

PERTEMUAN 18

EXCEPTION HANDLING

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami konsep dari *Exception Handling*
2. Mahasiswa dapat membedakan *Exception Handling* dan *Error*,
3. Mahasiswa dapat mengetahui Jenis jenis *Exception Handling*
4. Mahasiswa mengetahui *keyword Exception Handling*
5. Mahasiswa dapat mempraktekan penggunaan *Exception*

B. URAIAN MATERI

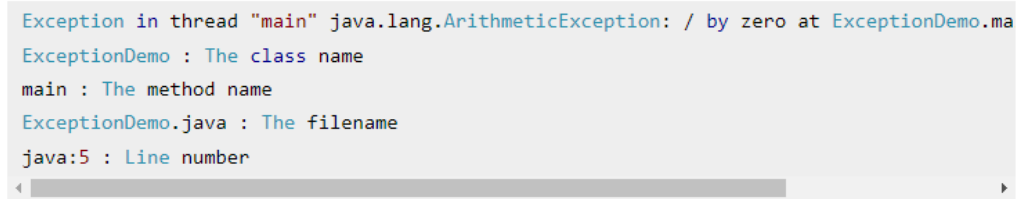
Exception Handling salah satu fitur terpenting dari pemrograman java yang memungkinkan kita untuk menangani kesalahan runtime yang disebabkan oleh *exception*. *Exception Handling* di Java adalah salah satu mekanisme yang ampuh sehingga program aplikasi dapat di normalkan dan dipertahankan.

Apa itu *exception*? adalah peristiwa yang tidak diinginkan yang mengganggu aliran normal program. Ketika *exception* terjadi, eksekusi program dihentikan. Dalam kasus seperti itu, kita mendapatkan pesan kesalahan yang dihasilkan sistem. Hal yang baik tentang *exception* adalah bahwa dapat ditangani (*handling*). Dengan *Exception Handling*, kita dapat memberikan pesan yang bermakna kepada pengguna tentang masalah tersebut daripada pesan yang dihasilkan sistem, yang mungkin tidak dapat dimengerti oleh pengguna.

Mengapa terjadi *exception*? ada beberapa alasan yang dapat menyebabkan program mengeluarkan *exception*. Misalnya: Membuka file yang tidak ada di program kita, kesalahan input yang diberikan oleh pengguna, mengakses array yang melebihi batas indeks dll.

1. Kelebihan menggunakan *Exception handling*

Jika terjadi *Exception*, yang belum ditangani oleh programmer maka eksekusi program akan dihentikan dan pesan kesalahan yang dihasilkan sistem akan ditampilkan kepada *user*. Contoh *exception* yang dihasilkan sistem di bawah ini:

A screenshot of a Java console window showing an exception message. The text is as follows:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero at ExceptionDemo.ma
ExceptionDemo : The class name
main : The method name
ExceptionDemo.java : The filename
java:5 : Line number
```

Gambar 17. 5 Contoh pesan *Excetion*

Pesan ini tidak di mengerti oleh user sehingga users tidak akan dapat memahami apa yang salah. Untuk memberi tahu kesalahan pada contoh program sederhana ini, kita dapat melakukan *exception handling*. kita menangani kondisi seperti itu dan kemudian menampilkan pesan peringatan kepada pengguna, yang memungkinkan mereka memperbaiki kesalahan karena sebagian besar waktu exception terjadi karena data yang salah yang diberikan oleh pengguna.

Exception handling memastikan bahwa aliran program tidak terputus saat eksepsi terjadi. Misalnya, jika sebuah program memiliki banyak pernyataan dan eksepsi terjadi di tengah jalan setelah menjalankan pernyataan tertentu, maka pernyataan setelah eksepsi tidak akan dijalankan dan program akan berhenti secara tiba-tiba.

Dengan exception kita pastikan bahwa semua pernyataan dijalankan dan aliran program tidak terputus.

2. Perbedaan error dan exception

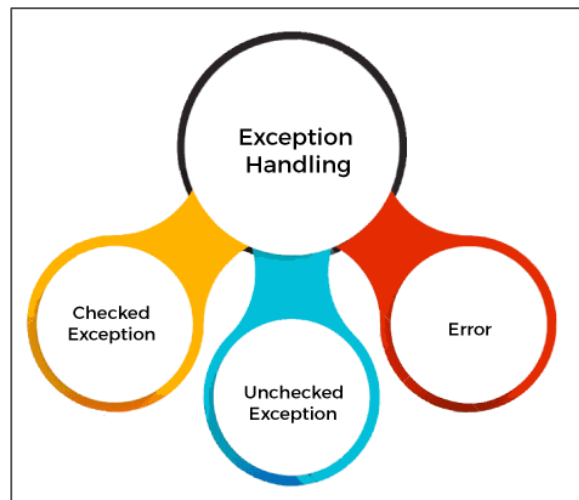
Error menunjukkan bahwa suatu kesalahan berupa operasi yang *illegal*. Kesalahan penulisan *keyword* perintah, kesalahan logical kode program Kesalahan operasi ini menyebabkan masalah pada kode program, Selama menulis kode program, kemunculan *error* sangat sulit dikenali.

Exception adalah peristiwa yang terjadi dalam kode. Seorang programmer dapat menangani kondisi seperti itu dan mengambil tindakan korektif yang diperlukan. Exception dapat ditangani oleh blok try-catch sehingga dapat membuat program tetap berjalan normal jika exception ini muncul. Sementara memulihkan *Error* adalah hal yang sangat tidak mungkin. Satu-satunya cara yang dilakukan adalah menghentikan program

3. Jenis jenis dari exception

Pada umumnya terdapat dua jenis *exception* pada bahasa Java namun menurut oracle terdapat 3 jenis exception yaitu :

- a. *checked exception*
- b. *unchecked exception*
- c. *error*



Gambar 18. 1 Jenis Jenis *Exception*

a. *Checked exception*

Merupakan eksepsi yang dikenal dan diketahui pada saat program di *compile class* yang secara langsung mewarisi adalah class *Throwable* misalnya, *IOException*, *SQLException*, dll.

b. *Unchecked Exception*

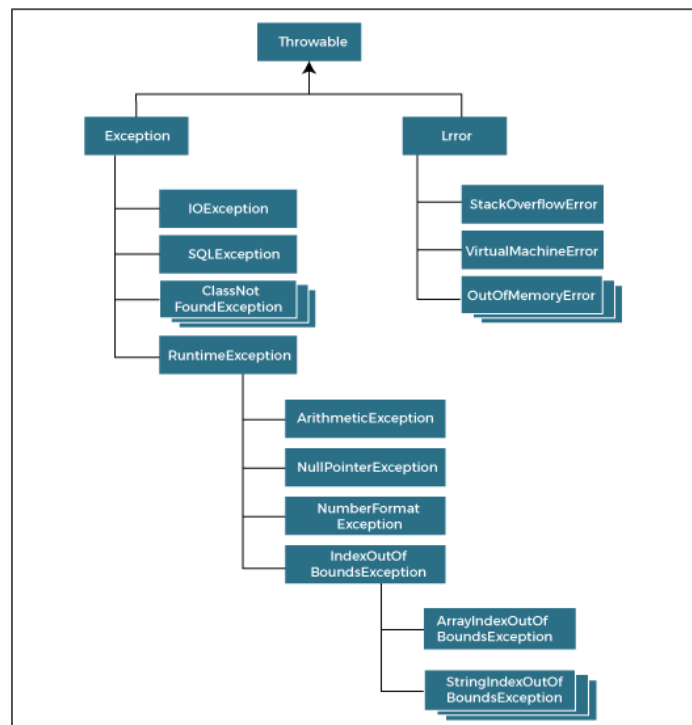
Merupakan class yang mewarisi *RuntimeException* dikenal Misalnya, *ArithmeticException*, *NullPointerException*, *ArrayIndexOutOfBoundsException*, dll. *Unchecked Exception* tidak diperiksa pada waktu kompilasi, tetapi diperiksa saat runtime.

c. *Error*

Kesalahan yang tidak dapat dipulihkan. Beberapa contoh kesalahan adalah *OutOfMemoryError*, *VirtualMachineError*, *AssertionError* dll.

4. Hirarki dari class exception

Class `java.lang.Throwable` adalah kelas akar dari hierarki exception Java yang diwarisi oleh dua subkelas: `checked` dan `error`. Hirarki class exception Java diberikan di bawah ini:



Gambar 18. 2 Hirarki *Exception*

5. Keyword Java exception

Java menyediakan lima kata kunci yang digunakan untuk menangani *exception*. Tabel berikut menjelaskan masing-masing.

Tabel 18. 1 Keyword pada exception

Keyword	Deskripsi
try	Kata kunci " <i>Try</i> " digunakan untuk menentukan blok di mana kita harus menempatkan kode exception . Artinya kita tidak bisa menggunakan blok <i>try</i> saja. Blok <i>try</i> harus diikuti oleh <i>catch</i> atau <i>last</i> .
catch	Blok " <i>catch</i> " digunakan untuk menangani <i>exception</i> harus didahului dengan blok <i>try</i> yang artinya kita tidak bisa menggunakan blok <i>catch</i> saja. bisa diikuti dengan keyword <i>finally</i> setelahnya
finally	Blok " <i>finally</i> " digunakan untuk mengeksekusi kode program yang diperlukan. dieksekusi apakah exception ditangani atau tidak.
throw	Kata kunci " <i>throw</i> " digunakan untuk melempar eksepsi .
throws	Kata kunci " <i>throws</i> " digunakan untuk menyatakan ekspresi atau mendeklarasikan exception untuk menentukan bahwa mungkin ada exception dalam metode.

6. Penggunaan Blok try catch

Blok *try* digunakan untuk menyertakan kode yang mungkin menimbulkan *exception*. Maka *try* harus digunakan dalam metode. Jika exception terjadi pada pernyataan tertentu di blok *try*, sisa kode dari blok tidak akan dieksekusi. Jadi, disarankan untuk tidak menyimpan kode di blok *try* yang tidak akan mengeluarkan exception.

Blok Java *try* harus diikuti oleh blok *catch* atau *finally*.

Bentuk Penulisan *Blok try catch*

```
try{
    //code that may throw an exception
}catch(Exception_class_Name ref){}
```

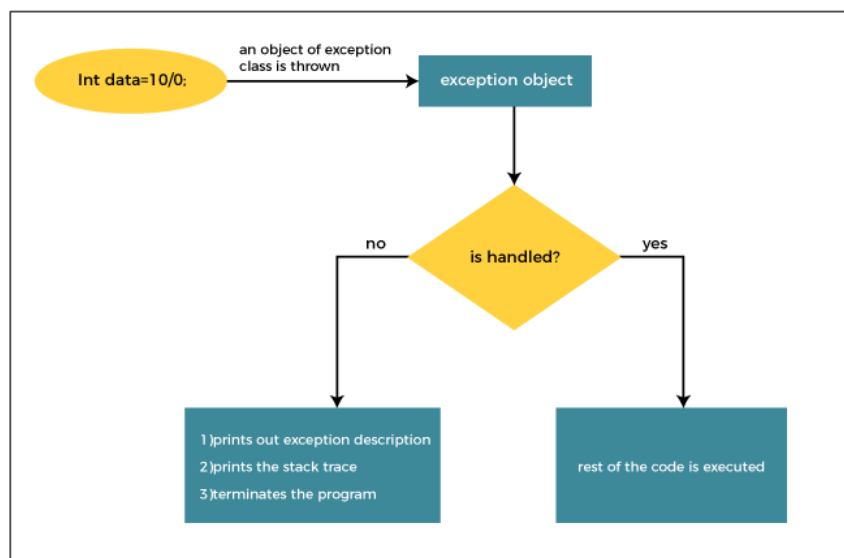
Bentuk Penulisan Blok *try finally*

```
try{
    //code that may throw an exception
}finally{}
```

Blok catch Java digunakan untuk menangani *Exception* dengan mendeklarasikan tipe exception di dalam parameter. *Exception* yang dideklarasikan harus merupakan *Exception* kelas induk atau jenis *Exception* yang dihasilkan.

7. Cara kerja blok try – catch

JVM pertama-tama memeriksa apakah exception ditangani atau tidak. Jika exception tidak ditangani, JVM menyediakan penanganan pengecualian default yang melakukan tugas-tugas berikut:



Gambar 18. 3 cara kerja blok try catch

- Mencetak deskripsi exception.
- Mencetak jejak tumpukan (Hirarki metode tempat exception terjadi).
- Menyebabkan program berhenti.

Contoh program tanpa *exception handling*

TryCatchcontoh1.java

```
public class TryCatchcontoh1 {  
  
    public static void main(String[] args) {  
  
        int data=50/0; //may throw exception  
  
        System.out.println("rest of the code");  
  
    }  
  
}
```

Output :

```
Exception in thread "main" java.lang.ArithmeticException:  
    / by zero
```

Contoh program menggunakan *exception handling* (blok *try catch*)

TryCatchcontoh2.java

```
public class TryCatchcontoh2 {  
  
    public static void main(String[] args) {  
  
        try  
        {  
  
            int data=50/0; //may throw exception  
  
        }  
  
        //handling the exception  
        catch(ArithmeticException e)  
        {  
  
        }  
  
    }  
  
}
```

```
        System.out.println(e);  
    }  
  
    System.out.println("rest of the code");  
}  
}
```

Output :

```
java.lang.ArithmeticException: / by zero  
  
rest of the code
```

Seperti yang ditampilkan dalam contoh di atas, *rest of code* dieksekusi.

Contoh lain program blok *try catch* :

trycatchContoh3.java

```
public class TryCatchContoh3{  
  
    public static void main(String[] args) {  
        int i=50;  
        int j=0;  
        int data;  
        try  
        {  
            data=i/j; //  
        }  
  
        // Penanganan dengan exception  
        catch(Exception e)  
        {  
            // solusi dengan blok try catch exception  
            System.out.println(i/(j+2));  
        }  
    }  
}
```


Output :

25

8. Penggunaan Keyword Throw

Keyword Java **throw** digunakan untuk melempar exception secara eksplisit. Kita menentukan objek exception yang akan dilempar. Exception ini memiliki beberapa pesan yang memberikan deskripsi kesalahan. Exception ini mungkin terkait dengan kesalahan input pengguna, server, dll.

Kita dapat membuang exception yang *unchecked* atau *checked* di Java dengan kata kunci **throw**.

Bentuk umum penulisan keyword **throw**

```
throw new exception_class("error message");
```

contoh penulisan **throw** IOException

```
throw new IOException("sorry device error");
```

Contoh program **throw** unchecked exception

Dalam contoh ini, kita telah membuat metode validasi yang mengambil nilai integer sebagai parameter. Jika usia kurang dari 17, kita melempar `ArithmeticException` jika tidak, cetak pesan selamat datang untuk memilih.

ThrowContoh1.java

```
public class ThrowContoh1{  
    //function mencheck jika seseorang eligible untuk  
    vote atau tidak  
  
    public static void validate(int age) {  
        if(age<17) {  
            //throw Arithmetic exception jika tidak eligible  
            throw new ArithmeticException("Orang ini  
            Belum eligible untuk vote");  
        }  
        else {
```

```
        System.out.println("  Orang  ini  sudah  eligible
untuk  vote!!");
    }
}

//main method

public static void main(String args[]){

    //calling the function

    validate(13);

    System.out.println("rest of the code...");

}

}
```

Output

```
Exception in thread "main" java.lang.ArithmeticException:
Person is not eligible to vote

    at
pkg01sifk001.ThrowContoh1.validate(ThrowContoh1.java:16)
    at
pkg01sifk001.ThrowContoh1.main(ThrowContoh1.java:27)
```

Contoh program *throw User defined exception*

UserDefinedException.java

```
public class UserDefinedException extends Exception
{

    public UserDefinedException(String str)

    {

        // Calling constructor of parent Exception

        super(str);

    }

}
```

```
}
```

throwContoh2.java

```
public class throwContoh2 {  
    public static void main(String[] args) {  
        // TODO code application logic here  
        try  
        {  
            // throw an object of user defined  
exception  
            throw new UserDefinedException("Ini Adalah  
user-defined exception");  
        }  
        catch (UserDefinedException ude)  
        {  
            System.out.println("Menangkap  
exception");  
            // Print the message from MyException  
object  
            System.out.println(ude.getMessage());  
        }  
    }  
}
```

Output

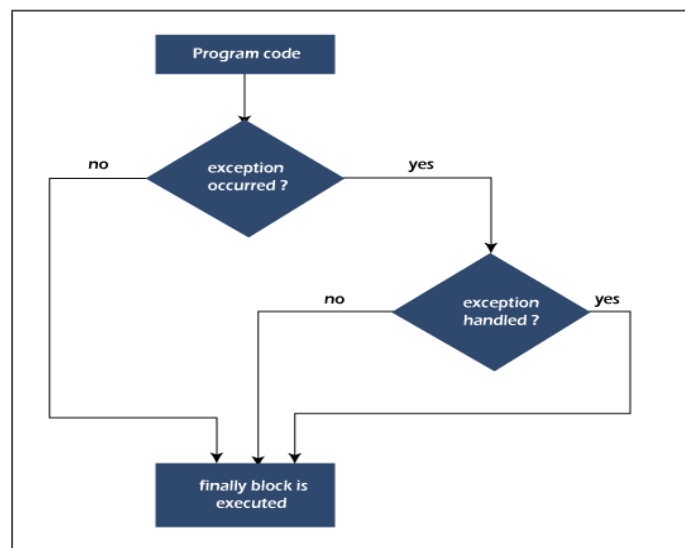
```
Menangkap exception  
Ini Adalah user-defined exception
```

9. Penggunaan Blok Finally

Blok finally pada Java adalah blok yang digunakan untuk mengeksekusi kode penting seperti menutup koneksi, dll. Blok ini selalu dieksekusi apakah *exception* ditangani atau tidak. Oleh karena itu, berisi semua pernyataan yang diperlukan yang perlu dicetak terlepas dari *exception* terjadi atau tidak.

Mengapa menggunakan blok finally ?

- Blok Finally di Java dapat digunakan untuk meletakkan kode "pembersihan" seperti menutup file, menutup koneksi, dll.
- Pernyataan-pernyataan penting yang akan dicetak dapat ditempatkan di blok terakhir.



Gambar 18. 4 Blok Finally

Contoh penggunaan blok finally

FinallyBlokContoh.java

```
class FinallyBlokContoh {  
    public static void main(String args[]){  
        try{  
            //below code do not throw any exception
```

```
int data=50/5;

System.out.println(data);

}

//catch won't be executed

catch(NullPointerException e){

System.out.println(e);

}

//executed regardless of exception occurred or not

finally {

System.out.println("blok finally akan selalu di eksekusi");

}

System.out.println("rest of the code...");

}

}
```

Output

```
10

blok finally akan selalu di eksekusi

rest of the code
```

C. LATIHAN/TUGAS

1. Terdapat 2 cara untuk *exception* yaitu :
 - a. Blok *try catch* (Menangkap Exception)
 - b. *throw* (Melempar exception)

Apa perbedaan dari kedua exception tersebut, buat contoh programnya untuk masing *exception*.

2. Buat program blok finally dibawah ini lalu apa hasil dari outputnya :

```
class FinallyBlokContoh {  
  
    public static void main(String args[]) {  
  
        try{  
//below code do not throw any exception  
  
            Float data=500/15;  
  
            System.out.println(data);  
  
        }  
  
        catch(NullPointerException e){  
System.out.println(e);  
}  
  
        finally {  
System.out.println("blok finally akan selalu di eksekusi  
");  
}  
}
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4th Edition* ,san jose university , united state Of America. RRD jefferson city publishing.

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , USA, publishing as prentice hall.

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, <https://beginnersbook.com/java-tutorial-for-beginners-with-examples/>, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , <https://www.javatpoint.com/java-tutorial>, diakses pada tanggal 1 Desember 2021

Wahono Satrio Romi, Java Fundamental, <https://romisatriawahono.net/java/>, diakses pada tanggal 5 Desember 2022