

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN KE-12



Disusun Oleh :

NAMA : Raden Isnawan Argi Aryasatya

NIM : 195410257

JURUSAN : Teknik Informatika

JENJANG : S1

KELAS : TI-5

Laboratorium Terpadu
Sekolah Tinggi Manajemen Informatika Komputer
AKAKOM
YOGYAKARTA

2021

PERTEMUAN KE-12 **(EXCEPTION HANDLING)**

TUJUAN

Dapat menggunakan perintah yang menangani Exception pada program, menggunakan throws untuk melempar eksepsi dan membuat Exception sendiri

DASAR TEORI

Dalam pembuatan program seringkali dijumpai error atau kesalahan. Oleh karena itu, diperlukan suatu mekanisme yang membantu menangani error atau kesalahan yang terjadi, baik saat pembuatan maupun implementasi program. Java menyediakan mekanisme dalam pemrograman untuk menangani hal-hal tersebut yang disebut dengan exception. Exception adalah event yang terjadi ketika program menemui kesalahan pada saat instruksi program dijalankan. Banyak hal yang dapat menimbulkan event ini, misalnya crash, harddisk rusak dengan tiba-tiba, sehingga program-program tidak bisa mengakses file-file tertentu. Programmer pun dapat menimbulkan event ini, misalnya dengan melakukan pembagian dengan bilangan nol, atau pengisian elemen array melebihi jumlah elemen array yang dialokasikan dan sebagainya.

PRAKTIK 1: Membuat kelas tanpa exception

```
BagiNol.java x
public class BagiNol {
    public static void main(String[] args) {
        System.out.println("Sebelum pembagian");
        System.out.println(5/0);
        System.out.println("Sesudah pembagian");
    }
}
```

output:

```
C:\Windows\system32\cmd.exe
Sebelum pembagian
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at BagiNol.main(BagiNol.java:4)
Press any key to continue . . . _
```

penjelasan:

pada contoh program di atas, kita akan membuat program sederhana dimana program tersebut sengaja kita buat menjadi error atau terjadi kesalahan yaitu dengan membagi suatu bilangan dengan angka 0. Jenis Exception yang tampil pada program tersebut adalah ArithmeticException, error tersebut terjadi karena ada pembagian 0 (nol). Jadi, pada intinya Baris ke-5 tidak akan dieksekusi karena ada kesalahan pembagian dengan bilangan nol pada baris ke-4.

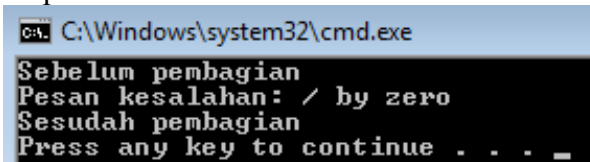
PRAKTIK 2: Membuat kelas menggunakan exception

```
BagiNol2.java x BagiNol.java
public class BagiNol2 {
    public static void main(String[] args) {
        System.out.println("Sebelum pembagian");

        try { System.out.println(5/0); }
        catch (Exception t) {
            System.out.print("Pesan kesalahan: ");
            System.out.println(t.getMessage());
        }

        System.out.println("Sesudah pembagian");
    }
}
```

output:



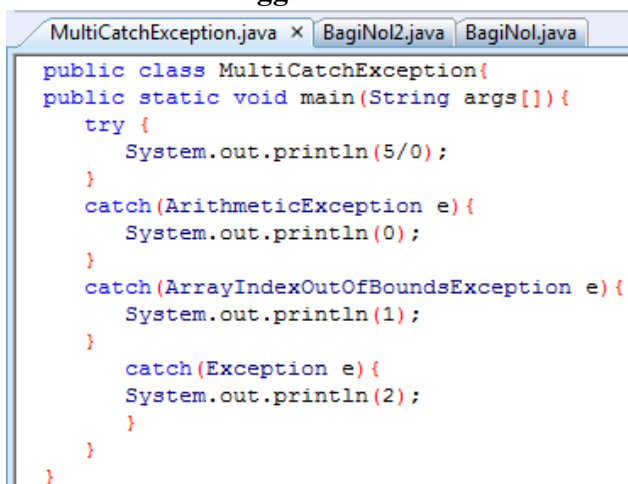
```
C:\Windows\system32\cmd.exe
Sebelum pembagian
Pesan kesalahan: / by zero
Sesudah pembagian
Press any key to continue . . . _
```

penjelasan:

program kedua ini adalah solusi untuk program pertama tadi. Untuk mengatasi masalah seperti di program praktik 1 tadi, kita dapat menggunakan statement try-catch. Setiap pernyataan yang dapat mengakibatkan exception harus berada didalam try, karena untuk menangani dimana munculnya kesalahan yang ingin di proses. Jadi try adalah tempat menuliskan statement yang akan di uji apakah ada kesalahan atau tidak. Sementara catch digunakan untuk menangani berbagai jenis exception, kesalahan yang muncul akan dianggap sebagai objek. Jadi catch bisa diartikan sebagai tempat menuliskan aksi program bila sebuah exception terjadi.

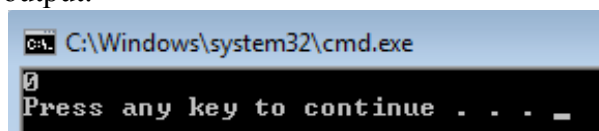
Untuk pembuatan programnya, kita mulai dengan menuliskan **public static void main(String[] args)** sebagai method untuk memulai sebuah eksekusi program. Langsung saja kita cetak kalimat dengan menuliskan baris kode **System.out.println("Sebelum pembagian");** sebagai kalimat awal pada program yang kita buat. Di bawahnya kita deklarasikan keyword try untuk mencoba kesalahan yang disengaja dengan menuliskan **try { System.out.println(5/0); }**. Kemudian kesalahan yang kita try tadi langsung saja kita catch dengan mendeklarasikan t sebagai exception. Kita lakukan itu dengan menuliskan baris kode **catch (Exception t)**. Setelah berhasil menggunakan catch, program akan mengeluarkan kalimat "Pesan kesalahan:" yang diikuti dengan t.getMessage (keterangan spesifik penyebab kesalahan). Lalu program kita akhiri dengan **System.out.println("Sesudah pembagian");**.

PRAKTIK 3: Menggunakan lebih dari satu statement catch



```
MultiCatchException.java x  BagiNol2.java  BagiNol.java
public class MultiCatchException{
public static void main(String args[]){
    try {
        System.out.println(5/0);
    }
    catch(ArithmeticException e){
        System.out.println(0);
    }
    catch(ArrayIndexOutOfBoundsException e){
        System.out.println(1);
    }
    catch(Exception e){
        System.out.println(2);
    }
}
```

output:



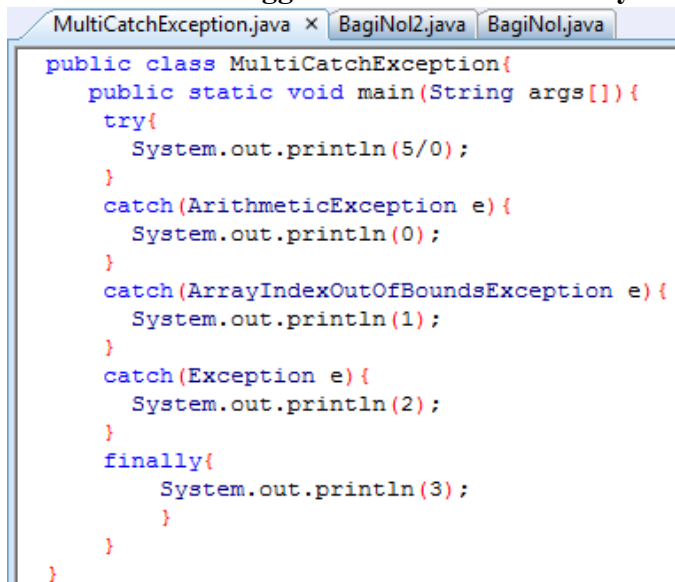
```
C:\Windows\system32\cmd.exe
Press any key to continue . . . _
```

penjelasan:

Kita dapat menggunakan catch lebih dari satu, untuk menangkap jenis exception yang berbeda pada pernyataan didalam try. Pada contoh berikut ini, kita akan menggabungkan kedua program yang sebelumnya sudah kita buat menjadi satu, didalam program tersebut kita akan menangkap 3 jenis exception yang berbeda, yaitu ArithmeticException, ArrayIndexOutOfBoundsException, dan

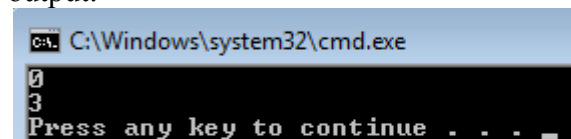
Exception. Pada program ini, kita mendeklarasikan 3 catch untuk menentukan catch mana yang akan digunakan untuk mengatasi kesalahan pada try {System.out.println(5/0);}. Maka dari itu, output adalah angka nol yang artinya pada blok try terdapat kesalahan arithmetik maka kesalahan tersebut ditangkap oleh blok catch dari kelas ArithmeticException, sementara catch lain diabaikan karena kita sudah menggunakan catch pertama.

PRAKTIK 4: Menggunakan statement finally



```
public class MultiCatchException{
    public static void main(String args[]){
        try{
            System.out.println(5/0);
        }
        catch(ArithmeticException e){
            System.out.println(0);
        }
        catch(ArrayIndexOutOfBoundsException e){
            System.out.println(1);
        }
        catch(Exception e){
            System.out.println(2);
        }
        finally{
            System.out.println(3);
        }
    }
}
```

output:

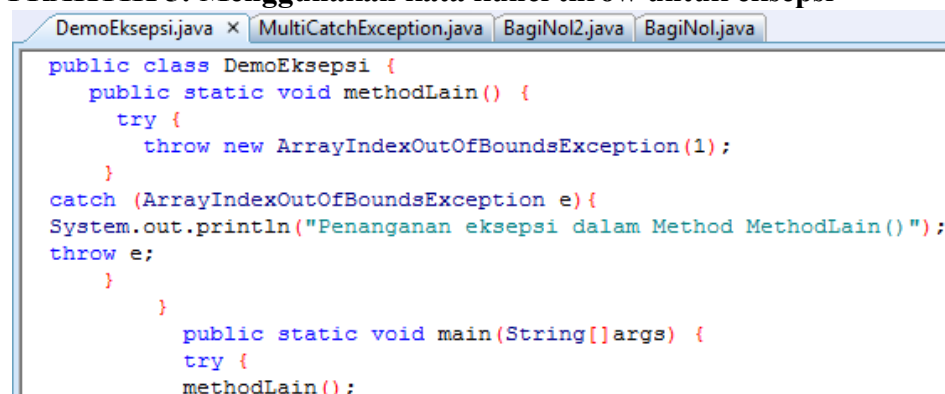


```
C:\Windows\system32\cmd.exe
0
3
Press any key to continue . . .
```

penjelasan:

Seperti tadi, kita membuat 3 catch yaitu ArithmeticException, ArrayIndexOutOfBoundsException, dan Exception. Perbedaannya adalah, di akhir program kita menambahkan statement finally. Statement finally digunakan untuk mengeksekusi kode program jika terjadi exception atau tidak terjadi exception, jadi blok kode didalamnya akan terus di eksekusi pada kondisi apapun. Jadi, output 0 dari catch ArithmeticException tetap muncul, tetapi diikuti oleh output milik finally yaitu 3. Intinya, output program adalah angka nol dan tiga, artinya setelah salah satu blok catch dieksekusi maka alur program selanjutnya adalah mengeksekusi bagian blok finally.

PRAKTIK 5: Menggunakan kata kunci throw untuk eksepsi



```
public class DemoEksepsi {
    public static void methodLain() {
        try {
            throw new ArrayIndexOutOfBoundsException(1);
        }
        catch (ArrayIndexOutOfBoundsException e){
            System.out.println("Penanganan eksepsi dalam Method MethodLain()");
            throw e;
        }
    }

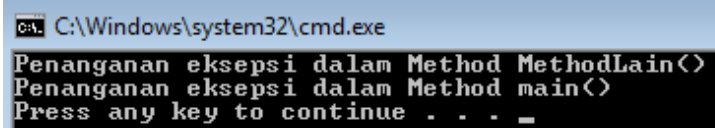
    public static void main(String[] args) {
        try {
            methodLain();
        }
    }
}
```

```

    }
    catch (ArrayIndexOutOfBoundsException e){
        System.out.println("Penanganan eksepsi dalam Method main()");
    }
}

```

output:



```

C:\Windows\system32\cmd.exe
Penanganan eksepsi dalam Method MethodLain()
Penanganan eksepsi dalam Method main()
Press any key to continue . . . _

```

penjelasan:

Keyword throw digunakan untuk melempar exception atau bug yang dibuat secara manual oleh kita, yaitu misalnya saat kita membuat sebuah program dan user salah memasukan input, tetapi program tidak menandakan bahwa itu terjadi error karena input yang dimasukan tidak berpotensi akan terjadi error. Disinilah keyword throw akan berperan. Dengan menggunakan kata kunci throw, kita dapat melempar exception pada kondisi yang telah kita tentukan.

Pada program di atas, exception/error yang telah di tangkap oleh blok try (**throw new ArrayIndexOutOfBoundsException(1)**) akan diubah menjadi objek (e) pada statement catch lalu menampilkan pesan error tersebut menggunakan method methodLain(). Sehingga yang muncul pada output adalah kalimat milik catch pertama yaitu System.out.println ("Penanganan eksepsi dalam Method MethodLain()"); dan catch kedua di dalam main method yaitu System.out.println ("Penanganan eksepsi dalam Method main()");

KESIMPULAN

Pada pertemuan kali ini, inti dari laporan yang kita kerjakan adalah menerapkan keyword-keyword dalam konsep exception handling di java yaitu try, catch, finally, throw, dan throws. Dari pengerjaan program yang sudah kita buat, saya bisa menyimpulkan bahwa penyebab exception ada 3 yaitu:

- Kesalahan kode atau data. Contohnya jika kode berusaha mengakses suatu indeks dari array yang di luar batas array.
- Metode standar exception. Contohnya, metode substring() dalam kelas String, dapat memunculkan pesan dalam bentuk StringIndexOutOfBoundsException.
- Kesalahan Java. Hal ini merupakan kesalahan dalam mengeksekusi Java Virtual Machine yang dijalankan pada saat kompilasi.

Pada praktik terakhir, kita mempraktekkan throw yaitu keyword digunakan untuk melempar exception atau bug pada program yang kita buat.

Terima Kasih