Modul Pemrograman Berbasis Objek (PBO)
File Stream Dan Exception Part 1

Pada modul ini kali ini kita akan mempelajari mengenai file stream. Sebelum memasuki materi file stream kita akan membahas terlebih dahulu mengenai Exception.

Untuk mengenal lebih tentang exception, temen temen bisa bayangin ketika temen temen buat suatu aplikasi yang berhubungan dengan database, nah tetapi temen temen menghubungkan ke database yang tidak pernah.

Ketika aplikasi di compile, aplikasi tidak memunculkan error, tetapi saat aplikasi dijalankan maka aplikasi muncul error karena database tidak ditemukan. Nah error yang ditemukan ketika aplikasi dijalankan dapat dikendalikan oleh kita. Hal itulah yang dinamakan exception.

Mungkin temen temen masih belum terlalu paham dengan maksud diatas, mari kita coba saja dalam kodingan

Buat folder bernama ErrorHandling, lalu buat file bernama FileError.java, dan masukan kode dibawah ini

```
public class FileError {
 public static void main(String[ ] args) {
   try {
     int[] myNumbers = {1, 2, 3};
     System.out.println(myNumbers[10]);
   } catch (Exception e) {
       System.out.println("Terdapat error");
   }
 }
}
```

Coba temen temen jalankan kode tersebut didalam terminal, apakah yang terjadi

Nah mungkin teman teman bertanya, kenapa ketika di compile tidak ada error, tetapi saat dijalankan terdapat error

Alasan nya, proses compile hanya menerjemahkan java ke dalam bahasa mesin versi java bukan menjalankan nya.

Contoh nya, ketika kode memiliki variabel dengan tipe data integer, lalu variabel tersebut dimasukan tipe data string, nah hal tersebut jelas akan error ketika proses compile, karena compiler tidak dapat menerjemahkan variabel yang memiliki data string tetapi variabel tersebut memiliki tipe data integer

int a = “fikri”;

compiler tidak bisa menerjemahkan variabel a kedalam bahasa mesin karena a bertipe data integer sedangkan yang dimasukan ke variabel tersebut adalah fikri yang bertipe data string

Sekarang mari kita lihat kembali pada kode

```
try {
  int[] myNumbers = {1, 2, 3};
  System.out.println(myNumbers[10]);
} catch (Exception e) {
    System.out.println("Terdapat error");
}
```

Pada kode diatas, kita membuat variabel array bernama myNumber bertipe data integer, lalu kita memasukan data sebanyak 3 data. Setelah itu kita memanggil kembali myNumber pada index 10. Secara penulisan kode, tidak terdapat penulisan kode yang salah sehingga program tetap dapat di compile secara normal dan menghasilkan terjemahan dari bahasa java ke bahasa mesin. Nah, saat program dijalankan barulah

muncul error, karena ketika dijalankan program tidak mendeteksi adanya array pada posisi ke 10.

Nah, sekarang mungkin teman teman penasaran penggunaan try catch pada kode diatas. Mari kita bahas kode diatas kembali, coba temen temen berikan komentar pada kode try dan catch seperti dibawah ini

```
public class FileError {
 public static void main(String[ ] args) {
   // try {
     int[] myNumbers = {1, 2, 3};
     System.out.println(myNumbers[10]);
   // } catch (Exception e) {
   //     System.out.println("Terdapat error");
 }
}
```

Lalu, coba temen temen jalankan pada terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$ javac FileError.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$ java FileError
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 3
        at FileError.main(FileError.java:5)
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$
```

Error tetap muncul meskipun kita menggunakan try catch, tetapi coba temen temen hapus komentar pada kode lalu jalankan file nya

```
public class FileError {
 public static void main(String[ ] args) {
   try {
       int[] myNumbers = {1, 2, 3};
       System.out.println(myNumbers[10]);
   } catch (Exception e) {
       System.out.println("Terdapat error");
   }
}
```

Nah sekarang, mungkin temen temen sadar bahwa pesan error yang dihasilkan tanpa try catch memiliki pesan yang sulit dibaca. Tetapi dengan try catch kita dapat membuat pesan error kita sendiri

Tanpa try catch

Dengan try catch

Dengan menggunakan try catch kita dapat dengan mudah mengenali error.

Nah, tapi pertanyaan nya, sebenarnya apa sih try catch itu. Simplenya kode yang didalam try akan dijalankan, tetapi jika kode yang berada didalam try memiliki error, maka kode akan langsung beralih ke catch dan menjalankan kode yang berada didalam catch.

Kebalikan nya, semisal kode didalam try tidak memiliki error, maka kode tersebut tidak akan beralih ke catch dan akan beralih ke kode setelah catch.

Hal ini mirip dengan if else, jika kondisi if terpenuhi maka else tidak akan dijalankan, tetapi jika if tidak terpenuhi maka kondisi else akan dijalankan.

Nah pertanyaan kembali, ketika ada error pada program saat dijalankan, program dapat memberikan pesan error itu sendiri, lantas kenapa kita perlu membuat pesan error secara manual menggunakan try catch?

Jawaban nya, saat aplikasi digunakan oleh user dan terjadi error, user tidak akan mengerti error yang dimaksud, dari pada memberikan error yang tidak dimengerti oleh user lebih baik membuat pesan error manual yang dapat dipahami oleh user. Karena itu kita menggunakan try and catch.

Sekarang, kita bedah kembali kode try and catch diatas

```
try {
    int[] myNumbers = {1, 2, 3};
    System.out.println(myNumbers[10]);
} catch (Exception e) {
    System.out.println("Terdapat error");
}
```

Terdapat tulisan Exception e pada catch, nah Exception e adalah sebagai parameter dari catch yang berfungsi jika terdapat error pada program saat dijalankan, maka catch tersebut dijalankan

Selain Exception e terdapat parameter lain di catch, contoh nya `ArrayIndexOutOfBoundsException`

`ArrayIndexOutOfBoundsException` menandakan bahwa jika index array yang di panggil tidak ada di posisi atau kita menambahkan array melewati batas yang telah ditentukan, maka error tersebut akan terjadi

Jadi semisal kita memasukan

catch(`ArrayIndexOutOfBoundsException e`) maka catch tersebut akan terpanggil jika terdapat error yang dimaksud. Tetapi jika errornya bukan hal tersebut maka catch nya tidak akan berjalan. Jadi kesimpulan

nya, bahwa catch ini dapat kita tentukan sesuai dengan error yang akan diterima. Dan dari sini kita dapat membuat multiple catch

```
   try {
       int[] myNumbers = {1, 2, 3};
       System.out.println(myNumbers[10]);
   } catch (Exception e) {
       "Semua error yang diterima akan terpanggil disini"
   } catch (ArrayIndexOutOfBoundsException) {
       "Error ini berkaitan dengan array"
   } catch (IOException e) {
       "Error ini berkaitan dengan input output"
   }
```

Kode diatas, tidak perlu ditulis di file teman teman, nanti kita akan menggunakan berbagai parameter exception pada penjelasan materi file stream. Intinya adalah kita dapat membuat berbagai catch yang akan dijalankan sesuai error yang ditentukan pada parameter nya. Semisal error nya ada pada array, maka catch yang akan dijalankan adalah catch yang memiliki parameter `ArrayIndexOutOfBoundsException`.

`Notes : Jumlah parameter pada catch ada banyak, tidak hanya ArrayIndexOutOfBoundsException dan Exception e. Temen temen dapat melakukan searching untuk mengetahui parameter lain nya. Beberapa parameter lain nya, akan dibahas saat saat penjelasan file stream`

Sekarang kita kembali kepada kodingan kita, teman teman tambahkan kode dibawah ini masih pada file yang sama

```
public class FileError {
 public static void main(String[ ] args) {
   try {
     int[] myNumbers = {1, 2, 3};
     System.out.println(myNumbers[10]);
   } catch (Exception e) {
       System.out.println("Terdapat error");
   } finally {
       System.out.println("Try catch berhasil kita jalankan");
   }
 }
}
```

Selain try catch, terdapat juga finally. finally digunakan setelah try atau catch dijalankan. Tidak ada hal yang spesial disini. Teman teman pun tetap bisa menambahkan kode setelah finally dan akan tetap dieksekusi di terminal

Referensi untuk try catch dapat teman temen lihat disini :

https://www.w3schools.com/java/java_try_catch.asp

https://www.tutorialspoint.com/java/java_exceptions.htm

Notes : huruf e pada parameter catch dapat diganti dengan huruf apa saja, seperti misal nya :

```
catch(Exception a)

catch(Exception b)
```

huruf huruf tersebut digunakan sebagai variabel dari exception. Variabel tersebut akan berisi informasi tentang error dari program

```
catch(Exception e) {
     // menampilkan error e nya pada terminal
     System.out.println("Errornya adalah " + e);
}
```

Kita telah membahas mengenai try dan catch. Mari beralih ke throw

throw kita gunakan bersamaan dengan exception. Simpelnya, throw kita gunakan untuk membuat “error” versi kita sendiri.

Mari buat file baru, buatlah file bernama ErrorFile.java, dan masukan kode di bawah ini

```
public class ErrorFile {
 static void checkAge(int age) {
   if (age < 18) {
     throw new ArithmeticException("Access denied - You must be at
                                   least 18 years old.");
   }
   else {
     System.out.println("Akses diberikan, umur telah mencukupi");
   }
 }

 public static void main(String[] args) {
   checkAge(13);
 }
}
```

Notes : Static method tidak hanya bisa dibuat dibawah method main. Tetapi dapat juga diatas method main

Mari kita coba jalankan kode diatas

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$ javac ErrorFile.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$ java ErrorFile
Exception in thread "main" java.lang.ArithmeticException: Access denied - You must be at least 18 years old.
        at ErrorFile.checkAge(ErrorFile.java:4)
        at ErrorFile.main(ErrorFile.java:12)
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$
```

Pada method checkAge, parameter checkAge yaitu variabel age akan melalui proses if else. Jika parameter yang diberikan adalah angka

dibawah 18 tahun maka akan terjadi error sesuai dengan error yang kita tentukan. Pada contoh diatas, kita memanggil error `ArithmeticException.`

Nah pada kodingan kita memasukan angka 13 pada method checkAge(). Karena 13 tidak lebih besar dari pada 18, maka error `ArithmeticException` dipanggil.

Nah itulah yang di maksud dengan membuat error kita sendiri, jadi kita dapat memanggil error yang kita perlukan pada kodingan yang kita buat dengan menggunakan throw

Selain menggunakan `ArithmeticException` , kita juga dapat berbagai error lain nya. Contoh nya kita juga dapat memanggil error `ArrayIndexOutOfBoundsException, FileNotFoundException, dan lain nya` pada kode yang kita buat dengan menggunakan throw.

Notes : Parameter error yang digunakan baik pada catch atau throw memiliki fungsi nya masing masing. Contohnya ArithmeticException , error tersebut memiliki arti bahwa terdapat error pada proses aritmatika. Untuk lebih mengetahui penggunaan masing masing error kalian dapat mensearching mengenai parameter error yang digunakan pada catch dan throw.

Mari kita memanfaatkan fungsi try catch pada throw. Tambahkan kode dibawah ini pada file ErrorFile.java

```
public class ErrorFile {
 static void checkAge(int age) {
   if (age < 18) {
     throw new ArithmeticException("Access denied - You must be at
                                    least 18 years old.");
   }
   else {
     System.out.println("Akses diberikan, umur telah mencukupi");
   }
 }
 public static void main(String[] args) {
   checkAge(20);
   try {
       checkAge(13);
   } catch (ArithmeticException e) {
```

```
        System.out.println("File error karena = " + e);
    }
  }
}
```

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$ javac ErrorFile.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$ java ErrorFile
Akses diberikan, umur telah mencukupi
File error karena = java.lang.ArithmeticException: Access denied - You must be at least 18 years old.
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 10/ErrorHandl
ing$
```

Yap, dapat kita lihat try catch juga bekerja pada kode yang telah kita buat. Saat kita memasukan angka 13 pada method checkAge didalam try, terjadi error karena 13 tidak lebih besar dari 18. Error tersebut dilempar ke catch, dan catch menjalankan kode perintah yang ada didalamnya.

Kita telah membahas mengenai exception, mari kita lanjut membahas file stream. Teman teman dapat membuka part 2 dari modul ini untuk pembahasan file stream.