Penanganan Kesalahan

Tujuan

- Mengerti Exception Handling dan mampu mengimplementasikan **dalam** bahasa pemrograman Java.

Pendahuluan

- Kesalahan sering terjadi pada saat perancangan dan implementasi
- Kesalahan dikategorikan :
  - syntak error menyebabkan kesalahan kompilasi
  - Semantic error , program menghasilkan keluaran yang tidak sesuai dengan harapan
  - Run-time error, kebanyakan mengakibatkan terminasi program secara tidak normal atau bahkan sistem crash. Misal : penggunaan tipe data yang salah.

Error Handling

- Setiap program yang berada dalam suatu kondisi yang tidak normal – Error Conditions.
- Program yang ‘baik’ harus dapat menangani kondisi ini.
- Java menyediakan suatu mekanisme untuk menangani kondisi ini - *exceptions*

Exception

- Exception merupakan suatu keadaan yang disebabkan oleh runtime error **dalam** program.
- Memungkinkan **kesalahan** ditangani tanpa harus ‘mengotori’ program (dengan rutin yang menangani **kesalahan**)
- Memungkinkan pemisahan **penanganan kesalahan** dengan program utama

Contoh: Pembagian bil dengan nol

```
public class soal {
    public static void main(String[]args)
    {
    System.out.println("Sebelum Pembagian");
    System.out.println(5/0);
    System.out.println("Setelah Pembagian");
     }
}
```

Hasil Running

```
D:\Handout dosen\PBO\latihan>java soal
Sebelum Pembagian
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at soal.main(soal.java:5)
```

Pernyataan try

- Pernyataan try digunakan utk keperluan exception.
- Bentuk 1:

```
try {
        //Blok yang akan ditangkap sekiranya terjadi exception
}
catch(parameter)
        //Blok yang akan dijalankan kalau terjadi exception
}
```

Contoh :

```
public class soal {
    public static void main(String[]args)
    {
    System.out.println("Sebelum Pembagian");
    try{
            System.out.println(5/0);
    }
    catch (Throwable t) {
            System.err.println("Terjadi Pembagian dengan nol");
            System.err.println(t.getMessage());
    }
    System.out.println("Setelah Pembagian");
    }
}
```

//Throwable – nama kelas yg digunakan utk menangani exception.

Pernyataan try

- Bentuk 2 :

```
try{
        //blok yang akan ditangkap sekiranya terjadi
          exception
}
finally
        //blok yang akan dijalankan terakhir kali
}
```

Pernyataan try

finally selalu dijalankan baik sewaktu terjadi exception maupun sewaktu tidak terjadi exception.

Contoh :

```
public class soal {
    public static void main(String[]args)
    {
    double bilangan = 100.0;
    System.out.println("Sebelum
    pembagian");
    for (int i=5; i>=0; i--){
        try{
            System.out.print(bilangan+
            "/"+i+"=");
            System.out.println((bilangan/i));
            }
            finally{
            System.out.println("Bagian
                    finally dijalankan");
            }
    }
    System.out.println("selesai");
    }
}
```

Hasil Running

```
D:\Handout dosen\PBO\latihan>java soal
Sebelum pembagian
100.0/5=20.0
Bagian finally dijalankan
100.0/4=25.0
Bagian finally dijalankan
100.0/3=33.33333333333336
Bagian finally dijalankan
100.0/2=50.0
Bagian finally dijalankan
100.0/1=100.0
Bagian finally dijalankan
100.0/0=Infinity
Bagian finally dijalankan
selesai
```

Catch secara bertingkat

- Kelas Throwable memiliki sub kelas yaitu:

  - Error digunakan utk menangani kesalahan spt memori habis (OutOfMemoryError) dan stack habis (StackOverFlowError).

  - Exception memiliki subkelas RuntimeException yg digunakan utk array tidak valid (IndexOutOfBoundsException) dan kesalahan aritmatika (ArithmeticException).

Catch secara bertingkat

```
try{
    //blok yg akan ditangkap sekiranya terjadi exception
}
catch(RuntimeException r){
    //blok yg akan dijalankan kalau terjadi eksepsi
    RuntimeError
}
catch(Exception e){
    //blok yg akan dijalankan kalau terjadi eksepsi Exception
}
catch(Throwable t){
    //blok yg akan dijalankan kalau terjadi eksepsi yg lain
}
```

Contoh:

```
public class soal {
    public static void main(String[]args)
    {
    System.out.println("Sebelum Pembagian");
    try{
            System.out.println(5/0);
    }
    catch (RuntimeException r){
            System.err.println("Runtime exception");
    }
```

Contoh :

```
catch (Exception e){
        System.err.println("Exception");
    }
    catch (Throwable t) {
        System.err.println("Terjadi Pembagian dengan
    nol");
        System.err.println(t.getMessage());
    }
    System.out.println("Setelah Pembagian");
    }
}
```

Hasil :

```
D:\Handout dosen\PBO\latihan>java soal
Sebelum Pembagian
Runtime exception
Setelah Pembagian
```

Melontarkan Exception

- Bentuk :

  throw variabelobjek;

- Variabelobjek merujuk ke suatu kelas eksepsi.

Contoh :

```
public class soal {
    public static void main(String[]args)
    {
        int[] larik = new int[10];
        try{
            larik[50] = 77;
            System.out.println(larik[50]);
        }
        catch (ArrayIndexOutOfBoundsException a) {
            a = new ArrayIndexOutOfBoundsException ("array harus
                berkisar antara 0 dan 9");
            throw(a);
        }
    }
}
```