Exception Handling

Teosofi Hidayah Agung
Hafidz Mulia

Departemen Matematika
Institut Teknologi Sepuluh Nopember

14 April 2025

Game 1

Carilah kesalahan/eror pada kode berikut!

```
1 public class Week4 {
2   public static void main(String[] args) {
3     int a[]=new int[5];
4     a[5]=100;
5     System.out.println("Tralalelo Tralala");
6   }
7 }
```

Game 2

Carilah kesalahan/eror pada kode berikut!

```
1  public class Week4 {
2    public static void main(String[] args) {
3      int result=1;
4      for (int i = 5; i > 0; i--) {
5        for (int j = 1; j < 5; j++) {
6            System.out.println("i = " + i + " j = " + j);
7            int result =  (i*j)/result;;
8            System.out.println("Hasil 1/i = " + result);
9        }
10     }
11     System.out.println("Bombardino Crocodilo");
12   }
13 }
```

Game 3

Carilah kesalahan/eror pada kode berikut!

```java
public static void main(String[] args) {
    String[] names = new String[15];
    names[0] = "Tung";
    names[2] = "Tung";
    names[4] = "Tung";
    names[6] = "Tung";
    names[8] = "Tung";
    names[9] = "Tung";
    names[5] = "Tung";
    names[7] = "Tung";
    names[3] = "Tung";
    names[10] = "Tung";
    names[11] = "Sahur";

    for (int i = 0; i < names.length; i++)
      System.out.println("names[" + i + "] = " +
            names[i]);

    System.out.println("\nStecu:");
```

```java
  for (int i = 0; i < names.length; i++) {
     if (names[i] != null)
        System.out.println("nama" + i + ": " +
              names[i].length());
     else {System.out.println("nama" + i + ":
           null");}
  }
  System.out.println("\nABCD:");
  int countNull = 0;
  for (int i = 0; i < names.length; i++)
   if (names[i] == null) countNull++;

  System.out.println("Kernel: " + countNull);
  System.out.println("\nNotul:");
  for (String name : names)
      if (name != null)
           System.out.println(name.toUpperCase());

  int length = names[1].length();
  System.out.println("Simpansini Bananini");
}
```

Masalah

Setelah di-*run*, 3 kode sebelumnya tidak bisa menjalankan baris kode terakhir dalam fungsi `main`. Hal tersebut terkadang membuat program yang eror tidak dapat terdeteksi dimana eror tersebut berada.

Dalam sebuah program yang **cukup besar** dan kelakuan *user* yang tidak bisa diprediksi, maka dibutuhkan suatu *tools* untuk menangani kesalahan apa saja yang mungkin saja terjadi dalam program yang telah dibuat.

Daftar isi

Exception

Definisi

Exception *di Java adalah mekanisme untuk menangani kesalahan (error) yang terjadi saat program berjalan (runtime error), agar program tidak langsung crash dan bisa menangani situasi yang tidak normal secara terkendali.*

Singkatnya ***Exception*** *adalah objek yang mewakili sebuah kondisi error yang terjadi saat eksekusi program.*

Exception

Contoh

Pada dasarnya, beberapa eror di java menghasilkan suatu variabel yang disebut **exception**.

- **ArithmeticException** adalah exception yang terjadi ketika terjadi kesalahan aritmatika, seperti pembagian dengan nol.
- **ArrayIndexOutOfBoundsException** adalah exception yang terjadi ketika mencoba mengakses indeks array yang tidak valid.
- **NullPointerException** adalah exception yang terjadi ketika mencoba mengakses metode atau variabel dari objek yang bernilai null.

Exception

Kode: *ArithmeticException*

```java
public class ArithmeticErrorExample {
  public static void main(String[] args) {
      int result = 1 / (9/11);
      System.out.println("Hasil: " + result);
  }
}
```

Kode: *NullPointerException*

```java
public class NullPointerErrorExample {
  public static void main(String[] args) {
      String text = null;
      System.out.println("Panjang string: " + text.length());
  }
}
```

Exception

Kode: *ArrayIndexOutOfBoundsException*

```java
public class ArrayIndexErrorExample {
  public static void main(String[] args) {
      int[] numbers = {10, 20, 30};
      System.out.println("Angka di indeks 5: " + numbers[5]);
  }
}
```

Daftar isi

Try-Catch

Definisi

Try *adalah blok kode yang digunakan untuk menangkap kesalahan yang mungkin terjadi dalam program. Jika terjadi kesalahan, maka program mengkonversinya dalam sebuah exception dan akan mengeksekusi kode di dalam blok* ***catch****.*

Eror dan Exception

Jika tidak ada yang menangkap variabel exception (tidak ada `try-catch`), maka akan menyebabkan program terhenti sebelum mencapai baris kode terakhir dalam sebuah fungsi.

Try-Catch

Kode: Struktur *Try-Catch* pada Java

```
1   try {
2       // Kode yang mungkin menyebabkan exception
3   } catch (Exception e) {
4       // Menangkap exception dan menampilkan pesan kesalahan
5   }
```

Print Console

Alangkah baiknya jika menampilkan pesan eror menggunakan `System.err.println()`

Try-Catch

Kode: Contoh *Try-Catch* #1

```
1  int bil = 10;
2  try {
3      System.out.println(bil / 0);
4  } catch (Exception e) {
5      System.out.println("Ini menghandle error yang terjadi");
6  }
```

Kode: Contoh *Try-Catch* #2

```
1  int bil = 10;
2  try {
3      System.out.println(bil / 0);
4  } catch (ArithmeticException e) {
5      System.out.println("Ini menghandle error yang terjadi");
6  }
```

Definisi

Finally *adalah bagian dari blok try-catch-finally yang selalu dijalankan, terlepas dari apakah terjadi exception atau tidak.*

Penerapan

Secara garis besar, *finally* biasanya digunakan untuk membersihkan resource seperti:

- Menutup koneksi jaringan
- Menutup stream file
- Menutup koneksi database
- Menghapus data sementara
- dll

Kode: *Try-Catch-Finally* pada Java

```java
public class FinallyExample1 {
  public static void main(String[] args) {
      try {
          int hasil = 10 / 0;
      } catch (ArithmeticException e) {
          System.out.println("Terjadi error: " + e);
      } finally {
          System.out.println("Blok finally dijalankan (walau ada
              error)");
      }
  }
}
```

Daftar isi

Throw

Definisi

Throw *adalah keyword yang digunakan untuk melempar (throw) sebuah objek exception secara eksplisit selama eksekusi program.*

Karena `exception` adalah objek, maka perlu diinisialisasi (menggunakan `new`) terlebih dahulu sebelum dilemparkan.

Kode: Syntax *Throw*

```
1       throw new ExceptionType("pesan error");
```

Throw

Kode: Contoh *Throw*

```
1  public static void main(String[] args) {
2      int umur = -5;
3      if (umur < 0)
4          throw new IllegalArgumentException("Umur tidak boleh negatif!");
5      System.out.println("Umur: " + umur);
6  }
```

Throw

Kode: Contoh *Throws*

```java
public class Main {
    public static int getElement(int[] arr, int index) throws Exception {
        return arr[index];
    }

    public static void main(String[] args) {
        int[] angka = {10, 20, 30};
        try {
            int hasil = getElement(angka, 5);  // Index 5 tidak ada
            System.out.println("Elemen: " + hasil);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Throw

throw	**throws**
Digunakan untuk melempar exception di dalam sebuah method	Digunakan untuk menyatakan jenis exception yang mungkin dilemparkan oleh sebuah method
Tidak dapat melempar lebih dari satu exception secara langsung	Dapat mendeklarasikan lebih dari satu jenis exception
• `throw` diikuti oleh objek baru (dari tipe exception) • Digunakan di dalam tubuh method	• `throws` diikuti oleh nama class exception • Digunakan di bagian deklarasi method (signature)

Tabel: Perbandingan antara `throw` dan `throws` dalam Java

Daftar isi

```java
String a;
String[] nama = new String[1];
try {
    statement1;
    nama[2] = "Dinda";
    statement2;
}
catch (ArithmeticException ex1) {
    System.err.println(ex1);
}
catch (Exception ex2) {
    System.out.println(ex2);
}
finally {
    System.out.println("stecu");
}
System.out.println("choco-minto");
```

Latihan 1

Apa output dari program tersebut apabila terjadi kondisi seperti ini, jelaskan:

1. Jika `statement1` dan `statement2` tidak terjadi exception (pernyataannya benar)
2. Jika `statement1` diganti dengan `a=3;` dan `statement2` tidak terjadi exception
3. Jika `statement1` tidak terjadi exception, dan `statement2` diganti dengan `System.out.println(1/0);`
4. Jika `statement1` diganti dengan `System.out.println(1/0);` dan `statement2` diganti `nama[0].length();`