

# Pemrograman Berorientasi Obyek

## Exception Handling

Oleh Politeknik Elektronika Negeri Surabaya  
2020




Politeknik Elektronika Negeri Surabaya  
Departemen Teknik Informatika dan Komputer

1

Politeknik Elektronika Negeri Surabaya

## Definisi Exception

- Suatu mekanisme penanganan kesalahan.
- Event yang terjadi ketika program menemui kesalahan saat instruksi program dijalankan.



Departemen Teknik Informatika & Komputer

2

## Exception

- Exception sering digunakan dalam akses sumberdaya non memori.

Catatan:

- Exception = untuk menangani kesalahan ringan (mild error).
- Error = menangani kesalahan berat (serious error).

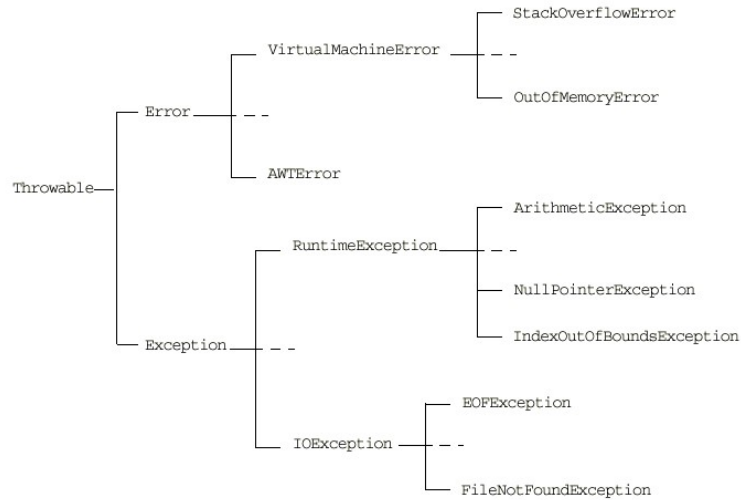


## Contoh kesalahan yang terjadi:

- Pembagian bilangan dengan 0
- Pengisian elemen array diluar ukuran array
- Kegagalan koneksi database
- File yang akan dibuka tidak exist
- Operand yg akan dimanipulasi out of prescribed range
- Mengakses obyek yang belum diinisialisasi



## Exception categories



## Purpose of each exception

- **Error** : mengindikasikan bahwa error yang terjadi adalah fatal error (severe problem) dimana proses recovery sangat sulit dilakukan bahkan tidak mungkin dilakukan.
  - Contoh : program running out of memory
- **RuntimeException** : mengindikasikan kesalahan implementasi atau desain program.
  - Contoh : ArrayIndexOutOfBoundsException
- **Other exception** : mengindikasikan kesalahan environment.
  - Contoh : file not found, invalid URL exception



## Common Exception

- ArithmeticException
  - Hasil dari operasi divide-by-zero pada integer
  - Misal: `int i = 12/0;`
- NullPointerException
  - Mencoba mengakses atribut atau method suatu object padahal object belum dibuat.
  - Misal: `Date d = null;`  
`System.out.println(d.toString());`
- NegativeArraySizeException
  - Mencoba membuat array dengan ukuran negatif.
- ArrayIndexOutOfBoundsException
  - Mencoba mengakses elemen array dimana index nya melebihi ukuran array.
- SecurityException
  - Biasanya dilempar ke browser, class security manager melempar exception untuk applet yang mencoba melakukan:
    - Mengakses lokal file
    - Open socket ke host yg berbeda dgn host yg di open oleh applet



## Apa yang terjadi jika terjadi kesalahan?

- Secara otomatis akan dilempar sebuah object yang disebut dgn **exception**.
- Exception dapat diproses lebih lanjut oleh fungsi-fungsi yang siap menangani kesalahan.
- Proses pelemparan exception disebut dgn **throwing exception**.
- Proses penerimaan exception disebut dengan **catch exception**.



## Contoh kejadian error (loading file from the disk)

```
int status = loadTexfile();
if (status != 1) {
    // something unusual happened, describe it
    switch (status) {
        case 2:
            // file not found
            break;
        case 3:
            //disk error
            break;
        case 4:
            //file corrupted
            break;
        default:
            // other error
    }
} else {
    // file loaded OK, continue with program
}
```



## Misal terdapat algoritma program:

### Fungsi bacaFile

BukaFile

BacaBarisFileSampaiHabis

TutupFile



Ditambahkan program untuk pengecekan berhasil tidaknya pembacaan file

Fungsi bacaFile

BukaFile

*Jika Gagal Buka File*

*Lakukan Sesuatu*

*Jika Berhasil Buka File*

BacaBarisFileSampaiHabis

TutupFile



- Bagaimana bila ditambahkan program untuk pengecekan terhadap status pembacaan file?
- Bagaimana bila ditambahkan program untuk pengecekan terhadap status penutupan file?
- Maka program akan menjadi sangat panjang dan banyak terdapat nested if-else.



## Solusi?

- Gunakan exception
- Bentuk:

```
try {  
.....  
} catch (ExceptionType x) {  
.....  
}
```



- 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.



## Implementasi 1

```
try {
    Fungsi bacaFile
    BukaFile
    BacaBarisFileSampaiHabis
    TutupFile
} catch (KesalahanBukaFile) {
    // lakukan sesuatu
}
```



## Try dgn banyak catch

- Dapat digunakan beberapa blok catch untuk satu blok try.
- Exception dalam satu program bisa mengatasi banyak exception.
- Contoh implementasi:
- Misal dalam satu blok try terdapat kemungkinan terjadi:
  - NullPointerException
  - IndexOutOfBoundsException
  - ArithmeticException

```
try {
    .....
} catch (ExceptionType1 x1) {
    .....
} catch (ExceptionType2 x2) {
    .....
}
```





## Implementasi 2

```
try {  
    Fungsi bacaFile  
    BukaFile  
    BacaBarisFileSampaiHabis  
    TutupFile  
} catch (KesalahanBukaFile) {  
    // lakukan sesuatu  
} catch (KesalahanAlokasiMemori) {  
    // lakukan sesuatu  
} catch (KesalahanTutupFile) {  
    // lakukan sesuatu  
}
```



## Object Exception

- Object exception yang dihasilkan dapat dimanfaatkan untuk mengetahui lebih lanjut mengenai error atau exception yang terjadi.
- Exception merupakan subclass dari class Throwable.



## Method yang diwarisi oleh exception:

- getMessage()
  - method ini mengembalikan isi pesan untuk menggambarkan exception yang terjadi
- printStackTrace()
  - method ini menampilkan pesan error dan stack trace ke standard error output stream yang biasanya merupakan konsol window apabila program merupakan program konsol.
- printStackTrace(PrintStream s)
  - method ini mengembalikan pesan error ke objek PrintStream yang dijadikan parameter. Apabila ingin menampilkan pesan ke konsol, anda dapat menggunakan `System.out` sebagai parameter.



## Blok Try – Catch - Finally

- Blok finally : digunakan untuk mendefinisikan kode program yang selalu dieksekusi baik ada exception yang terjadi maupun bila tidak terjadi exception sama sekali.
- Bentuk:

```
try {
    .....
} catch (Exception e) {
    .....
} finally {
    .....
}
```



## Contoh: Tanpa Exception Handling

```

1  public class HelloWorld {
2      public static void main (String[] args) {
3          int i = 0;
4
5          String greetings [] = {
6              "Hello world!",
7              "No, I mean it!",
8              "HELLO WORLD!!"
9          };
10
11         while (i < 4) {
12             System.out.println (greetings[i]);
13             i++;
14         }
15     }
16 }

```



## Contoh: Dengan Exception Handling

```

1  public class HelloWorld {
2      public static void main(String[] args) {
3          int i = 0;
4
5          String[] greetings = {
6              "Hello world!",
7              "No, I mean it!",
8              "HELLO WORLD!!"
9          };
10
11         while (i < 4) {
12             try {
13                 System.out.println(greetings[i]);
14                 i++;
15             } catch (ArrayIndexOutOfBoundsException e){
16                 System.out.println("Re-setting Index Value");
17                 i = 0;
18             } finally {
19                 System.out.println("This is always printed");
20             }
21         }
22     }
23 }

```



## Mendefinisikan method yang melempar exception

- Dilakukan bila method tidak ingin menangani exception sendiri.
- Method tertentu dlm program mungkin akan menghasilkan error yang tidak dikenali secara otomatis oleh Java Virtual Machine.
- Berlaku bagi kategori exception yg bukan subclass dari RuntimeException.
- Contoh: EOFException, MalformedURLException
- Dengan cara membuat method yang dapat melempar exception.



## Contoh method yang melempar exception

```

Class Gambar{
    public Image loadImage(String s)
        throws EOFException, MalformedURLException {
        If(error pembacaan akhir file)
            throw new EOFException()
    }
}

```



## Membuat class exception baru

- Sebuah subclass dari exception dapat dibuat sendiri oleh programmer untuk mendefinisikan sendiri secara lebih rinci tentang exception yang dapat terjadi.
- Class exception baru ini harus merupakan subclass dari `java.lang.Exception`.



## Membuat Exception

- Tujuan: mendefinisikan class exception yang lebih spesifik untuk keperluan tertentu.
- Untuk membuat class exception baru maka class itu harus merupakan subclass dari class Exception.



Contoh:  
Membuat class exception baru

```
class Salah extends Exception{  
    public Salah(){}  
    public Salah(String pesan){  
        super(pesan);  
    }  
}
```



```
public class TesSalah{  
    public static void main(String [] arg) throws Salah{  
        Salah s = new Salah("Salah disengaja ha..ha..");  
        int i = 0;  
        if (i==0)  
            throw s;  
    }  
}
```

\* Kode diatas adalah contoh method yang tidak menangani exception/  
melempar exception ke method yg menggunakan.



```
public class TesSalah{
    public static void main(String [] arg {
        Salah s = new Salah("Salah disengaja ha..ha..");
        int i = 0;
        try{
            if (i==0) throw s;
        } catch (Salah s){
            System.out.println(s.getMessage());
        }
    }
}
```

\* Kode diatas adalah contoh method yang menangani exception sendiri.



**bridge to the future**

<http://www.eepis-its.edu>

## Tugas

1. Buatlah review mengenai definisi exception, jenis exception, dan berikan 2 contoh program yang menyebabkan exception beserta cara penanganannya.
2. Buatlah review 1 halaman mengenai penanganan exception dengan cara melempar exception dan berikan 1 contoh program.



1. Oracle Java Documentation, The Java™ Tutorials, <https://docs.oracle.com/javase/tutorial/>, Copyright © 1995, Oracle 2015.
2. Tita Karlita, Yuliana Setrowati, Rizky Yuniar Hakkun, Pemrograman Berorientasi Obyek, PENS-2012
3. Sun Java Programming, Sun Educational Services, Student Guide, Sun Microsystems, 2001.
4. John R. Hubbard, Programming With Java, McGraw-Hill, ISBN: 0-07-142040-1, 2004.
5. Patrick Niemeyer, Jonathan Knudsen, Learning Java, O'reilly, CA, ISBN: 1565927184, 2000.
6. Philip Heller, Simon Roberts, Complete Java 2 Certification Study Guide, Third Edition, Sybex, San Francisco, London, ISBN: 0-7821-4419-5, 2002.
7. Herbert Schildt, The Complete Reference, Java™ Seventh Edition, Mc Graw Hill, Osborne, ISBN: 978-0-07-163177-8, 2007



## Overriding Method dan Exception

- Overriding method hanya boleh melempar exception yang merupakan subclass dari exception yang dilempar oleh overridden method atau sama.
- Overriding method boleh mendeklarasikan exception lebih sedikit dari jumlah exception kepunyaan overridden method.

Catatan;

- Overriding method = method yang mengoveride.
- Overridden method = method yang dioveride.



## Contoh 1: Method Overriding

```

1 public class TestA {
2     public void methodA() throws RuntimeException {
3         // do some number crunching
4     }
5 }

1 public class TestB1 extends TestA {
2     public void methodA() throws ArithmeticException {
3         // do some number crunching
4     }
5 }

1 public class TestB2 extends TestA {
2     public void methodA() throws Exception {
3         // do some number crunching
4     }
5 }

```

- Class TestB1 → ok karena ArithmeticException merupakan subclass dari RuntimeException.
- Class TestB2 → error karena Exception merupakan superclass dari RuntimeException.



## Contoh 2: Method Overriding

```

1  import java.io.*;
2
3  public class TestMultiA {
4      public void methodA()
5          throws IOException, RuntimeException {
6          // do some IO stuff
7      }
8  }

1  import java.io.*;
2
3  public class TestMultiB1 extends TestMultiA {
4      public void methodA()
5          throws FileNotFoundException, UTFDataFormatException,
6              ArithmeticException {
7          // do some IO and number crunching stuff
8      }
9  }

```

- Class TestMultiB1 → ok karena FileNotFoundException dan UTFDataFormatException merupakan subclass dari IOException
- Dan Arithmetic Exception merupakan subclass dari RuntimeException.



## Contoh 3: Method Overriding

```

1  import java.io.*;
2
3  public class TestMultiA {
4      public void methodA()
5          throws IOException, RuntimeException {
6          // do some IO stuff
7      }
8  }

1  import java.io.*;
2  import java.sql.*;
3
4  public class TestMultiB2 extends TestMultiA {
5      public void methodA()
6          throws FileNotFoundException, UTFDataFormatException,
7              ArithmeticException, SQLException {
8          // do some IO, number crunching, and SQL stuff
9      }
10 }

```

- Class TestMultiB2 → error karena SQLException atau superclass dari SQLException tidak dideklarasikan pada class TestMultiA.
- TestMultiB2 tidak boleh menambahkan exception baru



## Contoh 4: Method Overriding

```
1 import java.io.*;
2
3 public class TestMultiA {
4     public void methodA()
5         throws IOException, RuntimeException {
6         // do some IO stuff
7     }
8 }

```

```
1 public class TestMultiB3 extends TestMultiA {
2     public void methodA() throws java.io.FileNotFoundException {
3         // do some file IO
4     }
5 }
```

- Class TestMultiB3 → ok karena FileNotFoundException adalah subclass dari IOException.
- Contoh diatas menunjukkan bahwa overriding method boleh mendeklarasikan exception yang lebih sedikit dari exception kepunyaan override method.

