EXCEPTION HANDLING

A Little Demo

```
public class Test {
    public static void main(String[] args) {
        int i = 6;
        int j = 3;
        System.out.println(i/j);
      }
    }
```

Output :

```
2
```

A Little Demo

```
public class Test {
    public static void main(String[] args) {
        int i = 6;
        int j = 0;
        System.out.println(i/j);
      }
    }
```

Output :

```
Exception in thread "main"
        java.lang.ArithmeticException: / by zero
        at Test.main(Test.java:4)
```

Exception

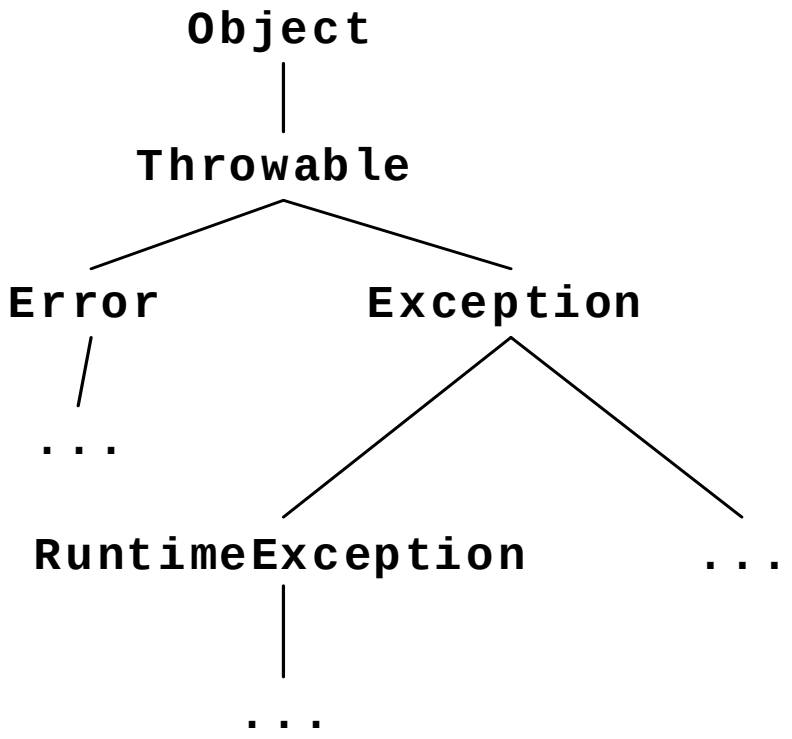
- Suatu object Error khusus yang dibuat ketika terjadi suatu kesalahan pada suatu program. Dengan exception handling yang baik suatu program akan terhindar dari ”Hang”.
- Exception dipicu oleh *run time error* dan digunakan sebagai sarana melaporkan kondisi-kondisi kesalahan.

Some Predefined Exceptions

Java Exception	Code to Cause It
NullPointerException	**String s = null; s.length();**
ArithmeticException	**int a = 3; int b = 0; int q = a/b;**
ArrayIndexOutOfBoundsException	**int[] a = new int[10]; a[10];**
ClassCastException	**Object x = new Integer(1); String s = (String) x;**
StringIndexOutOfBoundsException	**String s = "Hello"; s.charAt(5);**

Jenis-Jenis Exception

- Exception merupakan objek dari subkelas yang diturunkan dari kelas *Throwable*. Kelas *Throwable* ini terdapat dalam package *java.lang.object*.



Jenis-Jenis Exception

- **Kelompok Kelas Error**

  Error ini bersifat fatal sehingga sistem tidak dapat dimanipulasi untuk diperbaiki, contoh kelas: *LinkageError*, *VirtualMachineError*, dan *AWTError*.

- **Kelompok Kelas Exception**

  Jenis error ini masih dapat diantisipasi dengan menyisipkan statement tambahan untuk mendeteksi data yang berpotensi menimbulkan error.

Jenis-Jenis Exception

- Ada kelompok exception yang diperiksa oleh interpreter, apakah akan ditangani atau dilempar, namun ada pula exception yang akan tidak diperiksa interpreter.
- Disamping itu programmer dibolehkan membuat exception sendiri dengan cara *extends* atau *implements* kelas *Exception*

Tabel Checked Exception

No	*Exception*	Deskripsi
1	*ClassNotFoundException*	Kelas tidak ditemukan
2	*CloneNotSupportedException*	melakukan clone objek yang tidak mengimplementasikan interface *Cloneable*
3	*IllegalAccessException*	Pengaksesan ke kelas ditolak
4	*InstantiationException*	Menciptakan objek dari kelas abtract ataupun dari interface
5	*InterruptedException*	Thread telah diinterupsi oleh thread lain
6	*NoSuchFieldException*	Field yang diminta tidak ada
7	*NoSuchMethodException*	Method yang diminta tidak ada

Tabel Uncheked Exception

No	Exception	Deskripsi
1	*AritmaticException*	Kesalahan Aritmatik seperti pembagian dengan nol
2	*ArrayIndexOutBoundsException*	Index array di luar batas
3	*ArrayStoreException*	Pemberian nilai ke elemen array tidak sesuai dengan tipenya
4	*ClassCastException*	Cast yang tidak sah
5	*IllegalArgumentException*	Argument illegal
6	*IllegalMonitorStateException*	Operasi monitor illegal seperti menunggu di thread yang tidak terkunci
7	*IllegalStateException*	Lingkungan atau aplikasi state yang tidak benar
8	*IllegalThreadStateException*	Operasi yang diminta tidak kompatibel dengan state thread saat itu
9	*IndexOutOfBoundException*	Indeks di luar batas

Tabel Uncheked Exception

10	*NegativeArraySizeException*	Array diciptakan dengan ukuran negatif
11	*NullPointerException*	Penggunan null yang tidak sah
12	*NumberFormatException*	Konversi yang tidak sah dari string ke format numerik
13	*SecurityException*	Melanggar aturan security
14	*StringIndexOutOfBounds*	Index di luar batas string
15	*UnsupportedOperationException*	Ditemukan operasi yang tidak didukung

Dua Exception yang belum dikelompokkan, yaitu *IOException* dan *AWTException*.

Mengantisipasi Exception

a. Mendeklarasikan Exception

Bentuk umum :

```
[modifier] returntype namaMethod() throws tipeException {

}
```

Contoh :

```
public void operasiMatematika() throws IOException,
    ClassNotFoundException {
}
public void beriPinjaman() throws TolakException{
}
```

b. Melempar Exception

Bentuk umum :

```
TipeException namaObjek = new TipeException;
throw namaObjek;
```

Diringkas menjadi :

```
throw namaObjek TipeException;
```

atau

```
throw new TipeException();
```

Contoh :

```
TolakException t = new TolakException(”lagi pelit”);
throw t;
```

Diringkas menjadi :

```
throw new TolakException(”lagi pelit!”);
```

c. Menangkap Exception

Bentuk umum :

```
try {
   //pemanggilan method yg mungkin menghasilkan exception
   //blok statement yg mungkin menghasilkan exception
}
catch(TipeException1 namaObjek) {
    // penanganan salah-satu jenis exception
}
catch(TipeException2 namaObjek) {
    // penanganan salah-satu jenis exception
}
catch(TipeExceptionN namaObjek) {
    // penanganan salah-satu jenis exception
}
finally {
    // blok yang harus dieksekusi
}
```

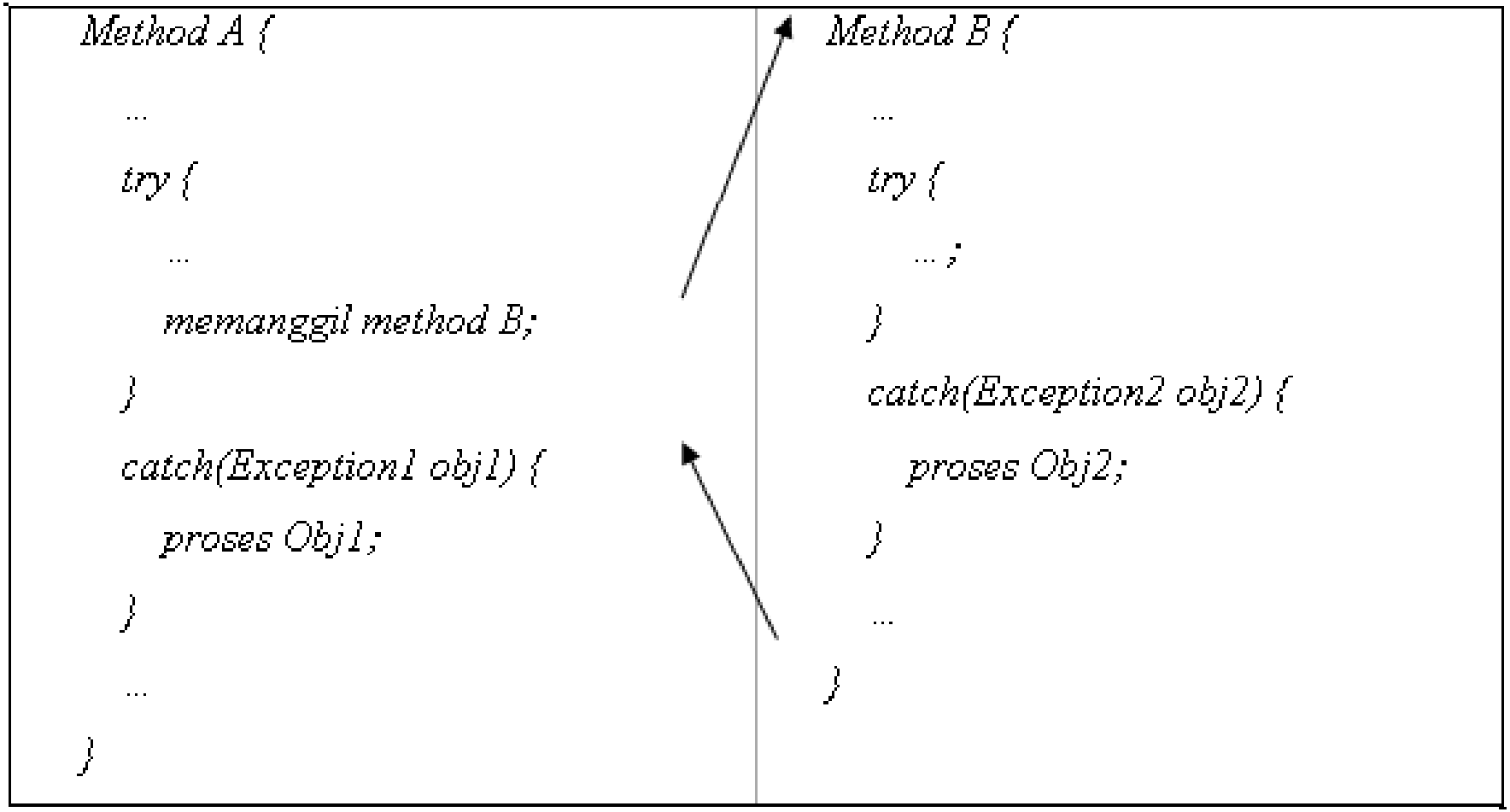
- Blok *try* tidak *exception*, maka blok *catch* tidak ada yang dieksekusi dan segera blok *finally* yang dieksekusi.
- Jika terjadi *exception* pada blok *try*, maka salah satu blok *catch* dieksekusi, kemudian blok *finally* dieksekusi.

Mekanisme Mengantisipasi Exception

tiga kemungkinan skenario exception :

1. jika tidak terjadi exception (tidak ada blok catch yang dieksekusi)
2. jika exception terjadi pada blok method tunggal (salah-satu blok catch dieksekusi)
3. jika terjadi exception pada blok tersarang.

Mekanisme Mengantisipasi Exception



Menampilkan Pesan Exception

Beberapa method standard yang dapat digunakan untuk menampilkan pesan exception merupakan anggota dari kelas *java.lang.Throwable*.

No	Method Pesan Exception	Deskripsi
1	*getMessage()*	Mengembalikan nilai string yang berisi pesan rinci tentang objek Throwable yang mengalami exception
2	*toString()*	Mengembalikan nilai string yang berisi pesan singkat tentang objek yang mengalami exception
3	*getLocalizedMessage()*	Menampilkan pesan exception lokal (yang terjadi pada subkelas saja)
4	*printStackTrace()*	Method ini bersifat void, dan hanya mencetak informasi tentang objek Throwable

Contoh Membuat Exception

```
class MyException extends Exception {
   MyException(String s) {
      super(s + " tidak diperbolehkan!");
   }
}
```

```
class Eksepsi {
   static void tampil(String s) throws MyException {
      System.out.println("Tampil: " + s);
      if (s.equals("amir"))
         throw new MyException(s);
      System.out.println("OK!");
   }
   public static void main(String argv[]) {
      try {
         tampil("ali");
         System.out.println("Heh");
         tampil("amir");
      }
      catch (MyException ex) {
         System.out.println(ex);
      }
      finally {
         System.out.println("OK");
      }
   }
}
```

Output :

Tampil : ali
OK!
Heh
Tampil : amir
MyExcepton : amir tidak diperbolehkan
OK