **line 10** disisipkan kode **"x[5] = 10;",** dan **line 18** hingga **line 23 dihapus**, apa output yang dihasilkan ?

4. [lanjutan dari soal nomor 2] jika kemudian disisipkan kembali kode "**throw new RuntimeException("error ...");**" pada **line 16**, apakah output yang dihasilkan ?

Soal terkait kode #2:

Apakah output dari program tersebut ?

Apa output untuk program di bawah ?

```
public class Question3 {
   public static void main (String[] args) {
      Question3 q = new Question3();
      q.doA(1);   //1
      q.doA(2);   //2
      q.doA(3);   //3
   }

   public void doA(int a)
   {
     try
     {
       doB(a);
     }
     catch (IllegalArgumentException e)
     {
       System.out.println("1");
     }
     catch (IllegalStateException e)
     {
       System.out.println("2");
     } finally {
       System.out.println("doA beres");
     }
   }

   public void doB(int a)
   {
     int[] x = new int[4];
     try
     {
       if (a == 1) {
             x[5] = 10;
       } else if (a == 2) {
          int p = 1 / 0;
       } else {
          throw new IllegalArgumentException("test exception ...");
       }
     }
     catch (ArrayIndexOutOfBoundsException e)
     {
       System.out.println("3");
     }
     catch(ArithmeticException e)
     {
       System.out.println("4");
     } finally {
       System.out.println("doB beres");
     }
   }
 }
```