



# INSTITUT TEKNOLOGI DEL

## MATERI PRAKTIKUM

### PEMROGRAMAN BERORIENTASI OBJEK

#### (IF321314/IF421314)

Semester I Tahun Ajar 2016/2017

Tanggal Sesi	: 4 November 2016
Minggu ke-/sesi	: 9/3
Topik	: Exception Handling
Aktifitas	: Mahasiswa mempelajari <i>exception handling</i> pada Java
Tujuan praktikum	: Mahasiswa mampu memahami <i>exception handling</i> pada Java
Waktu pengerjaan	: 2 x 50 menit
Setoran	: File proyek hasil pengerjaan contoh program dikompres menjadi file dengan ekstensi .rar. Nama file dengan format: pbo_w09s03_NIM.rar. NIM diganti dengan NIM Anda.
Batas penyerahan	: 4 November 2016 sebelum pukul 21.30 WIB
Tempat Penyerahan	: <a href="http://ecourse.del.ac.id">http://ecourse.del.ac.id</a>
Sarana	: IDE Net Beans 8.2, JDK 1.8, Laptop
Referensi	: -

## 1 Langkah-langkah Praktikum

Pada praktikum kali ini, Anda diharapkan untuk menuliskan program dan menyelesaikan persoalan dari masing-masing program. Coba Anda jawab semua pertanyaan yang muncul pada bagian-bagian langkah praktikum ini. Untuk submission, yang disetor hanya yang disebutkan pada bagian 2. Tugas dokumen ini.

### 1. Uncaught Exception

```
class UncaughtException {  
    public static void main(String args[]) {  
        System.out.println(3/0);  
    }  
}
```

- Analyze what will happen if the above program is compiled and executed.
- What's the result?

### 2. Handling the exception

```
public class Test{  
    public static void main(String[] args){  
        try{  
            System.out.println(3/0);  
        }  
    }  
}
```

```

    }
    catch(ArithmeticException ex) {
        System.out.println("Error"+ex);
    }
}
}

```

- a Replace **ex** with `ex.getMessage()` and `ex.printStackTrace()`. Try also to execute statement `ex.printStackTrace()` and see what's the result.
- b Replace `ArithmeticException` with `Exception`. What's the result? Explain.

### 3. Using finally

```

class CaughtExceptionFinally {
    public static void main(String args[]) {
        int n = args.length;
        try {
            if(n == 0) {
                System.out.println(3/n);
                System.out.println("This won't be executed");
            }else {
                System.out.println("This will be executed
if
                                arguments is not
zero");
            }
        }
        catch (Exception e) {
            System.out.println("Catching exception: " + e);
        }
        finally {
            System.out.println("After trying or catching...");
        }
    }
}

```

- a Try to execute it without arguments. What's the result?
  - b Try to execute it with arguments. What's the result?
  - c What's the conclusion?
4. Modify code in number 3 replace **Exception** in catch block with `NullPointerException`. Compile and run that code, what's the result? What happens if an exception does not have a matching catch?

### 5. Multiple catch

```

class MultipleCatch {
    public static void main(String args[ ]) {
        int n = args.length;
        int a = 1;
        int c[] = new int[1];
        try {
            if(n == 0)

```

```

        a = a / (a - 1);
    else
        c[21] = 5;
    }
    catch (ArithmeticException e) {
        System.out.println("Ups, seem out we have an Arithmetic Exception here...");
    }
    catch (Exception e) {
        System.out.println("Let the other exception handled here...");
    }
}
}

```

- Try to execute the program without any argument. Which catch statement is executed?
- Try to execute the program with an argument. Which catch statement is executed?
- Now replace `ArithmeticException` with `Exception` as shown below:

```

...
}
catch (Exception e) {
    System.out.println("Ups, seem out we have an
                        ArithMetic exception here...");
} catch (ArithmeticException e) {
    System.out.println("Let the other exception handled here...");
}
...

```

**What will happen? Why? Explain.**

## 6. Throwing an exception

```

class ThrowExc {
    public static void main( String args[ ] ) {
        try {
            System.out.println("I'm throwing an exception from
                                here...");
            throw new NullPointerException("throw demo");
        }
        catch(NullPointerException e) {
            System.out.println("I'm catching an exception: [" + e + "]
                                from here...");
        }
    }
}

```

## 7. Re-throwing an exception

```

Class ThrowExc2 {
    static void demoThrow() {
        try {
            System.out.println("I'm throwing an exception from demoThrow
                                method");
            throw new NullPointerException("demo");
        }
    }
}

```

```

        catch(NullPointerException e) {
            System.out.println("I'm catching the exception [" + e + "]
                               from demoThrow method");
            System.out.println("But I'll throw it again...\n");
            throw e;
        }
    }
    public static void main(String args[]) {
        try {
            demoThrow();
        }
        catch(NullPointerException e) {
            System.out.println("I'm catching the exception
                               [" + e + "]\nthrown from demoThrow method");
        }
    }
}

```

- a If exception `e` isn't rethrown from `demoThrow` method, will it still be caught in main function?
- b What can be concluded from the above code?

## 8. Throws demo

```

class ThrowsDemo {
    static void throwOne(){
        System.out.println("I'm throwing a non Runtime exception that I
                           don't handle");
        System.out.println("without specifying in method declaration");
        throw new IllegalAccessException("demo");
    }
    public static void main(String args[]) {
        throwOne();
    }
}

```

- a What will happen if the above code is compiled? Why?
- b What will happen if `IllegalAccessException` is replaced with `ArithmeticException`? Why?

Fix the above code as shown below:

```

class ThrowsDemo {
    static void throwOne() throws IllegalAccessException{
        System.out.println("I'm throwing a non Runtime
                           exception that I don't handle");
        System.out.println("without specifying in method declaration");
        throw new IllegalAccessException("demo");
    }
    public static void main(String args[]) {
        try {
            throwOne();
        }

        catch(IllegalAccessException e){
            System.out.println("Caught " + e);
        }
    }
}

```

```
}  
}  
}
```

**Write your conclusion.**

## 9. Creating your own exception

```
class MyException extends Exception {  
    private int detail;  
    MyException(int a) {  
        detail = a;  
    }  
    public String toString() {  
        return "MyException[" + detail + "];"  
    }  
}  
class ExceptionDemo {  
    public static void main(String args[]) {  
        try {  
            System.out.println("I'm throwing MyException from here...");  
            throw new MyException(221);  
        }  
        catch (MyException e) {  
            System.out.println("I'm catching MyException from here...");  
            System.out.println("MyException detail: " + e.toString());  
        }  
    }  
}
```

## 2 Tugas

Tugas ini dikerjakan perorangan. Anda diharapkan memodifikasi jawaban UTS Praktikum PBO Anda masing-masing dan tambahkanlah event handling pada program Anda tersebut. Hasil modifikasi tersebut harus memiliki paling sedikit satu *try*, *catch*, *finally*, *throw* dan *throws*. Selain itu, tambahkan satu buah *exception* buatan Anda sendiri yang sesuai dengan konteks aplikasi Anda.