# PRAKTIKUM 17

# MENANGANI EXCEPTION

## A. TUJUAN PEMBELAJARAN

1. Memahami mengenai exception
2. Memahami tipe exception yaitu Checked Exception dan Unchecked Exception.
3. Mengetahui cara menggunakan exception menggunakan blok try catch.

## B. DASAR TEORI

Exception adalah suatu kondisi abnormal yang terjadi pada saat menjalankan program. Karena dalam java segala sesuatu merupakan objek, maka exception juga direpresentasikan dalam sebuah objek yang menjelaskan tentang exception tersebut. Contoh exception adalah pembagian bilangan dengan 0, pengisian elemen array diluar ukuran array, kegagalan koneksi database, file yang akan dibuka tidak ada, dan mengakses objek yang belum diinisialisasi.

Terdapat dua penanganan exception yaitu:

- Menangani sendiri exception tersebut.
- Meneruskannya ke luar dengan cara membuat objek tentang exception tersebut dan melemparkannya (throw) keluar agar ditangani oleh kode yang memanggil method(method yang didalamnya terdapat exception) tersebut.

Ada lima keyword yang digunakan oleh Java untuk menangani exception yaitu try, catch, finally, throw dan throws.

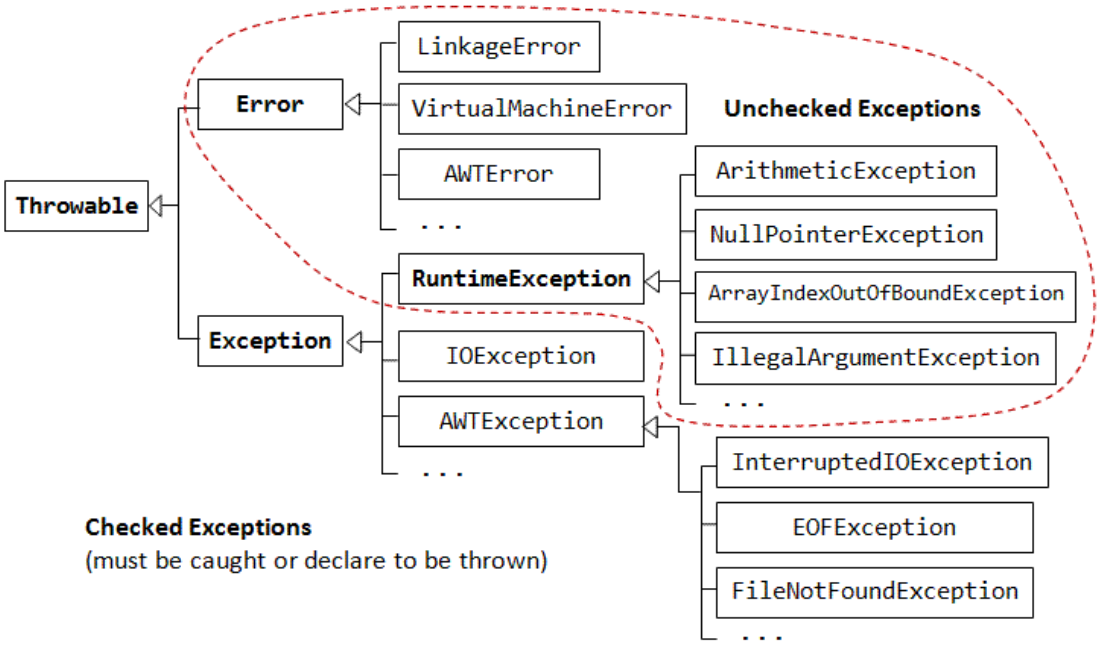
### B.1 Tipe-Tipe Exception

Pada exception, superclass tertinggi adalah class Throwable, tetapi kita hampir tidak pernah menggunakan class ini secara langsung. Dibawah class Throwable terdapat dua subclass yaitu class Error dan class Exception. Class Error merupakan tipe exception yang tidak ditangani dengan blok try-catch, karena berhubungan dengan Java run-time system/environment. Untuk exception dengan tipe class Exception, merupakan exception

yang dapat ditangani oleh program. Terdapat subclass dari class Exception diantaranya RunTimeException, IOException, AWTException dan lain-lain.

Semua exception bertipe RunTimeException dan turunannya tidak harus secara eksplisit ditangani dalam program (Unchecked Exception). Contoh subclass dari RunTimeException adalah ArrayIndexOutOfBoundsException, ArithmeticException, NullPointerException dan lain-lain.

Semua tipe exception yang bukan turunan dari class RuntimeException merupakan exception yang harus ditangani atau jika tidak ditangani menyebab error. Dibawah ini adalah hirarki dari exception.




**B.2 Penggunaan Blok try-catch**

Untuk menangani exception dalam program, dengan meletakkan kode program yang memungkinkan terjadinya exception didalam blok try, diikuti dengan blok catch yang menentukan jenis exception yang ingin ditangani. Contoh :

```
public class Percobaan2 {
    public static void main(String[] args) {
         int a[] = new int[5];
         try{
            a[5] = 100 ;
         }catch(ArrayIndexOutOfBoundsException e){
             System.out.println("Indeks Array melebihi batas");
         }
         System.out.println("Setelah blok try-catch");
    }
```

```
}
```

Output :

```
Terjadi exception karena Indeks Array melebihi batas
Setelah blok try-catch
```

Dapat terjadi kode yang terdapat dalam blok try mengakibatkan lebih dari satu exception. Dalam hal ini, kita dapat menuliskan lebih dari satu blok catch. Contoh :

```
public class Percobaan5 {
    public static void main(String[] args) {
        int bil=10;
        String b[] = {"a","b","c"};
        try{
            System.out.println(bil/0);
            System.out.println(b[3]);
        }catch(ArithmeticException e){
            System.out.println("Error Aritmetik");
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Error Kapasitas Array Melebihi Batas");
        }catch(Exception e){
            System.out.println("Terdapat Error");
        }
    }
}
```

Blok catch yang dijalankan tergantung dengan exception yang terjadi. Java akan menjalankan blok catch yang sesuai dengan tipe exceptionnya saja. Dalam penggunaannya, blok catch dengan tipe subclass harus ditulis lebih dahulu baru diikuti dengan blok catch dengan tipe data superclassnya.

### B.3 Menggunakan Keyword ”finally”

Terdapat kode yang harus dijalankan walaupun terjadi atau tidak terjadi exception, misalkan kita membuka file, hal ini memungkinkan terjadinya exception misal file tidak ada, file tidak bisa dibuka, selanjutnya yang harus dilakukan adalah menutup file tersebut.

```
public class Percobaan2 {
    public static void main(String[] args) {
         int a[] = new int[5];
         try{
            a[5] = 100 ;
         }catch(ArrayIndexOutOfBoundsException e){
```

```
            System.out.println("Terjadi exception karena Indeks Array
melebihi batas");
        }finally{
            System.out.println("Selalu Dijalankan");
        }
        System.out.println("Setelah blok try-catch");
    }
}
```

### B.4 Menggunakan Keyword ”throw ” dan ”throws”

Secara eksplisit, kita dapat melempar (throw) exception dari program menggunakan keyword throw. Jika exception tersebut adalah checked exception, maka pada method harus ditambahkan throws. Jika exception tersebut adalah unchecked exception, maka pada method tidak perlu ditambahkan throws.

```
public class Percobaan6 {
    public static void method1() throws FileNotFoundException{
        throw new FileNotFoundException("File Tidak Ada");
    }
    public static void main(String[] args) {
        try {
            method1();
        } catch (FileNotFoundException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

### B.5 Membuat sendiri Subclass dari Exception

Untuk melakukan ini, kita cukup membuat class baru yang merupakan subclass Exception. Didalam class ini kita cukup mendeklarasikan konstruktor.

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

## C. TUGAS PENDAHULUAN

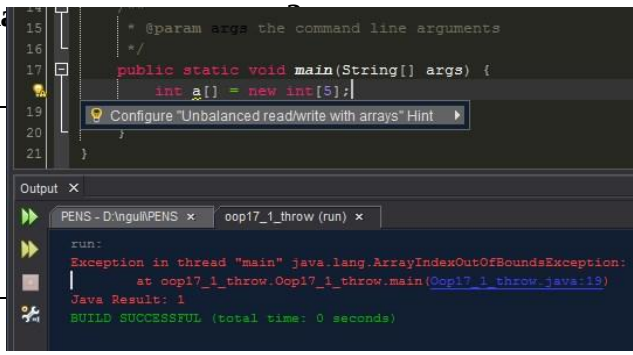
Buatlah review mengenai definisi exception, jenis exception, dan berikan 2 contoh program yang menyebabkan exception beserta cara penangannya.

## D. PERCOBAAN

**Percobaan 1 : Jalankan program di bawah ini ! Bagaima**

**Jelaskan !**

```
public class Percobaan1 {
    public static void main(String[] args) {
        int a[] = new int[5];
        a[5] = 100 ;
    }
}
```




**Percobaan 2 : Memahami cara menangkap Exception dengan tipe ArrayIndexOutOfBoundsException**

```
public class Percobaan2 {
    public static void main(String[] args) {
         int a[] = new int[5];
         try{
            a[5] = 100 ;
         }catch(ArrayIndexOutOfBoundsException e){
             System.out.println("Indeks Array melebihi batas");
         }
    }
}
```

**Percobaan 3 : Jalankan percobaan 3, bagaimana output program ? Perbaiki dengan Percobaan32 untuk menangani exception.**

```
public class Percobaan3 {
    public static void main(String[] args) {
        int bil = 10 ;
        System.out.println(bil/0);
    }
}
```

```
public class Percobaan32 {
     public static void main(String[] args) {
        int bil = 10 ;
        try{
            System.out.println(bil/0);
        }catch(ArithmeticException e){
            System.out.println("Tidak boleh membagi bilangan dengan
0");
        }
    }
}
```

**Percobaan 4 : Memahami try bertingkat.**

```
public class Percobaan4 {
```

```
    public static void main(String[] args) {
        int bil = 10 ;
        try{
            System.out.println(bil/0);
        }catch(ArithmeticException e){
            System.out.println("Terjadi  exception  karena  tidak  boleh
membagi bilangan dengan 0");
        }catch(Exception e){
            System.out.println("Terdapat Error");
        }
    }
}
```

**Percobaan 5 : Bandingkan output dua program di bawah ini ! Jelaskan !**

```
public class Percobaan5 {
    public static void main(String[] args) {
        int bil=10;
        String b[] = {"a","b","c"};
        try{
            System.out.println(bil/0);
            System.out.println(b[3]);
        }catch(ArithmeticException e){
            System.out.println("Error Aritmetik");
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Error Kapasitas Array Melebihi Batas");
        }catch(Exception e){
            System.out.println("Terdapat Error");
        }
    }
}
```

```
public class Percobaan52 {
    public static void main(String[] args) {
        int bil=10;
        String b[] = {"a","b","c"};
        try{
            System.out.println(b[3]);
            System.out.println(bil/0);
        }catch(ArithmeticException e){
            System.out.println("Error Aritmetik");
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Error Kapasitas Array Melebihi Batas");
        }catch(Exception e){
            System.out.println("Terdapat Error");
        }
    }
}
```

**Percobaan 6 : Penggunaan finally**

```
public class ExcepTest{
   public static void main(String args[]){
```

```
      int a[] = new int[2];
      try{
         System.out.println("Access element three :" + a[3]);
      }catch(ArrayIndexOutOfBoundsException e){
         System.out.println("Exception thrown  :" + e);
      }
      finally{
         a[0] = 6;
         System.out.println("First element value: " +a[0]);
         System.out.println("The finally statement is executed");
      }
   }
}
```

## E. LATIHAN

1. Semua exception yang berasal dari java.lang.RuntimeException adalah *unchecked exceptions*, sedangkan exception lainnya yang tidak berasal dari java.lang.RuntimeException adalah *checked exceptions*. Jelaskan mengenai *unchecked exceptions* dan *checked exceptions* ,berikan contoh !
2. Buatlah contoh program untuk menangani exception dengan cara menangkap exception seperti di bawah ini :
   - ArithmeticException
   - ArrayStoreException
   - ClassCastException
   - ArrayIndexOutOfBoundsException
   - StringIndexOutOfBoundsException
   - NegativeArraySizeException
   - NoSuchElementException
   - NullPointerException,
   - NumberFormatException

## F. TUGAS

1. Terdapat dua cara untuk menangani Exception yaitu dengan menangkap Exception dan melempar Exception. Lakukan penanganan exception dengan menangkap Exception menggunakan blok try-catch. Berilah penjelasan (apakah program termasuk unchecked exceptions atau checked exceptions) !

```
public class ReadFile {
     public static void main(String args[]){
        File file = new File("Data.txt");
        BufferedReader fileReader ;
```

```
         fileReader = new BufferedReader(new FileReader(file));
         while(true){
            String line = fileReader.readLine();
            if (line == null)
               break ;
            System.out.println(line);
         }
      }
}
```

## G. LAPORAN RESMI

Kerjakan hasil percobaan(D), latihan(E) dan tugas(F) di atas dan tambahkan analisa.