EXCEPTION HANDLING

KSP JAVA 2023

Dandy Candra Oktaviano | Tentor KSP JAVA

Image by AI with keyword
"Exception Handling"

Pendahuluan

Saat kita menulis sebuah kode program, kita pasti akan menemui sebuah error / masalah ketika mencoba menyelesaikan program tersebut. Beberapa jenis error yang sering ditemui adalah sebagai berikut:

1. Syntax error
   Error ini dapat terjadi ketika programmer menuliskan syntax yang salah saat penulisan kode. Sehingga ketika dilakukan kompilasi akan terjadi error.
   Contoh:
   - Lupa memberi titik koma pada akhir kode
   - Typo terhadap syntax tertentu, String namun ditulis string (‘s’ kecil)
2. Semantic error
   Error ini dapat terjadi ketika output yang dikeluarkan tidak sesuai dengan ekspetasi / harapan programmeer
   Contoh:
   - Kalian ingin membuat sebuah program untuk menghitung rata-rata nilai, namun ketika program dijalankan hasil output dari nilai rata-rata selalu bilangan bulat (40, 55 , 43). Ternyata terdapat kesalahan pada pemberian tipe data untuk menampung nilai rata – rata, yang seharusnya float/double bukan integer.
3. Runtime error
   Error ini dapat terjadi ketika program telah dikompilasi dengan sempurna tanpa adanya syntax error. Sehingga error akan terjadi pada saat program sedang berjalan yang menyebabkan program terhenti/*crash.*
   Contoh yang menyebabkan runtime error terjadi:
   - Pembagian dengan 0
   - Mengakses index array yang melebihi index maximal array tersebut
   - Program meminta inputan angka, namun huruf yang diinput.

Exception

Sebelum masuk ke exception handling, mari kita bahas exception terlebih dahulu. Exception sendiri merupakan sebuah error yang akan tampil ketika runtime error terjadi. Untuk lebih jelasnya dapat dilihat pada gambar di bawah.

```
16      public static void main(String[] args) {
17
18          System.out.println(5/0);
19
20          System.out.println("Masuk sinii");
21
22      }
```

```
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at exception_ksp.Exception_KSP.main(Exception_KSP.java:18)
C:\Users\VICTUS\AppData\Local\NetBeans\Cache\12.6\executor-snippets\run.x
C:\Users\VICTUS\AppData\Local\NetBeans\Cache\12.6\executor-snippets\run.x
BUILD FAILED (total time: 0 seconds)
```

Pada line 18 kita melakukan pembagian bilangan bulat dengan 0 yang menyebabkan JVM melemparkan exception berupa java.lang.ArithmeticException yang ditampilkan pada terminal,

setelah itu program mengalami crash dan terhenti pada line 18, sehingga line 20 tidak dijalankan oleh java.

Beberapa exception yang cukup terkenal pemrograman adalah sebagai berikut:

1. java.lang.ArithmeticException
   Terjadi ketika bilangan dibagi dengan 0.
2. java.lang.ArrayIndexOutOfBoundException
   Terjadi ketika mengakses sebuah index array yang melebihi index terakhir array yang diakses.
3. java.lang.NullPointerException
   Terjadi ketika objek masih merujuk pada null lalu objek tersebut digunakan.

   Masih banyak lagi jenis jenis exception yang dapat dilihat pada tautan berikut :
   https://programming.guide/java/list-of-java-exceptions.html

Lalu mengapa program tersebut langsung berhenti setelah menemukan sebuah Exception? Dikarenakan kita tidak "menangani" Exception tersebut, sehingga secara default program akan langsung berhenti dan tidak menjalankan syntax berikutnya. Maka dari itu sebuah Exception harus ditangani (handling) supaya program tetap menjalankan syntax berikutnya/tidak berhenti. Ibaratnya kita memberitahu program tersebut supaya ketika menemukan exception jangan berhenti namun tetap lanjutkan progran tersebut hingga akhir. Hal inilah konsep dari Exception Handling lahir.

Exception Handling

Beberapa kata kunci ketika kita ingin menerapkan Exception Handling adalah sebagai berikut:

1. Try
   Blok try akan diisi dengan kode yang berpotensi untuk mengeluarkan exception/error
2. Catch
   Blok catch digunakan untuk menangani exception, dalam kata lain jika ada sebuah exception/error makan program akan berpindah ke blok ini.
3. Finally
   Blok ini akan tetap dijalankan baik program tersebut tidak error atau mengeluarkan exception/error. Sehingga sifat dari blok ini adalah *optional.*
4. Throw
   Digunakan untuk melempar kelas Exception yang kita buat
5. Throws
   Digunakan untuk konstruktor atau method yang di dalamnya terdapat kemungkinan terjadi error.

Berikut adalah contoh penggunaan try, catch dan finally. Untuk throw dan throws akan dibahas pada bagian user defined eexception.

```
19
20          try{
21
22              System.out.println(5/0);
23
24          } catch (Exception e){
25
26          }
27
28
29          System.out.println("Masuk sinii");
30
```

Kita memberitahu program untuk mencoba (try) line code 22, jika terjadi exception maka akan di tangkap (catch) pada line 24 dan masuk ke blok catch.

Format penulisan dalam kurung catch adalah jenis exception + nama objek. Jika pada line 25 kita menulis System.out.println(e), maka terminal akan mengeluarkan output jenis exception yang terjadi.

```
run:
Masuk sinii
BUILD SUCCESSFUL (total time: 0 seconds)
```

Dapat dilihat ketika kita menggunakan try catch, meskipun ada sebuah exception/error program tetap menjalankan kode hingga akhir (line 29). Dikarenakan pada blok catch kita tidak memberikan syntax apapun, maka output yang dikeluarkan pada terminal tidak ada.

Beberapa kemungkinan yang terjadi ketika menggunakan try catch dan finally

1. Terjadi exception sehingga masuk ke blok catch

```
20          try{
21
22              System.out.println(5/0);
23              System.out.println("Tidak terjadi exception");
24
25          } catch (Exception e){
26              System.out.println(e);
27
28          } finally{
29              System.out.println("Masuk finally");
30          }
```

```
run:
java.lang.ArithmeticException: / by zero
Masuk finally
BUILD SUCCESSFUL (total time: 7 seconds)
```

Dapat dilihat baris kode 23 tidak dijalankan, karena pada baris kode 22 terjadi exception dan langsung masuk ke dalam blok catch. Blok finally akan dijalankan setelah blok catch. Baik terjadi exception atau tidak, blok finally akan tetap dijalankan.

2. Tidak terjadi exception sehingga blok catch tidak dijalankan

```
20          try{
21
22              System.out.println(5/2);
23              System.out.println("Tidak terjadi exception");
24
25          } catch (ArithmeticException e){
26              System.out.println(e);
27
28          } finally{
29              System.out.println("Masuk finally");
30          }
31
```

Tidak perlu bingung dengan penggunaan Exception / ArithmeticException. Kedua nya sama dalam penggunaan, perbedaanya Exception merupakan induk dari ArithmeticException, sehingga Exception dapat menangkap jenis Exception apapun, sedangkan ArithmeticException hanya dapat menangkap jenis exception pembagian bilangan 0 saja.

```
run:
2
Tidak terjadi exception
Masuk finally
BUILD SUCCESSFUL (total time: 0 seconds)
```

Dapat dilihat dikarenakan pada baris kode 22 tidak terjadi exception, maka blok catch tidak terjadi dan dilanjutkan dengan baris kode 23. Dan terakhir menjalankan blok finally.

3. Multiple Catch

Jika di dalam blok try terdapat beberapa kode yang memungkinkan terjadi jenis exception yang berbeda maka kita dapat menggunakan multiple catch. Lalu apakah dengan menggunakan 1 catch saja bisa? Tentu bisa, berikut adalah contohnya.

```
20              int a[] = {3,2,1,4};
21
22              try{
23
24                  System.out.println(a[4]);
25                  System.out.println(5/0);
26
27
28              } catch (Exception e){
29                  System.out.println(e);
30
31              } finally{
32                  System.out.println("Masuk finally");
33              }
```

```
java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4
Masuk finally
BUILD SUCCESSFUL (total time: 0 seconds)
```

Dapat dilihat terdapat 2 baris kode yang memungkinkan untuk terjadi error. Exception yang keluar adalah ArrayIndexOutOfBoundsException dikarenakan kode tersebut lebih dulu dijalankan. Lalu apa fungsi dari multiple catch jika menggunakan 1 catch aja sudah bisa asal jenis exceptionya adalah induk ?, fungsinya kita dapat mengetahui exception apa yang akan terjadi, contoh ketika yang terjadi adalah exception A maka kita akan melakukan hal yang ada di dalam blok exception A, jika yang terjadi adalah exception B maka kita akan melakukan hal yang ada di dalam blok exception B, sehingga kita dapat merencanakan apa yang terjadi jika exception A keluar ataupun exception B yang keluar dan tidak menjadi satu blok exception seperti contoh di atas. Berikut adalah contohnya.

```
22              try{
23
24                  //code code yang memungkinkan terjadinya error
25
26
27              } catch (ArithmeticException e){
28                  System.out.println(e);
29                  System.out.println("Bilangan tidak dapat dibagi dengan 0");
30
31              } catch(ArrayIndexOutOfBoundsException e){
32                  System.out.println(e);
33                  System.out.println("Kelebihan dalam mengakses index");
34
35              } catch(Exception e){
36                  System.out.println(e);
37
38              } finally{
39                  System.out.println("Masuk finally");
40              }
```

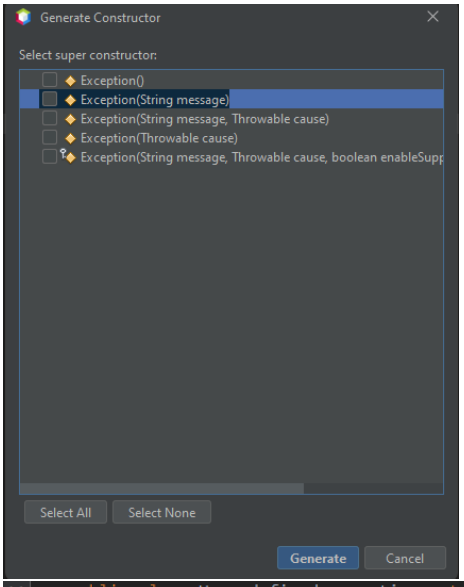
Dapat dilihat jika terdapat exception arithmetic maka akan masuk blok arithmetic, jika indexoutofbounds maka akan masuk blok indexoutofbounds, jika tidak keduanya namun terjadi exception dan kita tidak mengetahui jenis exception tersebut dan tidak menuliskanya secara hardcode seperti kedua exception di atas, maka akan masuk ke blok Exception yang general.

User Defined Exception

Ada kalanya ketika kita membuat sebuah program kita ingin menggunakan exception yang sesuai dengan program yang kita rancang, sehingga baik programmer maupun user dapat lebih mengerti ketika terjadi sebuah error karena informasi yang sekiranya lebih jelas, maka dari itu digunakanlah user defined exception. User Defined Exception adalah sebuah exception berbasis class yang kita buat sendiri dan merupakan kelas turunan dari kelas induk yaitu Exception. Untuk membuat exception sendiri wajib untuk mengextend kelas Exception supaya dapat menggunakan fungsi fungsi dari Exception seperti throw dan throws.

Langkah – langkah membuat kelas exception:

1. Membuat kelas user_defined_exception yang merupakan kelas turunan dari kelas Exception




```
11  public class User_defined_exception extends Exception {
12
13      // message nanti akan diisi dengan pesan yang ingin disampaikan ketika
14      // error terjadi
15      public User_defined_exception(String message) {
16          super(message);
17      }
18
19  }
```

Gambar 1.1

Dikarenakan kelas turunan dari Exception maka kita dapat menggunakan konstruktor yang berasal dari kelas Exception.

```java
public class User_defined_exception extends Exception {

    // Langsung mengisi pesan pada parameter supernya
    public User_defined_exception() {
        super("Ini pesan erornya");
    }

}
```

```java
public class User_defined_exception extends Exception {

    // Menggunakan method untuk menampilkan pesannya
    public void showMessage(){
        System.out.println("Ini pesan exception/errornya");
    }

}
```

Atau dapat langsung menulis pesan yang akan ditampilkan pada parameter supernya, atau kita juga bisa membuat sebuah method yang berisi pesan dari exception, karena pada dasarnya ini hanyalah kelas konkret biasa.

Kita akan menggunakan cara pada gambar 1.1 pada langkah ini.

2. Mengimplementasikan kelas exception yang telah dibuat pada kelas konkret/biasa
   Selanjutnya kita akan mengimplementasikan kelas exception yang telah kita buat.

```java
public class Mahasiswa {
    private String nama;
    private String npm;

    public Mahasiswa(String nama, String npm) throws User_defined_exception {

        if(nama.equals("") || npm.equals("")){
            throw new User_defined_exception("Nama atau NPM tidak boleh kosong[!]");
        }else{
            this.nama = nama;
            this.npm = npm;
        }

    }

}
```

Setelah tanda kurung tutup pada konstruktor, kita menuliskan keyword thorws dilanjutkan dengan kelas exception yang telah kita buat

NB : kita dapat menuliskan lebih dari 1 kelas exception yang kita buat

Jika nama atau npm nya kosong maka kita akan masuk ke baris kode 18, throw new memiliki arti bahwa kita akan melempar User_defined_exception dan akan ditangkap oleh blok catch pada main program jika nama atau npm nya kosong.

Pertama kita harus tentukan terlebih dahulu Exception yang kita buat, akan keluar/terlempar karena apa, pada gambar di atas kita menentukan bahwa kita akan mengeluarkan exception tersebut jika nama atau npm dari mahasiswa kosong.

3. Implementasi pada main program

```
23              try {
24                  Mahasiswa mahasiswa = new Mahasiswa("","200710670");
25              } catch (User_defined_exception e) {
26                  System.out.println(e.getMessage());
27              }
```

```
run:
Nama atau NPM tidak boleh kosong[!]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Kita membuat objek mahasiswa dengan nama yang kosong

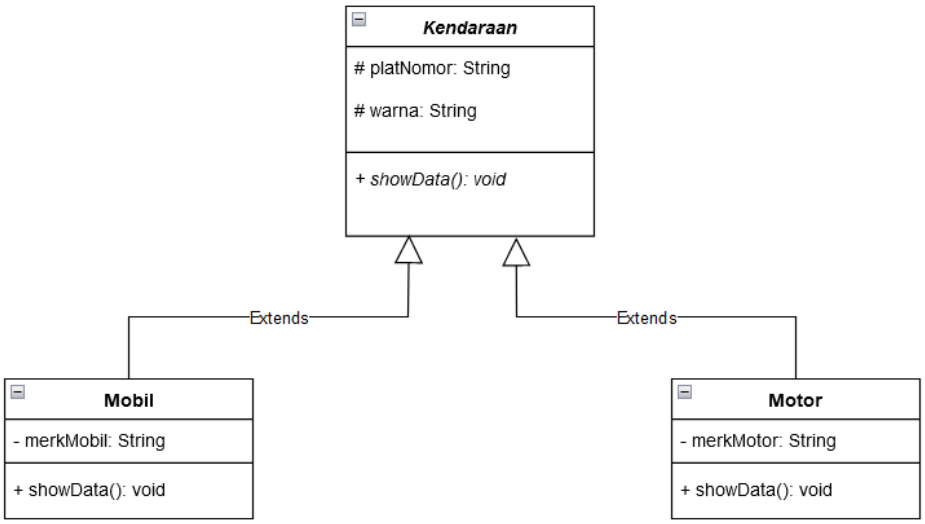
Kita tuliskan jenis exception yang telah kita buat

getMessage merupakan sebuah fungsi bawaan dari kelas induk Exception dengan tujuan mereturnkan string yang berasal dari parameter yang kita beri saat pembuatan kelas exception

Pada main program kita menulis pembentukan objek kelas Mahasiswa di dalam try, supaya jika nama / npm yang dimasukan kosong dapat ditangkap dan mengeluarkan errornya.

Guided / Challenge

Membuat sistem pencatatan kendaraan, berikut bentuk diagramnya:



Selain diagram di atas buatlah 3 buah kelas exception yang mempunyai ketentuan sebagai berikut :

1. Kelas exception pertama digunakan untuk memberi error handling yaitu atribut platnomor harus diawali dengan (‘AB’) karena basisnya di yogyakarta.
2. Kelas exception kedua digunakan untuk memberi error handling yaitu atribut warna memiliki maksimal 10 karakter
3. Kelas exception ketiga digunakan untuk memberi error handling yaitu atribut merkMobil atau merkMotor tidak boleh kosong.

Hasil:

```
run:
Plat nomor harus diawali dengan AB [!]
BUILD SUCCESSFUL (total time: 0 seconds)
```

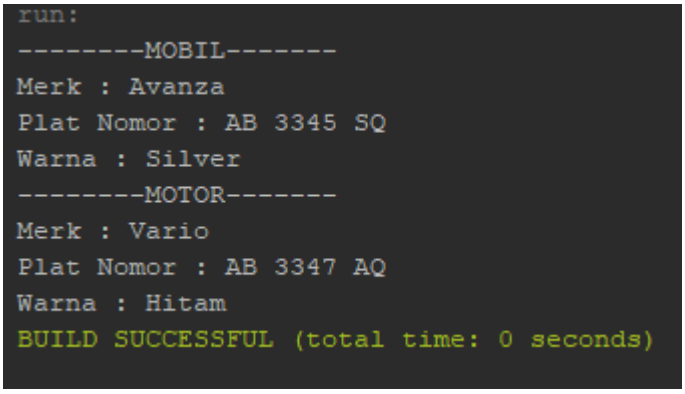
Jika plat nomor tidak diawali dengan AB

```
run:
Warna Harus Dibawah 10 Karakter [!]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jika warna lebih dari 10 karakter

```
run:
Merk Mobil Tidak Boleh Kosong [!]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jika merk mobil / motor kosong



Jika tidak terjadi error.

Bagi teman-teman yang ingin mengerjakan selamat mengerjakan!, bagi yang tidak atau bingung dapat langsung menonton video penjelasan modul ini ya, pembahasan soal ada pada bagian akhir video. Untuk kode program dapat didownload pada tautan di bawah, Terima kasih!.

Source code: https://github.com/Dandy-Candra/GDExceptionHandling_KSP_JAVA