MODUL PRAKTIKUM

Exception Dan Assertions

Versi 1.3

1. Tujuan

- Menangani exception dengan menggunakan try, catch dan finally
- Membedakan penggunaan antara throw dengan throws
- Menggunakan exception class yang berbeda – beda
- Membedakan antara checked exceptions dan unchecked exceptions
- Membuat exception class tersendiri
- Menjelaskan keunggulan penggunaan assertions
- Menggunakan assertions

2. Latar Belakang

Bugs dan error dalam sebuah program sangat sering muncul meskipun program tersebut dibuat oleh programmer berkemampuan tinggi. Untuk menghindari pemborosan waktu pada proses error-checking, Java menyediakan mekanisme penanganan exception.

Exception adalah singkatan dari Exceptional Events. Kesalahan (errors) yang terjadi saat runtime, menyebabkan gangguan pada alur eksekusi program. Terdapat beberapa tipe error yang dapat muncul. Sebagai contoh adalah error pembagian 0, mengakses elemen di luar jangkauan sebuah array, input yang tidak benar dan membuka file yang tidak ada.

Seluruh exceptions adalah subclasses, baik secara langsung maupun tidak langsung, dari sebuah root class *Throwable*. Kemudian, dalam class ini terdapat dua kategori umum : Error class dan Exception class.

Exception class menunjukkan kondisi yang dapat diterima oleh user program. Umumnya hal tersebut disebabkan oleh beberapa kesalahan pada kode program. Contoh dari exceptions adalah pembagian oleh 0 dan error di luar jangkauan array.

Error class digunakan oleh Java run-time untuk menangani error yang muncul pada saat dijalankan. Secara umum hal ini di luar control user karena kemunculannya disebabkan oleh run-time environment. Sebagai contoh adalah *out of memory* dan *harddisk crash.*

3. Percobaan

Percobaan 1 : Menangkap Exception

```
public class DivideByZero {
    public DivideByZero() {}
    public static void main(String args[]) {
        try {
            System.out.println(3/0);
            System.out.println("Please print me.");
        } catch (ArithmeticException exc) {
            System.out.println(exc);
        }
        System.out.println("After exception.");
    }
}
```

Percobaan 2 : Multiple Catch

```
public class MultipleCatch {
    public MultipleCatch() {}
    public static void main(String args[]) {
        try {
            int den = Integer.parseInt(args[0]);     //line 4
            System.out.println(3/den);       //line 5
        } catch (ArithmeticException exc) {
            System.out.println("Divisor was 0.");
        } catch (ArrayIndexOutOfBoundsException exc2) {
            System.out.println("Missing argument.");
        }
        System.out.println("After exception.");
    }
}
```

Berikan argument :

- Tidak Ada argument,
- 1
- 0

Percobaan 3 : Nested Try

```
public class NestedTryDemo {
    public NestedTryDemo() {}
    public static void main(String args[]){
        try {
            int a = Integer.parseInt(args[0]);
            try {
                int b = Integer.parseInt(args[1]);
                System.out.println(a/b);
            } catch (ArithmeticException e) {
            System.out.println("Divide by zero error!");
            }
        } catch (ArrayIndexOutOfBoundsException exp2) {
            System.out.println("2 parameters are required!");
        }
    }
}
```

Berikan argument :

- **Tidak Ada argument,**
- **15**
- **15 3**
- **15 0**

Percobaan 4 : Nested Try dengan Method

```
public class NestedTryDemo2 {
    public NestedTryDemo2() {}
    static void nestedTry(String args[]) {
        try {
            int a = Integer.parseInt(args[0]);
            int b = Integer.parseInt(args[1]);
            System.out.println(a/b);
        }catch (ArithmeticException e) {
            System.out.println("Divide by zero error!");
        }
    }
    public static void main(String args[]){
        try {
            nestedTry(args);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("2 parameters are required!");
        }
    }
}
```

Berikan argument :

- **Tidak Ada argument,**
- **15**
- **15 3**
- **15 0**

```
public class FinallyDemo {
    public FinallyDemo() {}
    static void myMethod(int n) throws Exception{
        try {
            switch(n) {
            case 1: System.out.println("first case");
                    return;
            case 3: System.out.println("third case");
                    throw new RuntimeException("third case demo");
            case 4: System.out.println("fourth case");
                    throw new Exception("fourth case demo");
            case 2: System.out.println("second case");
            }
        } catch (RuntimeException e) {
            System.out.print("RuntimeException caught: ");
            System.out.println(e.getMessage());
        } finally {
            System.out.println("try-block is entered.");
        }
    }
    public static void main(String args[]){
        for (int i=1; i<=4; i++) {
            try {
                FinallyDemo.myMethod(i);
            } catch (Exception e){
                System.out.print("Exception caught: ");
                System.out.println(e.getMessage());
            }
            System.out.println();
        }
    }
}
```

Percobaan 5 : Demo Keyword Finally

Percobaan 6 : Demo Keyword throw

```
public class ThrowDemo {
    public ThrowDemo() {}
    public static void main(String args[]){
        String input = "invalid input";
            try {
                if (input.equals("invalid input")) {
                    throw new RuntimeException("throw demo");
                } else {
                    System.out.println(input);
                }
                System.out.println("After throwing");
            } catch (RuntimeException e) {
                System.out.println("Exception caught here.");
                System.out.println(e);
            }
    }
}
```

Percobaan 7 : Demo Keyword throws

```
class ThrowingClass {
    static void myMethod() throws ClassNotFoundException {
        throw new ClassNotFoundException("just a demo");
    }
}

public class ThrowsDemo {
    public ThrowsDemo() {}
    public static void main(String args[]) {
        try {
            ThrowingClass.myMethod();
        } catch (ClassNotFoundException e) {
            System.out.println(e);
        }
    }
}
```

Percobaan 8 : Multiplecatch Exception ERROR

```
public class MultipleCatchError {
    public MultipleCatchError() {}
    public static void main(String args[]){
        try {
            int a = Integer.parseInt(args [0]);
            int b = Integer.parseInt(args [1]);
            System.out.println(a/b);
        } catch (Exception e) {
            System.out.println(e);
        } catch (ArrayIndexOutOfBoundsException e2) {
            System.out.println(e2);
        }
        System.out.println("After try-catch-catch.");
    }
}
```

Percobaan 9 : User Defined Exception

```
class HateStringException extends RuntimeException{}
public class TestHateString {
    public TestHateString() {}
    public static void main(String args[]) {
        String input = "invalid input";
        try {
            if (input.equals("invalid input")) {
            throw new HateStringException();
            }
            System.out.println("String accepted.");
        } catch (HateStringException e) {
            System.out.println("I hate this string: " + input +".");
        }
    }
}
```

Percobaan 10 : Assertions

```
public class AgeAssert {
    public AgeAssert() {}
    public static void main(String args[]) {
        int age = Integer.parseInt(args[0]);
        assert(age>0);
        if (age >= 18) {
            System.out.println("Congrats! You're an adult! =)");
        }
    }
}
```

Perintah Kompilasi dan eksekusi :

javac –source 1.4 AgeAssert.java
java –enableassertions AgeAssert 'arguments'

4. Latihan

4.1 Heksadesimal ke Desimal

Tentukan sebuah angka heksadesimal sebagai input. Konversi angka tersebut menjadi bilangan desimal. Tentukan exception class Anda sendiri dan lakukan penanganan jika input dari user bukan berupa bilangan heksadesimal.

4.2 Menampilkan Sebuah Berlian

Tentukan nilai integer positif sebagai input. Tampilkan sebuah berlian menggunaakan karakter asterisk (*) sesuai angka yang diinput oleh user. Jika user memasukkan bilangan integer negatif, gunakan assertions untuk menanganinya.

Sebagai contoh, jika user memasukkan integer bernilai 3, program Anda harus menampilkan sebuah berlian sesuai bentuk berikut :

```
  *
 ***
*****
 ***
  *
```