Exception Handling

BRIGIDA ARIE MINARTINGTYAS, M.KOM

Jarang sekali sebuah program dibuat dapat berjalan dengan sukses pada saat pertama sekali dijalankan

Kesalahan sering terjadi pada saat perancangan atau pemrograman

Kategori Error

Syntax Errors

- mengakibatkan kesalahan kompilasi.

Semantic Errors

- program menghasilkan keluaran yang tidak sesuai dengan harapan.

Runtime Errors

- kebanyakan mengakibatkan terminasi program secara tidak normal atau bahkan sistem crash

Contoh Run Time Error

- Pembagian bilangan dengan nol.
- Akses elemen yang berada di luar indeks array
- Menyimpan elemen data yang tidak kompatibel
- Menggunakan nilai negatif untuk ukuran array
- Mengubah data string menjadi data integer (misal., mengubah “abc” ke nilai integer).
- Banyak lagi …

Contoh Tanpa Error Handling

```java
public class NoErrorHandling {
    public static void main(String[] args){
            int a,b;
            a = 7;
            b = 0;

            System.out.println("Result is " + a/b);
                    System.out.println("Program reached this line");
        }

}
```

Output - JavaApplication1 (run)

```
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ErrorHandling.NoErrorHandling.main(NoErrorHandling.java:18)
C:\Users\Tyas\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```

Contoh dengan Error Handling Tradisional

```java
public class WithErrorHandling {
    public static void main(String[] args){
        int a,b;
        a = 7;   b = 0;
        if (b != 0){
            System.out.println("Result is " + a/b);
        }
        else{
            System.out.println(" B is zero");
        }
            System.out.println("Program is complete");
    }

}
```

Output - JavaApplication1 (run)

```
run:
 B is zero
Program is complete
BUILD SUCCESSFUL (total time: 0 seconds)
```

Error Conditions	Setiap program dalam berada dapat suatu kondisi yang tidak normal
	Program yang ‘baik’ harus dapat menangani kondisi ini.
Exceptions	Java menyediakan suatu mekanisme untuk menangani kondisi ini

Exception

- merupakan suatu keadaan yang disebabkan oleh runtime error dalam program.
- memungkinkan kesalahan ditangani tanpa harus ‘mengotori’ program (dengan rutin yang menangani kesalahan)
- memungkinkan pemisahan penanganan kesalahan dengan program utama (*main business logic*)

	Ketika JVM menjumpai error, Java akan membuat objek exception dan melemparkannya – sebagai tanda bahwa error telah terjadi
	Jika objek exception tidak ditangkap dan ditangani secara tepat, interpreter akan menampilkan error dan mengakhiri program
Exception Handling	Apabila ingin program menjalankan program yang tersisa, maka objek exception yang dilempar tadi harus ditangkap dan diambil tindakan yang sesuai.

Exception yang sering terjadi

- ArithmeticException
- ArrayIndexOutOfBoundException
- ArrayStoreException
- FileNotFoundException
- IOException – general I/O failure
- NullPointerException – merujuk ke null object
- StackOverflowException
- StringIndexOutOfBoundException

Exception di Java

Throws

- Suatu fungsi dapat memberi tanda suatu kesalahan dengan melempar suatu exception

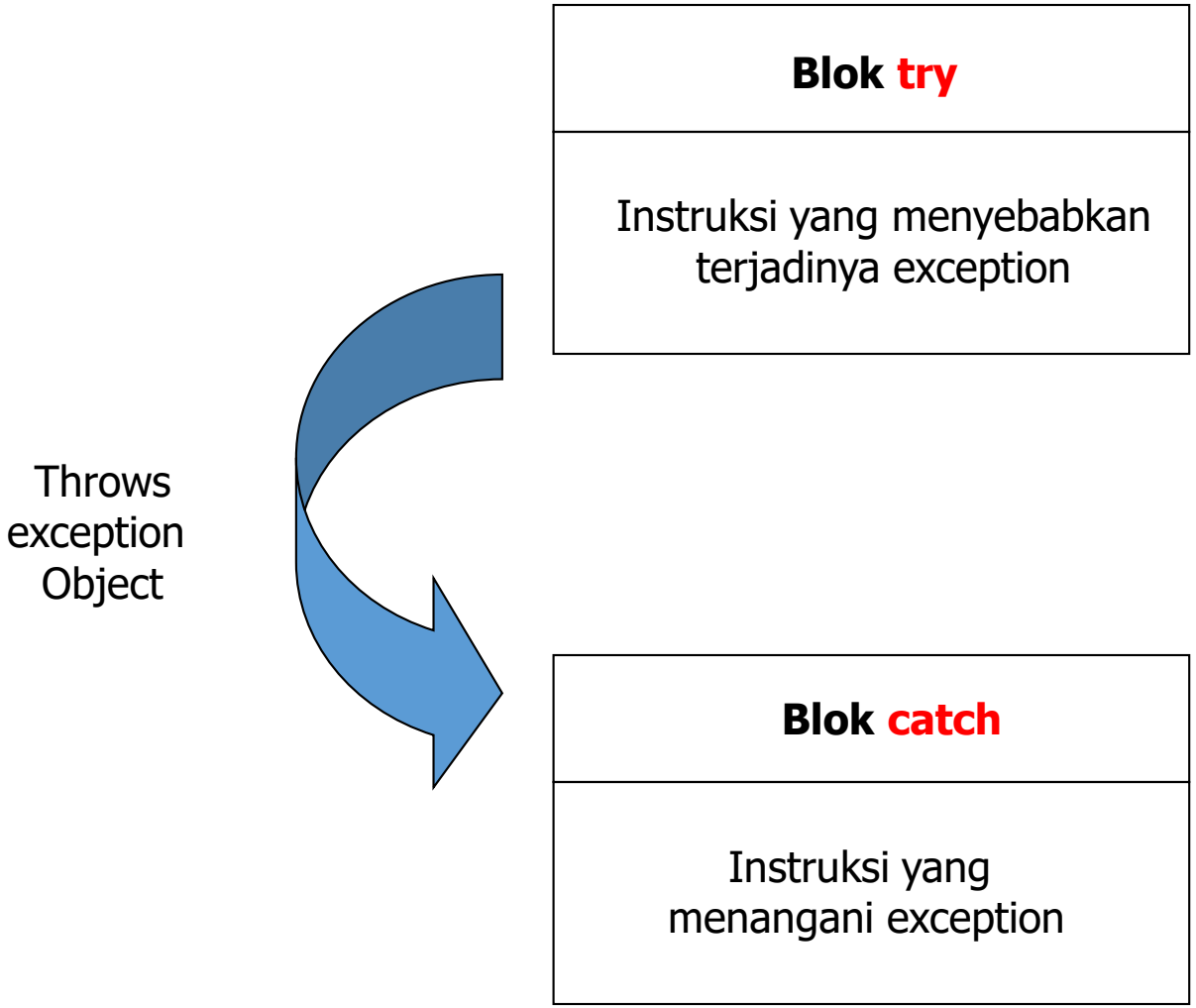
Try, Catch

- Fungsi pemanggil dapat menyerahkan kendali ke exception handler dengan menangkap (catching) exception

Finally

- Clean up

Penanganan Exception



Blok Try - Catch

```
try {
    instruksi yang berpotensi menghasilkan exception
    }
catch(tipe_exception e) {
    instruksi untuk menangani exception
    }
```

Catch Exception

Letakkan instruksi yang akan diperiksa (berpotensi menghasilkan exception) dalam blok **try**

Buat satu atau lebih blok catch setelah blok try

- Handlers di-check berdasarkan urutan penulisan. Letakkan handler yang paling sering digunakan pertama sekali.
- Eksekusi normal dilanjutkan setelah handler yang terakhir

Catch Exception

Terjadi eksepsi
(asumsi instruksi-2 melempar eksepsi)

```
try {
        instruksi-1
        instruksi-2
        instruksi-3
        .......

        diabaikan
        instruksi-n
}
catch(Exception e)
{
        instruksi-catch-1
        instruksi-catch-2
        .....
        instruksi-catch-4
}

instruksi-lain
......
```

Tidak terjadi eksepsi

```
try {
        instruksi-1
        instruksi-2
        instruksi-3
        .......
        instruksi-n
}
catch(Exception e)
{
        instruksi-catch-1
        instruksi-catch-2
        .....
        instruksi-catch-4
}

instruksi-lain
......
```

Contoh Tanpa Exception (1)

```java
public class NoErrorHandling {
    public static void main(String[] args){
            int a,b;
            a = 7;
            b = 0;

            System.out.println("Result is " + a/b);
                    System.out.println("Program reached this line");
        }

}
```

Output - JavaApplication1 (run)

```
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ErrorHandling.NoErrorHandling.main(NoErrorHandling.java:18)
C:\Users\Tyas\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```

Contoh Try Catch (1)

```java
public class WithExceptionHandling {
    public static void main(String[] args){
            int a,b;  float r;
            a = 7;   b = 0;
            try{
                r = a/b;
                System.out.println("Result is " + r);
            }
            catch(ArithmeticException e){
                System.out.println(" B is zero");
            }
                System.out.println("Program reached this line");
        }

}
```

Output - JavaApplication1 (run)

```
run:
 B is zero
Program reached this line
BUILD SUCCESSFUL (total time: 0 seconds)
```

Contoh Tanpa Exception (2)

```java
public class TryCatch {
    public static void main(String[] args) {
        int angka = 7;
        int hasil = angka/0;
        System.out.println(hasil);
    }
}
```

Output - JavaApplication1 (run)

```
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ErrorHandling.TryCatch.main(TryCatch.java:15)
C:\Users\Tyas\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```

Contoh Try Catch (2)

```java
public class TryCatch2 {
    public static void main(String[] args) {
        try{
            // pernyataan yang berpotensi mengakibatkan Exception
            int angka = 7;
            int hasil = angka/0;
            System.out.println(hasil);
        }catch(ArithmeticException ex){
            // pernyataan disini akan di eksekusi jika terjadi Exception
            System.out.println("Tidak Boleh Menggunakan Pembagian dengan 0 (nol)");
        }
    }
}
```

Output - JavaApplication1 (run)

```
run:
Tidak Boleh Menggunakan Pembagian dengan 0 (nol)
BUILD SUCCESSFUL (total time: 0 seconds)
```

Contoh Try Catch (3)

```java
public class TryCatch3 {
    public static void main(String[] args) {
        try{
            // pernyataan yang berpotensi mengakibatkan Exception
            String[] siswa = new String[2];
            siswa[0] = "Wildan";
            siswa[1] = "Ferdi";
            siswa[2] = "Taufiq";
            System.out.println(siswa[4]);
        }catch(ArrayIndexOutOfBoundsException ex){
            // pernyataan disini akan di eksekusi jika terjadi Exception
            System.out.println("Data Array Yang Ingin Dikeluarkan Tidak Ada");
        }
    }
}
```

Output - JavaApplication1 (run)

```
run:
Data Array Yang Ingin Dikeluarkan Tidak Ada
BUILD SUCCESSFUL (total time: 0 seconds)
```

Apabila blok try mungkin menimbulkan lebih dari satu exception, maka menggunakan ***multiple catch***

```
…
…
try {
 // instruksi
}
catch(Exception-Type1 e)
{
// instruksi untuk menangani exception 1
}
..
..
catch(Exception-TypeN e)
{
// instruksi untuk menangani exception 1
}
```

Multiple Catch

```
try {
        instruksi-1
        instruksi-2
        instruksi-3
        .......
                        diabaikan
        instruksi-n
}
catch(tipe_eksepsi_1 e1)
{
}
catch(tipe_eksepsi_2 e2)
{
}
catch(tipe_eksepsi_3 e3)
{
}
catch(tipe_eksepsi_4 e4)
{
}
....
catch(tipe_eksepsi_n en)
{
}

instruksi-lain
......
```

```
try {
        instruksi-1
        instruksi-2
        instruksi-3
        .......
        instruksi-n
}
catch(tipe_eksepsi_1 e1)
{
}
catch(tipe_eksepsi_2 e2)
{
}
catch(tipe_eksepsi_3 e3)
{
}
catch(tipe_eksepsi_4 e4)
{
}
....
catch(tipe_eksepsi_n en)
{
}

instruksi-lain
......
```

Contoh Multiple Catch

```java
public class MultipleCatch {
    public static void main(String[] args) {
        try{
            String[] siswa = new String[2];
            siswa[0] = "Wildan";
            siswa[1] = "Ferdi";
            siswa[2] = "Taufiq";
            System.out.println(siswa[4]);
            //=====================================
            int angka = 7;
            int hasil = angka/0;
            System.out.println(hasil);
        }catch(ArrayIndexOutOfBoundsException ex){
            System.out.println("Data Array Yang Ingin Dikeluarkan Tidak Ada");
        }catch(ArithmeticException ex2){
            System.out.println("Tidak Boleh Menggunakan Pembagian dengan 0 (nol)");
        }
    }
}
```

Output - JavaApplication1 (run)

```
run:
Data Array Yang Ingin Dikeluarkan Tidak Ada
BUILD SUCCESSFUL (total time: 0 seconds)
```

Blok Finally	Menangani setiap exception yang tidak ditangkap oleh blok sebelumnya
	Blok ini dijalankan tidak peduli apakah exception terjadi atau tidak.
	Digunakan untuk melakukan house keeping operation misal menutup file dan mengembalikan system resources

Blok Finally

```
try {
 // statements
}
catch( Exception-Type1 e)
{
// statements to process exception 1
}
..
..
finally {
   ….
}
```

Contoh Finally

```java
public class Finally {
    public static void main(String[] args) {
        try{
            // pernyataan yang berpotensi mengakibatkan Exception
            int angka = 10;
            int hasil = angka/0;
            System.out.println(hasil);
        }catch(ArithmeticException ex){
            // pernyataan disini akan di eksekusi jika terjadi Exception
            System.out.println("Tidak Boleh Menggunakan Pembagian dengan 0 (nol)");
        }finally{
            /*
            Pernyataan disini akan di eksekusi jika terjadi Exception
            Ataupun tidak terjadi Exception
            */
            System.out.println("Program Diakhiri");
        }
    }
}
```

Output - JavaApplication1 (run)

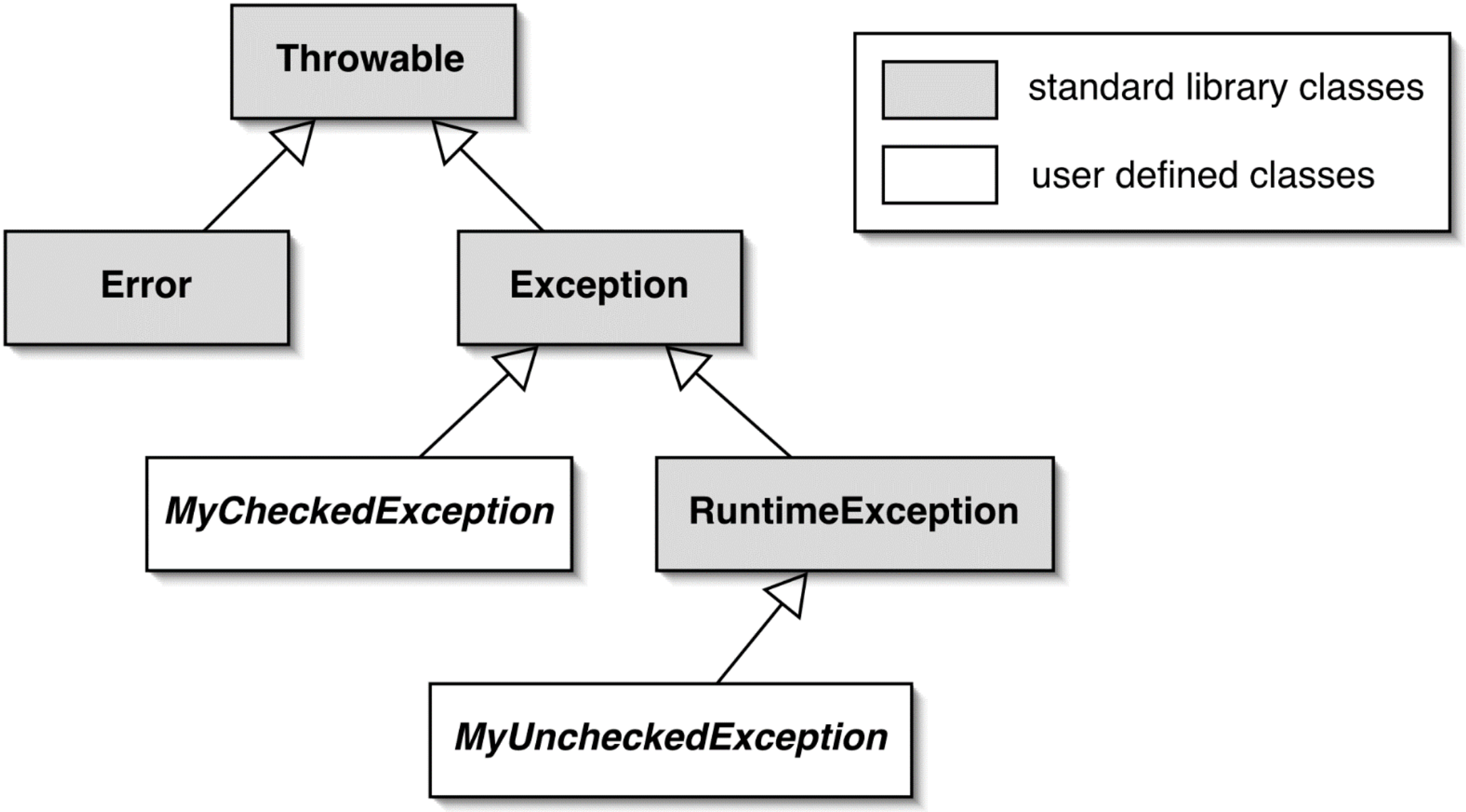
```
run:
Tidak Boleh Menggunakan Pembagian dengan 0 (nol)
Program Diakhiri
BUILD SUCCESSFUL (total time: 0 seconds)
```

User Defined Exception	Merupakan turunan **Exception** or **Runtime-Exception**
	Membuat tipe baru untuk memberikan informasi diagnostik yang lebih baik
	Perlu mengetahui mekanisme **throw** suatu objek exception

Throw Exception

- Objek yang di-throw harus merupakan turunan dari kelas **Throwable** atau kelas-kelas turunannya
- Biasanya diturunkan dari kelas java.lang.Exception
- Kelas yang dibuat (optional) menyediakan 2 konstruktor
  - Default : MyException( )
  - Argumen string : MyException(String s)

Throw Exception



Fungsi untuk Pemrosesan Throw Exception

```
getMessage( )
```

```
toString( )
(class name + message)
```

```
printStackTrace( )
```

Throw Exception

```
if (condition)
    throw new MyException();
```

atau

```
if (condition)
    throw new MyException("bla ..bla ..bla");
```

User Defined Exception - Format Standard -

```
class MyException extends Exception
{
MyException(String message)
{
super(message);
// pass to superclass if parameter is not handled by used defined exception
}
}

class TestMyException {
…
  try {
      …
      throw new MyException("This is error message");
      }
catch(MyException e)
{
System.out.println("Message is: "+e.getMessage());
}
}
```

```java
public class A {
    public static void main(String[] args) {
    try {
        throw new Exception("kesalahan terjadi");
    }
    catch(Exception e){
        System.out.println(e);
    }
    }
}
```

Output - JavaApplication1 (run)

```
run:
java.lang.Exception: kesalahan terjadi
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class CobaThrow {
    public void tampil() {
            try {
                    int x=0;
                    if (x<5)
                            throw new Exception("Lebih kecil 5");
            }
            catch (Exception e) {
                    System.out.println(e.getMessage());
            }
    }
}
```

```java
public class DemoThrow {
    public static void main (String args[]) {
            CobaThrow c = new CobaThrow();
            c.tampil();
            System.out.println("Program Selesai");
    }
}
```

DemoThrow

tput - Modul7 (run)

```
run:
Lebih kecil 5
Program Selesai
BUILD SUCCESSFUL (total time: 0 seconds)
```

User Defined Exception

```
public Circle(double centreX, double centreY, double radius){
     x  = centreX; y = centreY;  r = radius;
}
```

Bagaimana memastikan bahwa radius tidak nol atau negatif?

```java
import java.lang.Exception;
public class InvalidRadiusException extends Exception{
    private double r;
        public InvalidRadiusException(double radius){
            r = radius;
        }
        public void printError(){
            System.out.println("Radius [" +  r + "] is not valid");
        }
}
```

```java
public class Circle {
        double x, y, r;
    public Circle (double centreX, double centreY, double radius ) throws InvalidRadiusException
        {
        x = centreX;
            if (r <= 0 )
            {
                throw new InvalidRadiusException(radius);
            }
            else
            {
                x = centreX ; y = centreY;  r = radius;
            }
        }
}
```

```java
public class CircleTest {
    public static void main(String[] args) {
        try {
            Circle cl = new Circle(10, 10, -1);
            System.out.println("Circle created");
        }
        catch(InvalidRadiusException e)
        {
            e.printError();
        }
    }

}
```

Output - JavaApplication1 (run)

```
run:
Radius [-1.0] is not valid
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class tes {
    public static void main(String[] args) {
        int a[]=new int[5];
        a[5]=100;
    }
}
```

out - JavaApplication1 (run)

```
run:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at ErrorHandling.tes.main(tes.java:15)
C:\Users\Tyas\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```

```java
public class tes2 {
    public static void main(String[] args) {
        int i=0;
        String greeting[]={
            "Hello World!",
            "No, I mean it!",
            "Hello World"
        };
        while(i<4){
            System.out.println(greeting[i]);
            i++;
        }
    }
}
```

ErrorHandling.tes2 > main >

out - JavaApplication1 (run)

```
run:
Hello World!
No, I mean it!
Hello World
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
        at ErrorHandling.tes2.main(tes2.java:21)
C:\Users\Tyas\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```