Modul 4
Exception, I/O, dan Operasi File

Tujuan:

1. Mengerti konsep exception, I/O, dan operasi file.
2. Bisa implementasiin konsep exception dan I/O dalam sebuah pemrograman sederhana untuk operasi file dengan tepat.

Sering kali, ketika menjalankan sebuah program kita menemukan kesalahan. Kesalahan itu bisa berasal dari hardware, software, atau bahkan dalam algoritma atau kesalahan logika itu sendiri.

Nah, jadi masing-masing kesalahan punya solusi yang beda-beda dan teknik penanganannya juga berbeda-beda untuk kesalahan yang satu dengan yang lain.

Untuk itu, mari kita bahas satu-satu kesalahan yang ada.

Kesalahan itu ada apa aja sih?

Kesalahan Hardware dan Software

Hardware saja

Jika sebuah aplikasi berusaha membuka sebuah file yang gak ada, misalnya ngirimin karakter ke printer yang belom kita nyalain atau ketika berkomunikasi dengan port serial yang gak merespon.

Biasanya, kesalahan menyangkut hardware dideteksi dan dilaporkan oleh sistem.

Software saja

Ketika kode berusaha mengakses sebuah elemen yang di luar ikatan array atau berusaha nyimpen nilai yang ngelewatin kapasitas format data.

Kesalahan yang terkait sama software, dipihak lain, harus dideteksi sama kode

Hardware dan Software

sebuah aplikasi mungkin memeriksa operan pembagi untuk memastikan bahwa sebuah pembagi dengan nol tidak dicoba. Bagaimanapun juga, jika sebuah pembagian oleh nol terjadi, perangkat-keras dalam sebagaian besar sistem komputer menghasilkan sebuah respon kesalahan.

Kesalahan Algoritma

berhubungan dengan kesalahan atau keterbatasan intrinsik dari pemodelan dunia-nyata yang dilakukan oleh komputer.

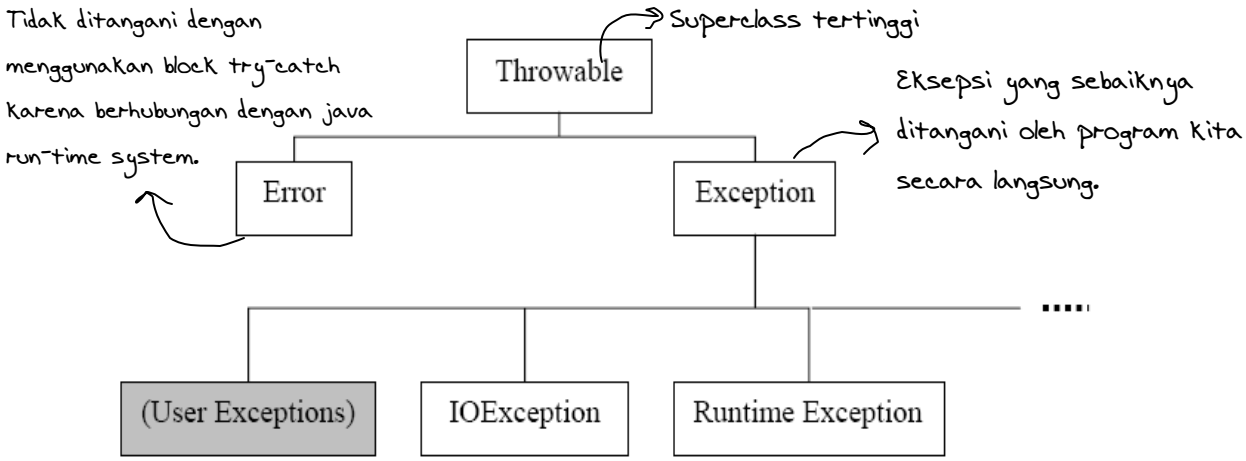
Untuk mengatasi kesalahan pembulatan dan pemotongan pada berbagai algoritma muncullah disiplin ilmu bernama analisa numerik.

Jenis kesalahan ini adalah yang paling sering diabaikan oleh programmer.

Perkiraan yang dilakukan oleh komputer bisa menyebabkan kesalahan pembulatan yang bisa menyebar dalam perhitungan dan mengakibatkan kesalahan hasil.

Exception

Exception adalah kondisi abnormal / tidak wajar yang terjadi pada saat pengeksekusian suatu perintah. Dalam java ada lima keywords yang digunakan untuk menangani eksepsi ini yaitu : try, catch, finally, throw, dan throws.



Semua class exception terdapat dalam package java.lang.

try this at home

Exception yang tidak dicek

Semua yang bertipe Runtime Exception dan turunannya tidak harus ditangani dalam program kita.

```
class RuntimeException
{
   public static void main (String [] args)
   {
      int[] arr = new int[1];
      System.out.println(arr[1]);
   }
}
```

Outputnya:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
        at RuntimeException.main(RuntimeException.java:6)
```

Terjadi kesalahan ArrayIndexOutOfBoundsException karena array pada contoh diatas Cuma punya 1 record arr[0], tapi program minta nampilin arr[1].

Exception yang dicek

Semua tipe eksepsi yang bukan turunan dari class RuntimeException merupakan eksepsi yang harus ditangani oleh program anda. Java bahkan tidak mengijinkan anda mengompilasi program yang anda buat, jika tidak menangani eksepsi tersebut. (BIASANYA YANG BERHUBUNGAN DENGAN PENGAKSESAN I/O).

Penggunaan Try Catch

Untuk menangani eksepsi yang terjadi dalam program, Anda cukup meletakkan kode yang ingin anda inspeksi terhadap kemungkinan eksepsi di dalam blok try, diikuti dengan blok catch yang menentukan jenis eksepsi apa yang ingin Anda tangani.

Struktur penulisan:

```
try{
            ......//blok program
} catch (tipeException e) {
            ......//blok program
}
```

Try menyatakan bahwa dalam blok program dapat terjadi suatu eksepsi dan bila itu terjadi, maka eksekusi program berlanjut dalam blok catch sesuai dengan tipe eksepsi yang terjadi. Variabel obyek e dari tipe exception dapat digunakan sebagai referensi / informasi untuk mengetahui penyebab exception tersebut.

try this at home

```
public class TryCatch {
   public static void main (String[] args){
      try{
            int[] arr = new int[1];
            System.out.println(arr[1]);
            System.out.println("Baris ini tidak akan dieksekusi,
            karena statement baris diatas terkadi exception");
      }catch (ArrayIndexOutOfBoundsException e){
            System.out.println("Terjadi exception karena indeks di luar
            kapasitas array");
      }
      System.out.println("Setelah blok try catch");
   }
}
```

Perhatikan baris ini!

Outputnya:

```
Terjadi exception karena indeks di luar kapasitas array
Setelah blok try catch
```

Dapat terjadi code yang terdapat dalam blok try mengakibatkan lebih dari satu jenis eksepsi. Dalam hal ini, kita dapat menuliskan lebih dari satu blok catch untuk setiap blok try.

Kira-kira seperti apa ya bentuk kodenya??

try this at home

```
public class TryCatch2 {
    public static void main (String [] args){
      try{
            int x = args.length; //banyak argumen
            int y = 100/x;

            int[] arr = {10,11};
            y = arr[x];

            System.out.println("Tidak terjadi exception");

      }catch (ArithmeticException e){
            System.out.println("Terjadi exeption karena
            pembagian dengan nol");

      }catch (ArrayIndexOutOfBoundsException e){
            System.out.println("Terjadi exeption karena indeks
            di luar kapasitas array");
      }

      System.out.println("Setelah blok try catch");

    }
}
```

```
C:\Java_projects >java TryCatch2

Terjadi exception karena pembagian dengan nol
```

Outputnya jika tanpa parameter

Outputnya jika dengan sebuah parameter

```
C:\Java_projects >java TryCatch2 param1

Tidak terjadi exception
```

```
C:\Java_projects >java TryCatch2 param1 param2

Terjadi exception karena indeks di luar kapasitas array
```

Outputnya jika dengan 2 parameter

Penggunaan Throw

Mekanisme throw memungkinkan program untuk mengirim / melempar exception untuk kemudian program menyikapi (bereaksi) atas exception tersebut.

Cara penulisan:

```
Throw ObjekEksepsi;
```

ObjekEksepsi adalah semua objek yang merupakan turunan dari class Throwable. Salah satu contoh objek eksepsi adalah ArrayIndexOutOfBounds.

Seperti apa kira-kira bentuk programnya?

try this at home

```
public class CobaThrow{
   public static void methodLain() {
      try{
            throw new ArrayIndexOutOfBoundsException(1);
      }catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("Penanganan exception
            dalam method methodLain()");
            throw e;
      }
   }

   public static void main (String [] args){
      try{
            methodLain();
      }catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Penanganan exception
            dalam method main()");
      }
   }
}
```

objek dari class ArrayIndexOutOfBoundsException, yang kemudian dikreasi dengan konstruktor plus parameter integer sebagai identitas objek tersebut.

Outputnya:

```
Penanganan exception dalam method methodLain()
Penanganan exception dalam method main()
```

Eksepsi juga dapat dibangkitkan secara manual melalui kode yang kita tulis. Kita dapat melakukan hal tersebut dengan kata kunci throw ini, seperti pada contoh diatas.

```
int[] arr = new int[1];
System.out.println(arr[1]);
```

Eksepsi ini gak perlu ditulis lagi, karena kita udah pake throw

Penggunaan Throws

Sangat erat hubungannya dengan penggunaan exception yang dicek oleh Java.

Keyword ini dipake kalo misalnya ada method yang mungkin menyebabkan exception tapi dia gak menangani exception tersebut.

Exception akan dilempar ke luar, maka dari itu ketika keyword "throws" digunakan saat pendeklarasian method.

Masih bingung? Let's see the code..

Cara penulisan:

```
class NamaClass throws Exception{
            .........
}

atau fungsi / prosedur

int f1() throws Exception{
            ..........
}
```

Karena ada ini, makanya gak ada blok try-catch nya

Outputnya:

```
Lebih kecil 5
Program selesai
```

```
public class CobaThrows{
      public void tampil() throws Exception {
            int x=0;
            if (x<5)
                  throw new Exception ("Lebih kecil 5");
      }

}


   public class DemoThrows{
         public static void main (String [] args){
               CobaThrows c = new CobaThrows ();
               try {
                     c.tampil();
               }catch (Exception e) {
                     System.out.println(e.getMessage());
               }
               System.out.println ("Program selesai");
         }
```

Nah pas method di atas dipanggil di sini, makanya, method itu ada di antara blok try-catch

try this at home

Penanganan Dengan Finally

Apa itu Finally?

Penggunaan blok try catch terkadang membingungkan karena kita tidak dapat menentukan dengan pasti alur mana yang akan dieksekusi. Apalagi penggunaan throw yang mengakibatkan kode setelah throw tidak akan dieksekusi atau justru terjadi kesalahan pada blok catch, menyebabkan program akan berhenti. Untuk mengatasi problem ini, Java memperkenalkan keyword finally. Dimana semua kode yang ada dalam blok finally "pasti" akan dieksekusi apapun yang terjadi di dalam blok try catch.

Cara penulisan:

```
try{
            ......//blok program
} catch (tipeException1 e) {
            ......//blok program untuk
TipeException1
} catch (tipeException2 e) {
            ......//blok program untuk
TipeException2
} finally {
            ......//blok program
}
```

Supaya output yang terjadi seperti ini, bagaimana kah kodenya?

```
Terjadi exception
Program Selesai
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: -1
        at DemoFinally.main(DemoFinally.java:11)
```