

MODUL 8

EXCEPTION HANDLING

Praktikum Pemrograman Berorientasi Objek
2023



Pemegang Modul: Vannes Satria Gunawan

Tujuan Pembelajaran

1. Memahami konsep exception handling dan cara mengimplementasikannya dalam pemrograman Java.
2. Dapat membuat User Defined Exception untuk suatu kasus tertentu.

Pendahuluan

Biasanya dalam perancangan atau pemrograman, kita tidak dapat menghindari kesalahan atau kita dapat menyebutnya sebagai *Error*. *Error* umumnya diklasifikasikan menjadi tiga bagian, yaitu:

1. Syntax Error

Error ini dapat terjadi ketika programmer menggunakan syntax yang salah saat menulis code, sehingga menyebabkan error pada waktu kompilasi. Contoh:

- Lupa menambahkan titik koma (;) ke code program
- Kesalahan pada beberapa huruf tertentu (string → dengan ‘s’ kecil)

2. Semantic Error

Error ini terjadi ketika output dari program tidak seperti dengan apa yang diharapkan.

Contoh:

- Kalian ingin mengeluarkan “Anda Lulus” jika nilai lebih besar dari 80 (nilai > 80), tetapi code program mengeluarkan “Anda Gagal”.
- Kalian ingin menampilkan kata “Aku Bisa!” sebanyak 5 kali, tetapi hasil yang tampil hanya 4 kali.

3. Runtime Error

Error ini pada awalnya akan berfungsi normal, hingga mencapai kode program yang menyebabkan error. Error ini dapat menyebabkan program berhenti secara tidak normal yang bisasa disebut *crash*.

Contoh:

- Pembagian dengan 0.
- Menggunakan indeks array yang lebih besar dari ukuran array.
- Ketika program meminta anda untuk menginput angka, tetapi anda menginputkan huruf.

Exception adalah kondisi yang disebabkan oleh runtime error, sehingga kondisi exception ini harus ditangani (handling), agar program dapat berlanjut hingga keluar secara normal. Oleh karena itu, Exception Handling adalah mekanisme yang diperlukan untuk menangani error.

Terdapat kata kata kunci yang dapat digunakan di Exception Handling, yaitu:

1. Try

Untuk meletakkan code yang berpotensi menyebabkan error.

2. Catch

Intruksi yang menangani exception.

3. Finally

Bagian ini akan tetap dijalankan baik ketika ada exception atau tidak, bagian ini bersifat opsional.

4. Throw

Keyword ini digunakan untuk melemparkan kelas exception yang telah kita buat.

5. Throws

Keyword ini digunakan untuk method atau konstruktor yang di dalamnya ada kondisi yang akan menyebabkan error.

Berikut adalah bentuk umum dari try, catch, finally

```
try{
// berisikan code-code yang berpotensi adanya eksepsi
}
catch(Exception e1) {
// berisikan kode yang digunakan untuk menangani exception ke-1
}
catch(Exception eN) {
// berisikan kode yang digunakan untuk menangani exception ke-N
}
finally {
// blok ini akan jalan ketika catch-catch sudah terlewati
}
```

Logikanya, blok try-catch-finally akan bekerja dengan mekanisme sebagai berikut:

1. Blok **Try**, baris code yang memungkinkan akan terjadinya suatu exception
2. Ketika terjadi exception, maka suatu objek exception akan dilemparkan ke blok penangkap error, **Catch**

Note: Karena exception bisa terdiri banyak jenisnya, maka dimungkinkan untuk membuat lebih dari satu blok catch (multiple catch)

3. Kemudian apabila setelah blok catch sudah tidak ada lagi, maka blok **finally** akan dijalankan.

User Defined Exception

Terkadang suatu program memerlukan penggunaan exception untuk menangani kasus yang tidak dapat ditangani oleh kelas default exception. Atau program memerlukan exception yang dapat memberikan informasi yang lebih baik kepada pengguna tentang kesalahan yang saat ini terjadi dalam program. Dalam kasus seperti itu, harus ada user defined exception.

User defined exception adalah suatu kelas exception baru yang dibuat oleh programmer yang mana merupakan turunan dari kelas Exception. Suatu user defined exception harus dibuat sebagai turunan dari kelas Exception untuk mewarisi sifat Throwable dari kelas Exception. Hal ini dilakukan agar kelas exception yang dihasilkan dapat di Throw ketika terjadi kesalahan pada kode program. Throw dan Throws adalah kata kunci yang digunakan untuk pada user defined exception. Kerangka penggunaan throw-throws yaitu :

```
public class User_defined_Exception extends
    Exception{public void showMessage(){
        System.out.println("Isi Pesan Errornya apa");
    }
}

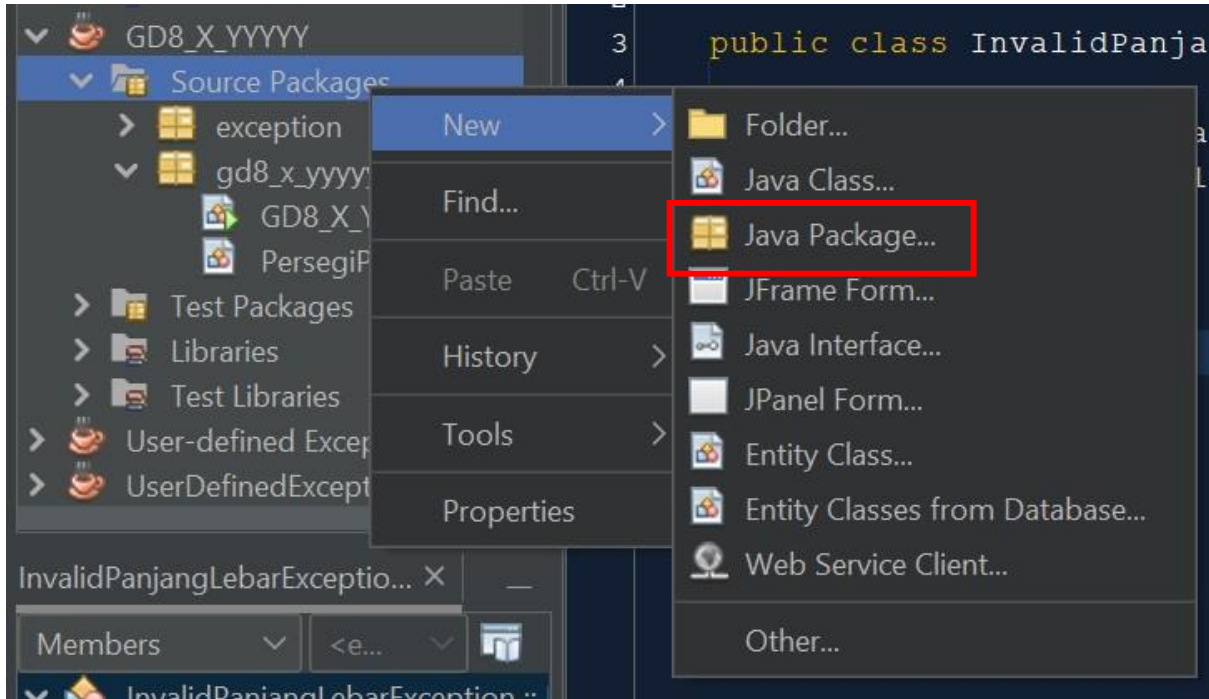
public class Something {
    public Something() throws
        User_defined_Exception{if(kondisi){
            throw new User_defined_Exception();
        }
        else{
            ...
        }
    }
}
```

Berikut ini adalah fungsi-fungsi yang tersedia untuk pemrosesan exception:

1. `getMessage()` : Akan mengembalikan detail pesan tentang exception yang terjadi.
2. `toString()` (class name + message) : Akan mengembalikan nama kelas dan akan menggabungkan dengan fungsi `getMessage()`.
3. `printStackTrace()` : Akan menampilkan hasil dari fungsi `toString()` beserta dengan jejak error.

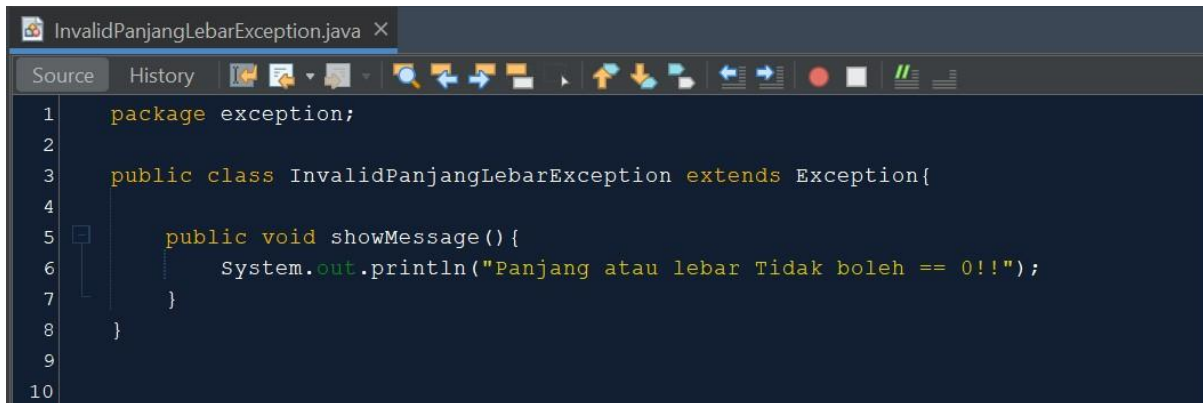
GUIDED**LANGKAH-LANGKAH**

1. Buatlah sebuah projek pada netbeans kalian dengan nama GD8_X_YYYYY (X = Kelas, YYYYYY = 5 digit terakhir NPM).
2. Buatlah Package baru bernama exception.



3. Kita akan membuat **java class** baru di dalam package exception bernama `InvalidPanjangLebarException` dan `InputNegativeException`, dua kelas ini akan mengextends kelas `Exception`.

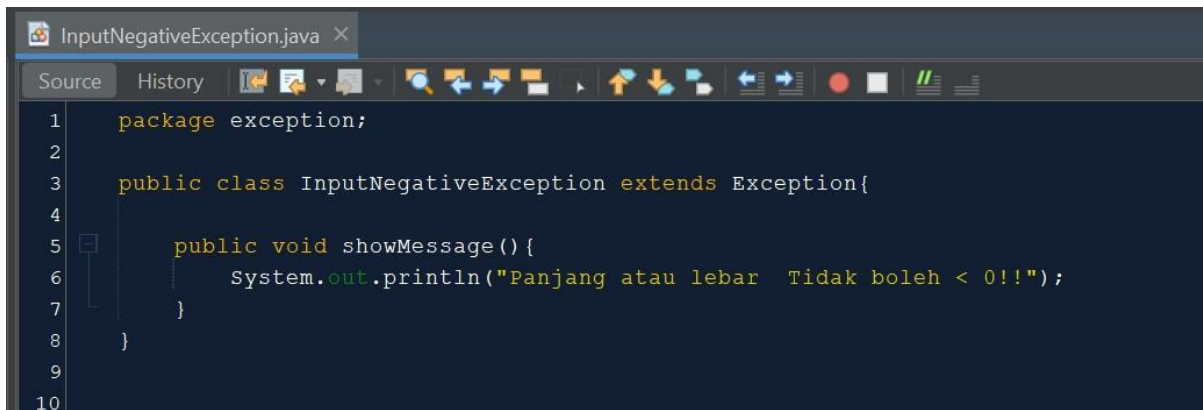
Code pada kelas `InvalidPanjangLebarException`:

A screenshot of an IDE window titled 'InvalidPanjangLebarException.java'. The code is as follows:

```
1 package exception;
2
3 public class InvalidPanjangLebarException extends Exception{
4
5     public void showMessage(){
6         System.out.println("Panjang atau lebar Tidak boleh == 0!!");
7     }
8 }
9
10
```

Pada kelas `InvalidPanjangLebarException` ini akan mempunyai sebuah Method yang berfungsi untuk menampilkan sebuah pesan jika exception tersebut terjadi.

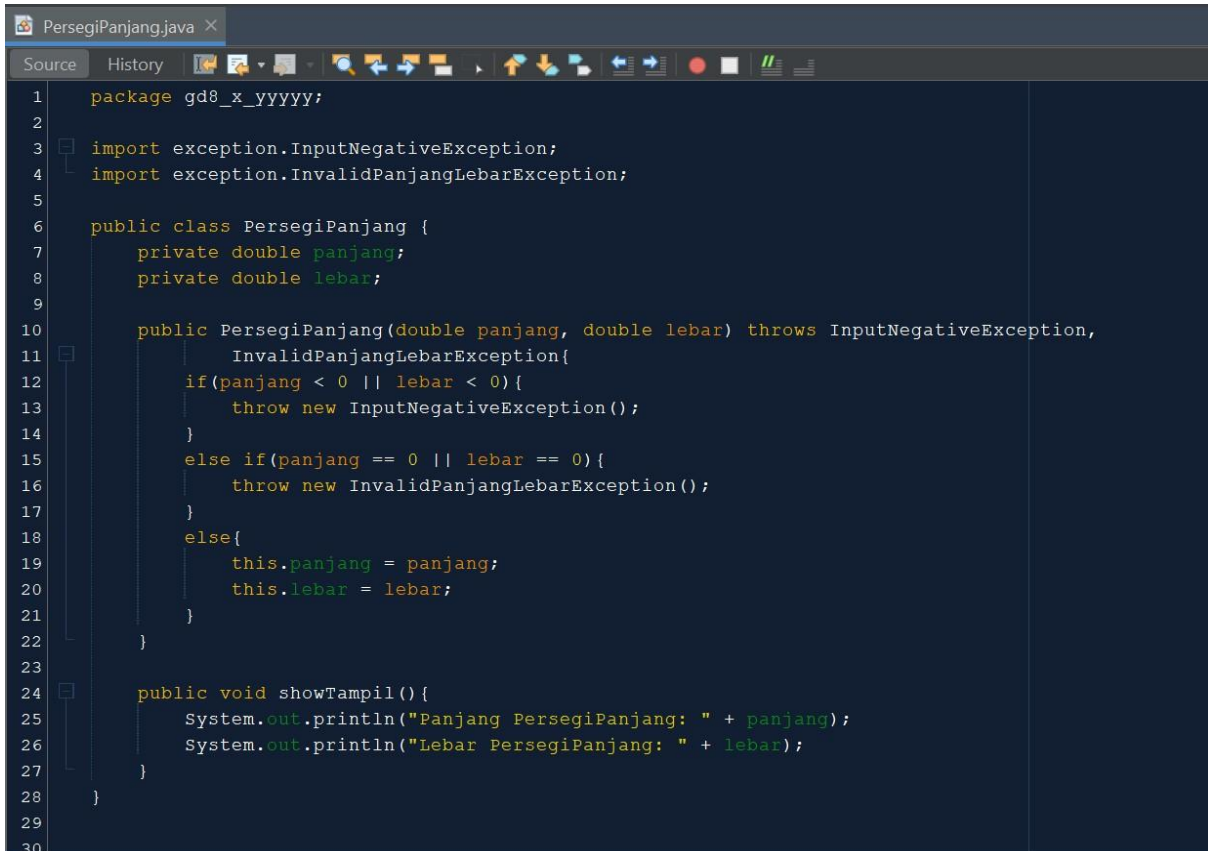
Code pada kelas `InputNegativeException`:

A screenshot of an IDE window titled 'InputNegativeException.java'. The code is as follows:

```
1 package exception;
2
3 public class InputNegativeException extends Exception{
4
5     public void showMessage(){
6         System.out.println("Panjang atau lebar Tidak boleh < 0!!");
7     }
8 }
9
10
```

Pada kelas `InputNegativeException` ini juga mempunyai sebuah Method yang berfungsi untuk menampilkan sebuah pesan jika user menginputkan panjang atau lebar negatif.

4. Selanjutnya, kita membutuhkan kelas baru yaitu PersegiPanjang, seperti berikut ini:



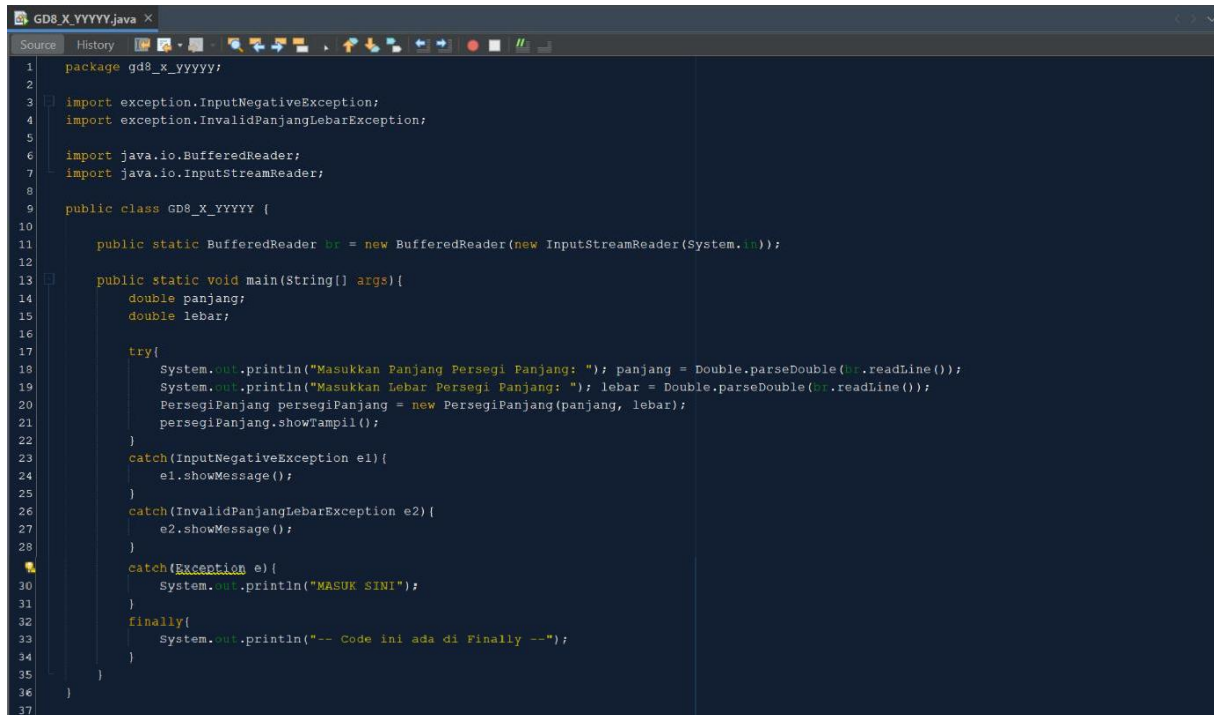
```
1 package gd8_x_yyyyy;
2
3 import exception.InputNegativeException;
4 import exception.InvalidPanjangLebarException;
5
6 public class PersegiPanjang {
7     private double panjang;
8     private double lebar;
9
10    public PersegiPanjang(double panjang, double lebar) throws InputNegativeException,
11        InvalidPanjangLebarException{
12        if(panjang < 0 || lebar < 0){
13            throw new InputNegativeException();
14        }
15        else if(panjang == 0 || lebar == 0){
16            throw new InvalidPanjangLebarException();
17        }
18        else{
19            this.panjang = panjang;
20            this.lebar = lebar;
21        }
22    }
23
24    public void showTampil(){
25        System.out.println("Panjang PersegiPanjang: " + panjang);
26        System.out.println("Lebar PersegiPanjang: " + lebar);
27    }
28 }
29
30
```

JANGAN LUPA UNTUK MENGIMPORT KELAS EXCEPTION!!

Terlihat pada konstruktor dari kelas PersegiPanjang terdapat keyword **Throws** (disebelah parameter) dan **Throw** di bagian isi dari konstruktor tersebut, **Throws** dan **Throw** tersebut diikuti dengan kelas Exception yang kalian panggil.

Untuk penulisan kelas Exception jika lebih dari satu, kalian hanya perlu menambahkan koma (,).

5. Lalu pada main class kalian, kalian memakai try catch untuk meletakkan code yang berpotensi menyebabkan exception pada try, lalu kita akan menambahkan sebuah catch baru. Pada catch baru kita akan menambahkan parameter dari kelas exception yang telah kita buat, yaitu `InvalidPanjangLebarException` dan `InputNegativeException` lalu untuk memanggil method yang ada di dalamnya kita akan memanggil method yaitu `showMessage()` seperti berikut ini:

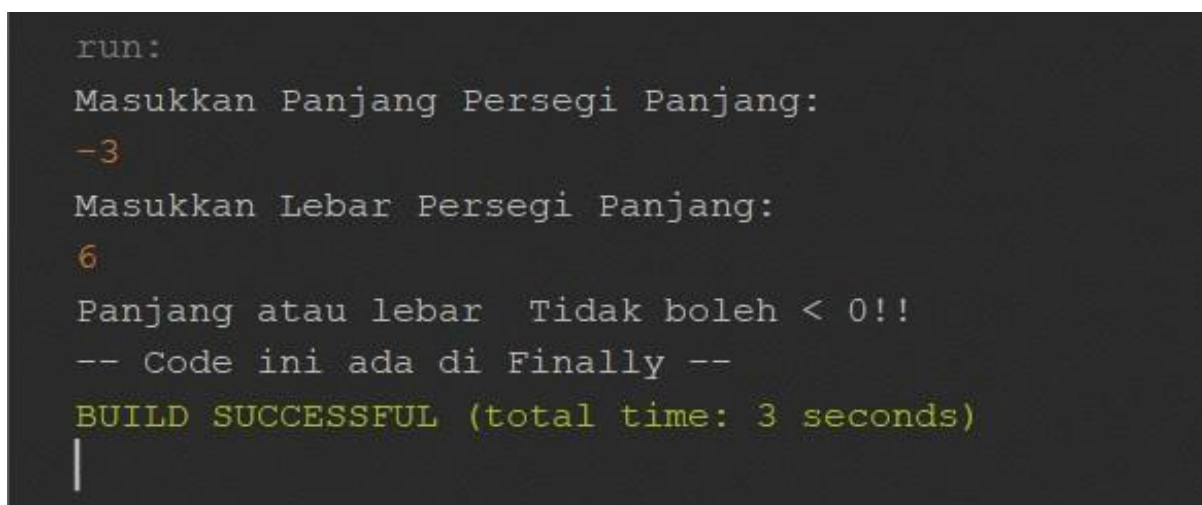


```

1 package gd8_x_yyyyy;
2
3 import exception.InputNegativeException;
4 import exception.InvalidPanjangLebarException;
5
6 import java.io.BufferedReader;
7 import java.io.InputStreamReader;
8
9 public class GD8_X_YYYYY {
10
11     public static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12
13     public static void main(String[] args) {
14         double panjang;
15         double lebar;
16
17         try {
18             System.out.println("Masukkan Panjang Persegi Panjang: "); panjang = Double.parseDouble(br.readLine());
19             System.out.println("Masukkan Lebar Persegi Panjang: "); lebar = Double.parseDouble(br.readLine());
20             PersegiPanjang persegiPanjang = new PersegiPanjang(panjang, lebar);
21             persegiPanjang.showTampil();
22         }
23         catch (InputNegativeException e1) {
24             e1.showMessage();
25         }
26         catch (InvalidPanjangLebarException e2) {
27             e2.showMessage();
28         }
29         catch (Exception e) {
30             System.out.println("MASUK SINI");
31         }
32         finally {
33             System.out.println("-- Code ini ada di Finally --");
34         }
35     }
36 }
37

```

6. Setelah itu apabila kita mencoba memasukkan value minus atau < 0 maka pesan tersebut akan muncul.



```

run:
Masukkan Panjang Persegi Panjang:
-3
Masukkan Lebar Persegi Panjang:
6
Panjang atau lebar Tidak boleh < 0!!
-- Code ini ada di Finally --
BUILD SUCCESSFUL (total time: 3 seconds)
|

```

7. Setelah itu kita akan mencoba untuk menginput panjang atau lebar dengan inputan value 0.

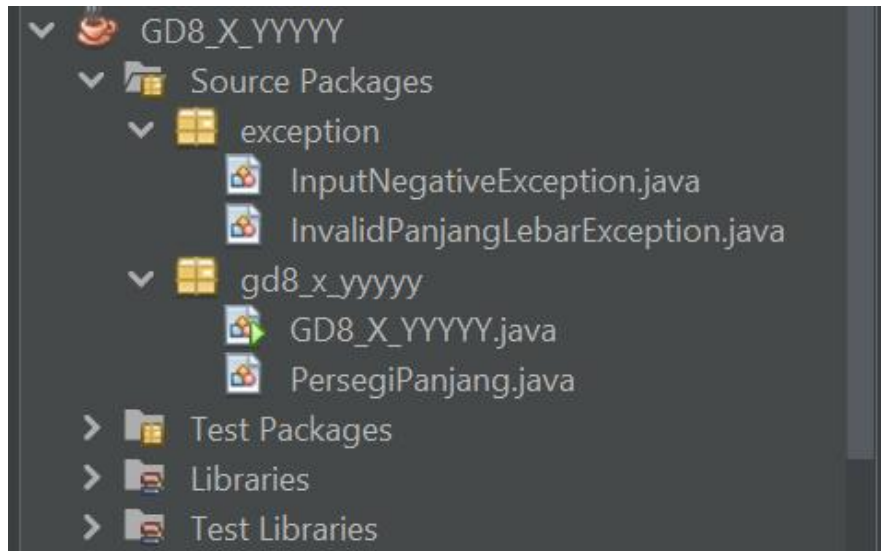
```
run:
Masukkan Panjang Persegi Panjang:
6
Masukkan Lebar Persegi Panjang:
0
Panjang atau lebar Tidak boleh == 0!!
-- Code ini ada di Finally --
BUILD SUCCESSFUL (total time: 6 seconds)
|
```

8. Apabila kalian menginputkan value dengan benar maka program akan mengeluarkan hasil sebagai berikut

```
run:
Masukkan Panjang Persegi Panjang:
8
Masukkan Lebar Persegi Panjang:
4
Panjang PersegiPanjang: 8.0
Lebar PersegiPanjang: 4.0
-- Code ini ada di Finally --
BUILD SUCCESSFUL (total time: 12 seconds)
```

Pada contoh-contoh output di atas kita bisa melihat bahwa pada keyword **Finally** akan tetap terpanggil pada saat terjadi Exception atau tidak.

9. Berikut ini adalah susunan file yang dituliskan pada guided ini. Pembuatan package baru bernama “exception” bertujuan untuk pengelolaan file agar mudah dilihat dan dipahami.

**KETENTUAN PENGUMPULAN GUIDED:**

Format pengumpulan adalah GD8_X_YYYYYY.zip (X=Kelas, YYYYYY=5 Digit terakhir NPM)