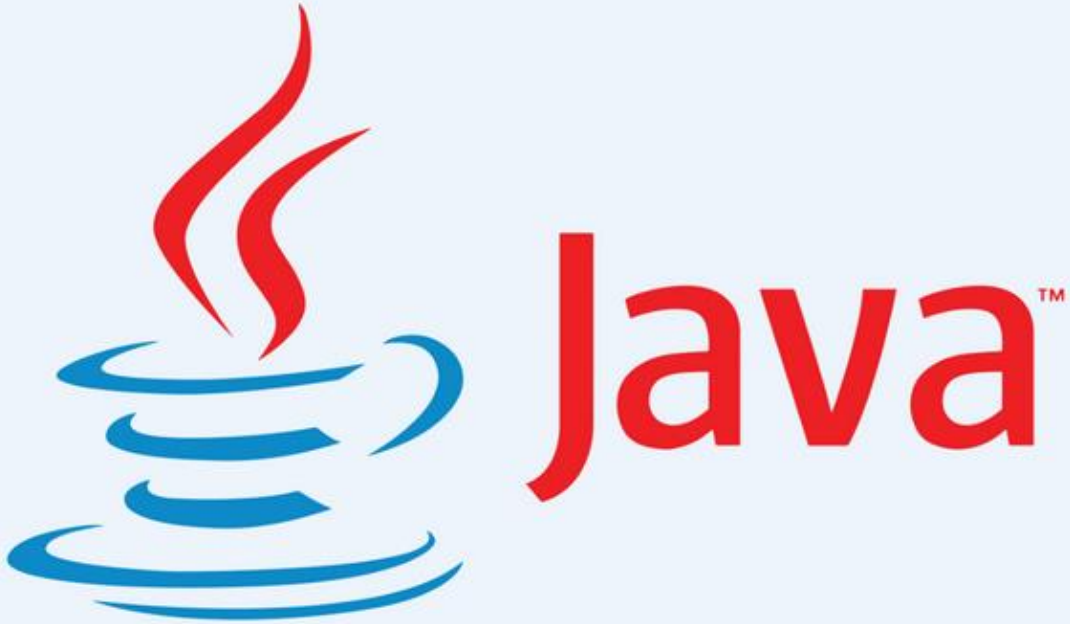
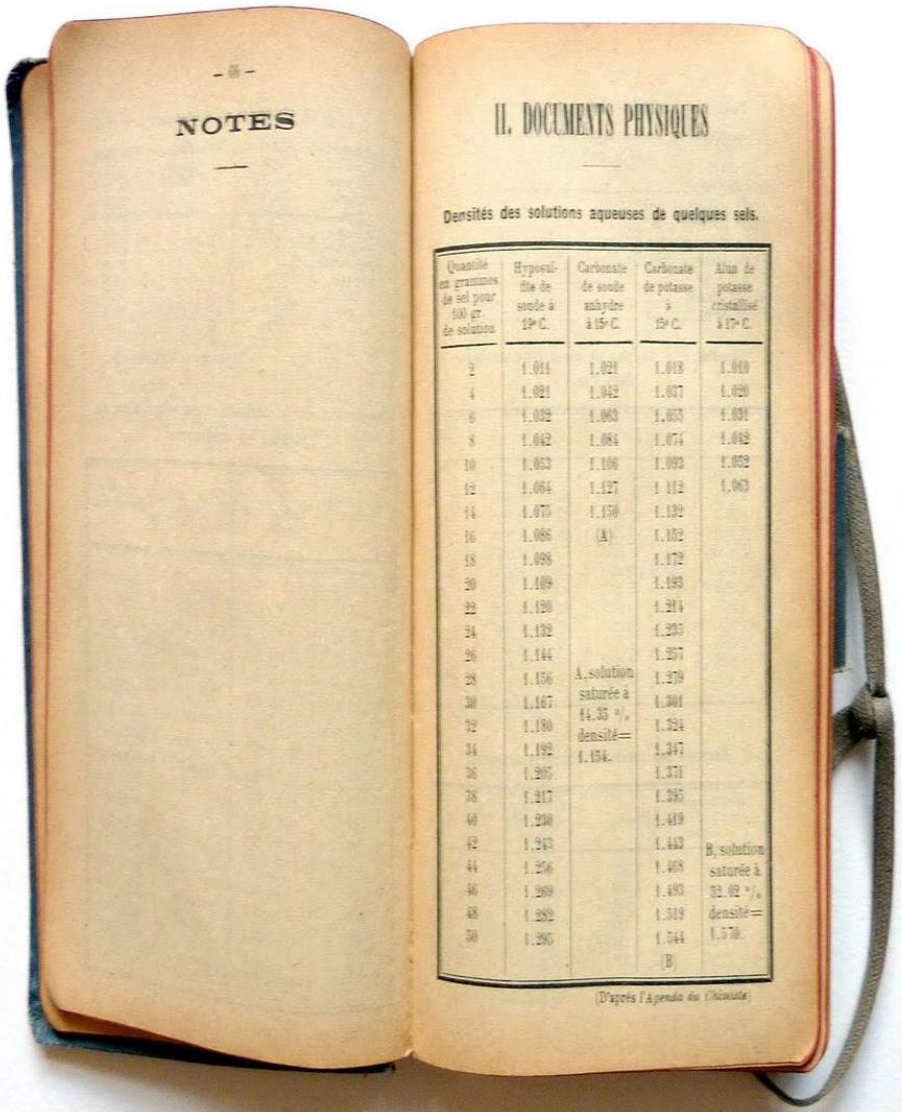
Pemrograman Berorientasi Objek



Pertemuan 11

Exception & Error Handling

Pemateri : Chrismikha Hardyanto S.Kom., M.Kom.



KONTEN PERKULIAHAN

- ***Running Time Error*** **didalam Program**
- **Konsep Dasar** ***Exception***
- ***Exception Handling*** **pada JAVA**
  - **Block Try & Catch**
  - **Block Finally**
- **Melempar** ***Exception*** **Dari Method (Throw)**

Didalam pembuatan sebuah program, salah satu hal yang perlu dipikirkan oleh programmer adalah bagaimana menangani error yang mungkin terjadi didalam program

Latar Belakang

- Pembangunan sebuah perangkat lunak tidak akan bisa lepas dari yang namanya **error.**
- Error mungkin saja terjadi ketika kita menuliskan kode – kode dari program **(Coding Error)** atau ketika program dijalankan **(Runtime Error)**
- Apabila terjadi error didalam program, umumnya akan menyebabkan **alur (flow) program** tidak berjalan seperti semestinya. Terburuknya program akan **berhenti running(force close/stop)**
- Program yang baik adalah program yang **jika terjadi ERROR**, sebisa mungkin ia **tidak berhenti running**

Runtime Error

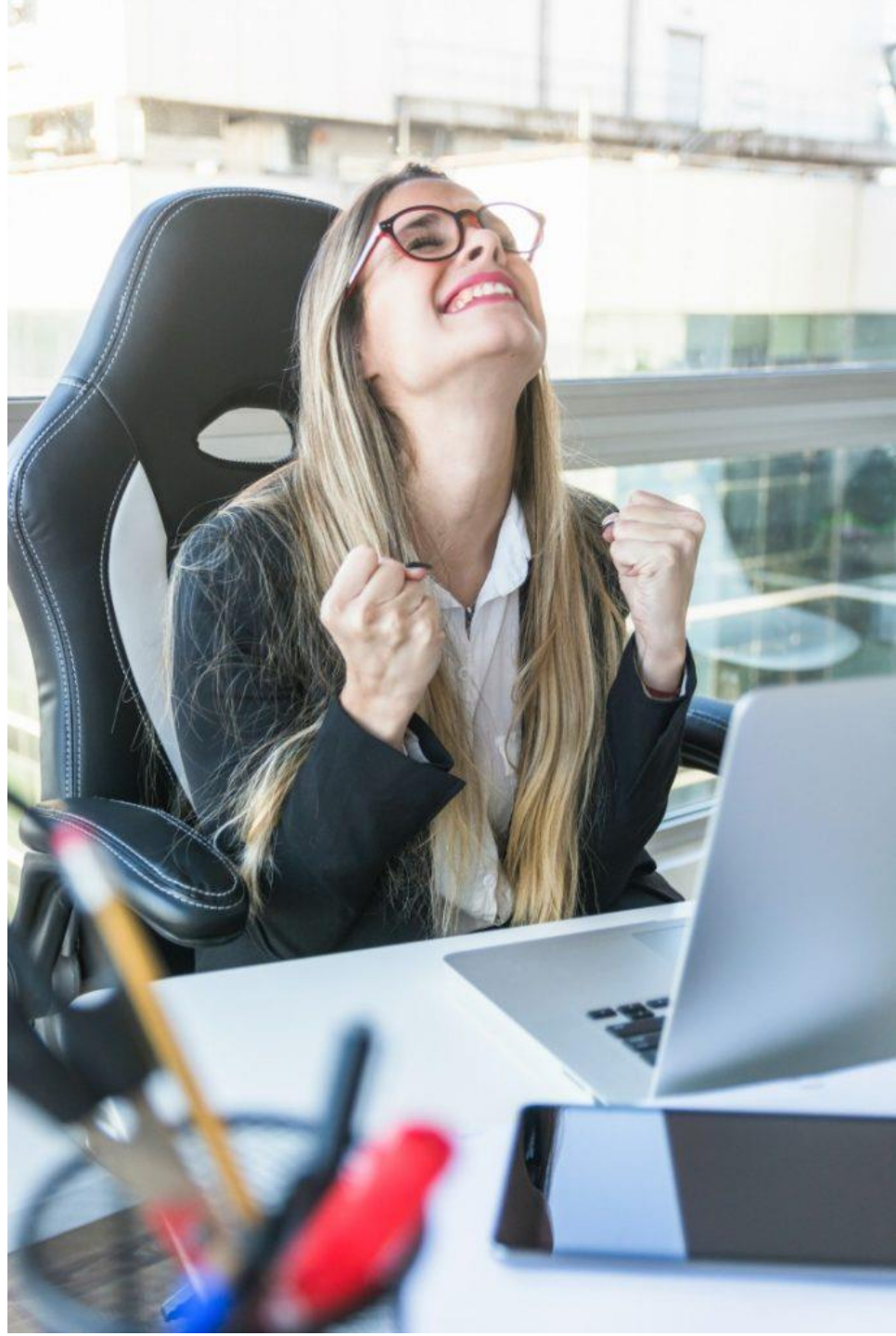
- **Runtime error** adalah error yang terjadi ketika **program sedang berjalan (running)**. **->** Secara kode tidak ada masalah.
- Error jenis umumnya disebabkan oleh aksi dari **pengguna (user)** program, walaupun tidak menutup jika disebabkan oleh perilaku **internal program**
- Jika penyebab runtime error didalam program **tidak ditangani**, maka akan menyebabkan program **tidak berjalan dengan baik** atau fatalnya menyebabkan **crash (force close)**
- error ini sulit untuk **diprediksi**. Sebagai programmer kita tidak bisa menebak kapan error terjadi selama program sedang **berjalan** (running time)

Apa itu Exception

Di JAVA, **kondisi/perilaku** yang menyebabkan program tidak bisa berjalan normal (abnormal running time) disebut sebagai **EXCEPTION**

- **Exception** = **Pengecualian** (kondisi tidak normal)
- Contoh **Perilaku** yang menyebabkan Exception pada JAVA :

1. **Operasi Pembagian dengan bilangan 0 (aritmathic exception)**
2. **Input nilai namun berbeda Tipe Data (InputMismatchException)**
3. **Menyisipkan elemen diluar rentang dari array(Out of bound exception)**
4. **File/Class tidak ditemukan (IO exception)**
5. **dll**

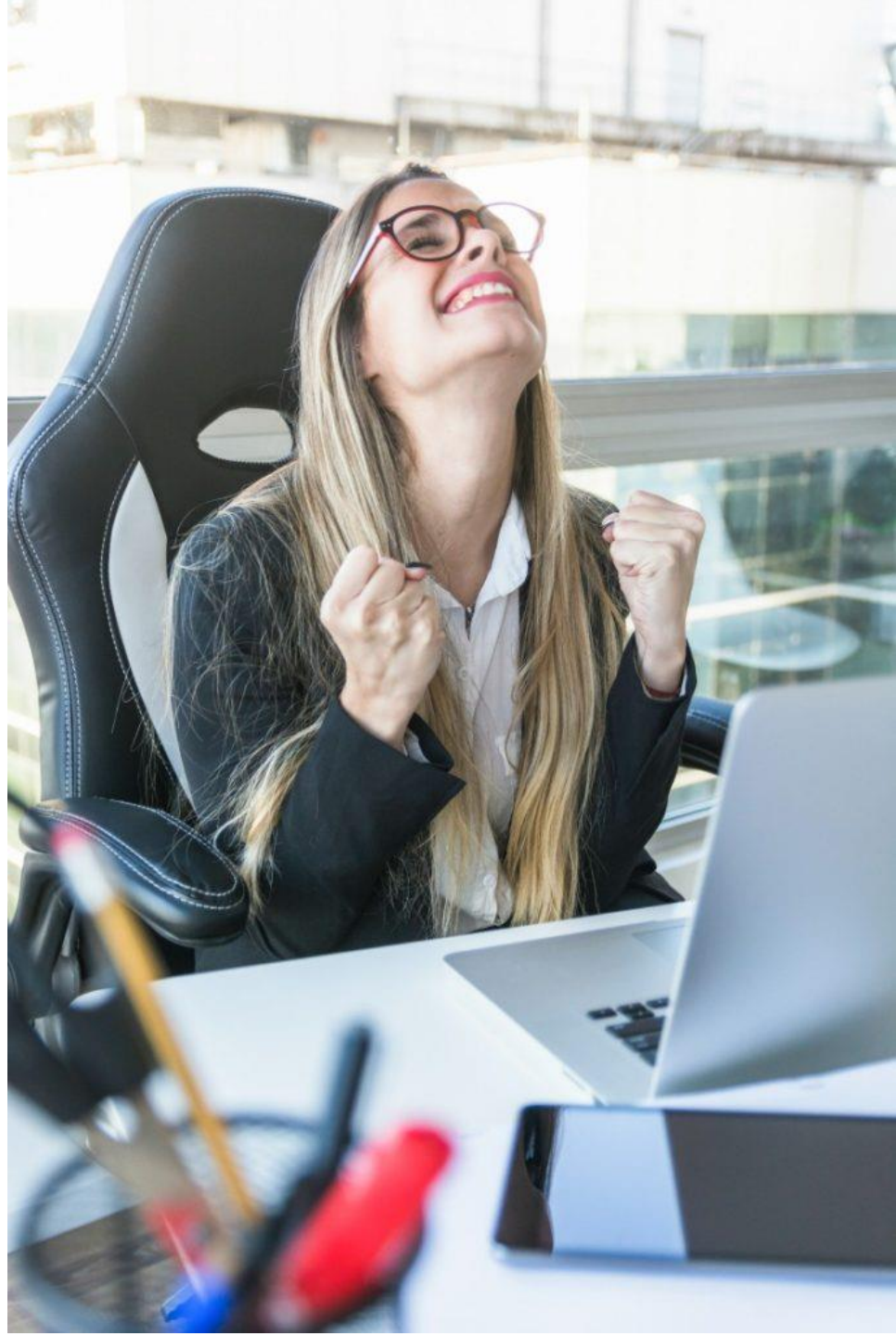


Exception Handling

Exception Handling (Penangan eksepsi), merupakan suatu **mekanisme** pada JAVA **untuk menangani kondisi** yang menyebabkan **runtime error** pada program

Keuntungan utama dari menerapkan penagangan eksepsi adalah untuk **mempertahankan aliran program** selalu **berjalan normal** walaupun **terjadi runtime error**.

*Agar lebih paham mari kita lihat ilustrasi berikut



Ilustrasi Konsep Exception

```
statement 1;
statement 2;
statement 3;
statement 4;
statement 5;//exception occurs
statement 6;
statement 7;
statement 8;
statement 9;
statement 10;
```

- Misalkan Ada **10 Statement** di Dalam sebuah program JAVA. Jika Flow normal maka seluruh statement dapat dieksekusi. Namun misalkan **terjadi exception pada statement ke-5**, Maka **statement 6-10 menjadi tidak dapat dieksekusi** (program force close)

Contoh Exception pada JAVA

- Untuk melihat bagaimana **exception terjadi** didalam program, buatlah program dengan kode sebagai berikut : (buat sebuah project)

```
1  package tutorial1;
2  import java.util.Scanner;
3
4  public class Main {
5
6      public static void main(String[] args) {
7          //Coba Simulasi Exception
8          Scanner inputBilangan = new Scanner(System.in);
9
10         System.out.println("MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN");
11         System.out.println("==========================================\n");
12         System.out.print("BILANGAN ke-1      : ");
13         int bilangan1 = inputBilangan.nextInt();
14         System.out.print("BILANGAN ke-2      : ");
15         int bilangan2 = inputBilangan.nextInt();
16         double hasilBagi = (double)bilangan1 / (double)bilangan2;
17         System.out.println("");
18         System.out.println("HASIL BAGI BILANGAN : "+hasilBagi);
19         System.out.println("PROGRAM DITUTUP NORMAl");
20     }
21 }
```

Secara penulisan coding, contoh program ini tidak ada masalah (**build successful**), coba running programnya

Contoh Exception pada JAVA

- Dapat dilihat ketika program dijalankan (running time) , program ini dapat berjalan tampa ada error

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
==============================================

BILANGAN ke-1      : 10
BILANGAN ke-2      : 10

HASIL BAGI BILANGAN : 1.0
PROGRAM DITUTUP NORMAl
BUILD SUCCESSFUL (total time: 6 seconds)
```

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
==============================================

BILANGAN ke-1      : 10
BILANGAN ke-2      : 30

HASIL BAGI BILANGAN : 0.3333333333333333
PROGRAM DITUTUP NORMAl
BUILD SUCCESSFUL (total time: 27 seconds)
```

Contoh Exception pada JAVA

- Bagaimana jika user memasukan nilai bilangan **bukan dengan angka (tipe data integer) ???**

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
================================================


BILANGAN ke-1      : 10
BILANGAN ke-2      : dua
```

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
================================================

BILANGAN ke-1      : 10
BILANGAN ke-2      : dua
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:864)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at tutorial1.Main.main(Main.java:15)
C:\Users\Mika\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 1 minute 14 seconds)
```

Nama Class Exception yang terjadi

{Terjadi **RUNTIME ERROR**}

Contoh Exception pada JAVA

- Bagaimana jika user memasukan nilai untuk **bilangan ke-2 dengan nilai 0 ???**

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
================================================


BILANGAN ke-1      : 10
BILANGAN ke-2      : 0
```

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
================================================

BILANGAN ke-1      : 20
BILANGAN ke-2      : 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at tutorial1.Main.main(Main.java:16)
C:\Users\Mika\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 17 seconds)
```

Nama Class Exception yang terjadi

{Terjadi **RUNTIME ERROR**}

*__note :__ modifikasi dulu coding program dengan menghapus perintah **cast tipe data double** pada operasi pembagian

Menerapkan Exception Handling

- Untuk menerapkan exception handling di JAVA, Cara yang paling mudah adalah dengan menggunakan Blok **Try & Catch**

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.

*__sumber :__ JavaPoint

Bentuk Umum Try & Catch

- Berikut adalah bentuk umum pembuatan block Try & catch pada JAVA :

[Block Jika hanya 1 **Catch**]

```
try {

        //Statement (baris – baris kode) yang dapat menyebabkan exception
} catch(ClassException object) {
        //Statement untuk penanganan exception
}
```

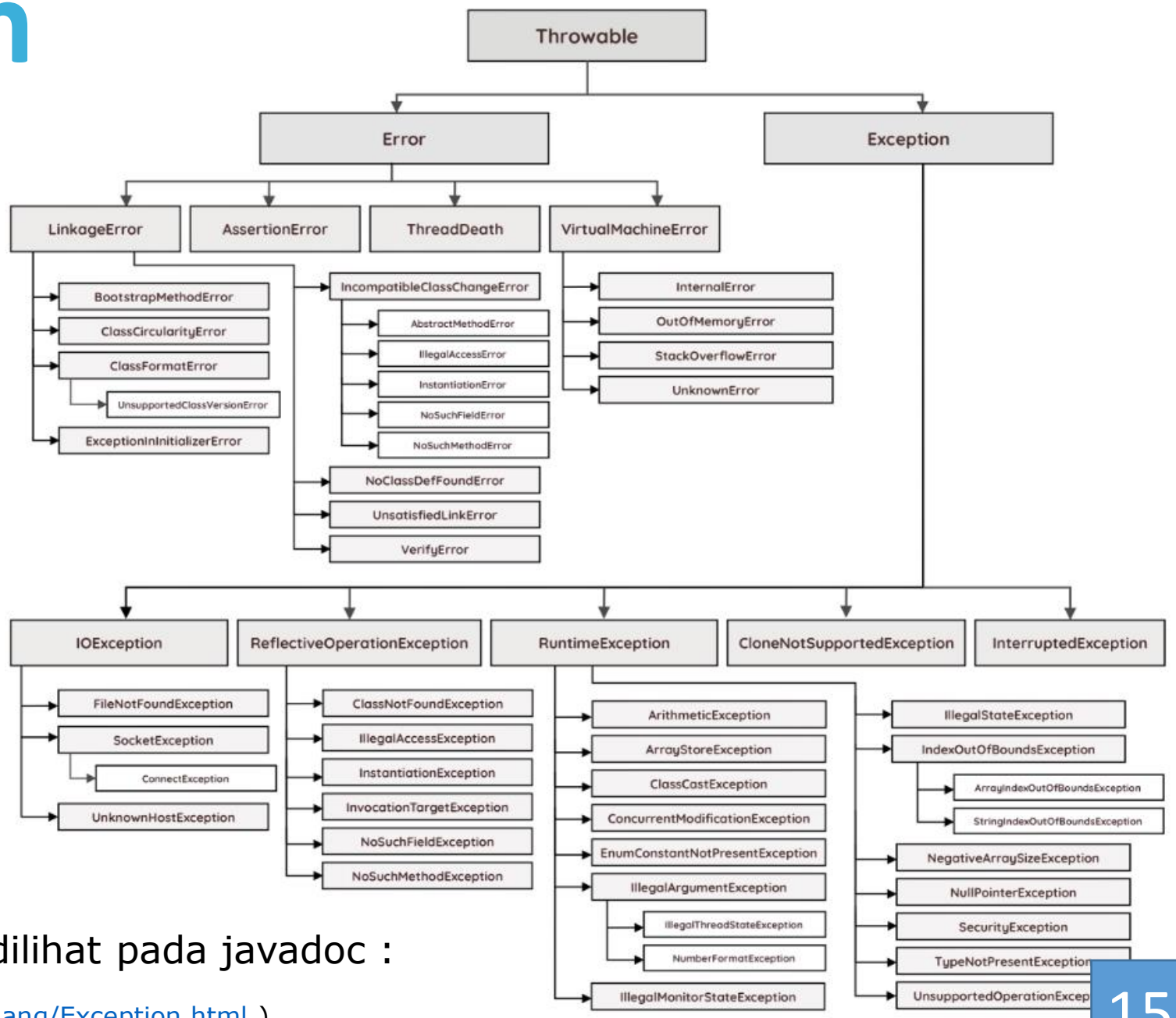
[Block Jika lebih dari 1 **Catch**]

```
try {
        //Statement (baris – baris kode) yang dapat menyebabkan exception
} catch(ClassException ke-1 object-1) {
        //Statement untuk penanganan exception ke-1
} catch(ClassException ke-n object-n) {
        //Statement untuk penanganan exception ke-n
}
```

Class Exception

- Mengetahui dan memahami jenis class exception di JAVA sangat bermanfaat ketika kita akan menangani exception didalam program

- Object dari Class – Class inilah yang akan dipanggil didalam catch (sesuai dengan jenis exception yang akan ditangani)



- Dokumentasi class exception dapat dilihat pada javadoc :

(link: https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html)

Contoh Exception Handling pada JAVA

- Berikut adalah contoh exception handling (block try & catch) untuk menangani permasalahan di kasus sebelumnya :

```
4   public class Main {
5
6       public static void main(String[] args) {
7           //Coba Simulasi Exception
8           Scanner inputBilangan = new Scanner(System.in);
9
10          System.out.println("MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN");
11          System.out.println("================================================\n");
12          try{
13              //Statement Yang Mungkin Menyebabkan Exception
14              System.out.print("BILANGAN ke-1      : ");
15              int bilangan1 = inputBilangan.nextInt();
16              System.out.print("BILANGAN ke-2      : ");
17              int bilangan2 = inputBilangan.nextInt();
18              System.out.println("");
19              double hasilBagi = bilangan1 / bilangan2;
20              System.out.println("HASIL BAGI BILANGAN : "+hasilBagi);
21
22          }catch(Exception e){
23              //Statement penanganan exception (umumnya tampilkan message)
24              System.out.println("\nTerjadi Exception " +e);
25          }
26
27          System.out.println("PROGRAM DITUTUP NORMAl");
28      }
29  }
```

[Block **Try**]

[Block **Catch**]

Contoh Exception Handling pada JAVA

- Berikut adalah hasil dari program setelah diterapkan penanganan exception didalam kodenya. Dapat dilihat bahwa ketika terjadi exception, **program tetap running sampai selesai**

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
================================================

BILANGAN ke-1      : sepuluh

Terjadi Exception java.util.InputMismatchException
PROGRAM DITUTUP NORMAl
BUILD SUCCESSFUL (total time: 16 seconds)
```

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
================================================

BILANGAN ke-1      : 10
BILANGAN ke-2      : 0


Terjadi Exception java.lang.ArithmeticException: / by zero
PROGRAM DITUTUP NORMAl
BUILD SUCCESSFUL (total time: 4 seconds)
```

Contoh Exception Handling pada JAVA(2)

- Berikut adalah contoh exception handling jika ada lebih dari 1 exception untuk menangani permasalahan di kasus sebelumnya :

```
7       public static void main(String[] args) {
8           //Coba Simulasi Exception
9           Scanner inputBilangan = new Scanner(System.in);
10
11          System.out.println("MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN");
12          System.out.println("================================================\n");
13          try{
14              //Statement Yang Mungkin Menyebabkan Exception
15              System.out.print("BILANGAN ke-1     : ");
16              int bilangan1 = inputBilangan.nextInt();
17              System.out.print("BILANGAN ke-2     : ");
18              int bilangan2 = inputBilangan.nextInt();
19              System.out.println("");
20              double hasilBagi = bilangan1 / bilangan2;
21              System.out.println("HASIL BAGI BILANGAN : "+hasilBagi);
22
            }catch(InputMismatchException e){
24              //Statement penanganan exception (umumnya tampilkan message)
25              System.out.println("\nTerjadi Exception " +e);
26          }catch(ArithmeticException e){
27              System.out.println("\nTerjadi Exception " +e);
28          }
29
30          System.out.println("PROGRAM DITUTUP NORMAl");
31      }
32  }
```

[Block **Try**]

[Block **Catch ke-1**]

[Block **Catch ke-2**]

Contoh Exception Handling pada JAVA(2)

- Berikut adalah hasil dari program setelah diterapkan penanganan exception didalam kodenya. Dapat dilihat bahwa ketika terjadi exception, **program tetap running sampai selesai**

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
================================================

BILANGAN ke-1      : sepuluh

Terjadi Exception java.util.InputMismatchException
PROGRAM DITUTUP NORMAl
BUILD SUCCESSFUL (total time: 16 seconds)
```

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
===============================================

BILANGAN ke-1      : 10
BILANGAN ke-2      : 0


Terjadi Exception java.lang.ArithmeticException: / by zero
PROGRAM DITUTUP NORMAl
BUILD SUCCESSFUL (total time: 4 seconds)
```

Bagaimana Jika Exception terjadi didalam sebuah **method**

Bagaimana Penanganannya didalam kode program ??

Contoh Exception Didalam Method

- Untuk kasus ini, buatlah sebuah class dengan nama kalkulator. Dan isilah class tersebut dengan atribut dan method seperti contoh berikut :

```
4   public class Kalkulator {
5
6       private int bilangan1;
7       private int bilangan2;
8
9       public int getBilangan1() {...3 lines }
12      public void setBilangan1(int bilangan1) {...3 lines }
15
16      public int getBilangan2() {...3 lines }
19      public void setBilangan2(int bilangan2) {...3 lines }
22
23      public void tambahBilangan(){
24          Scanner inputBilangan = new Scanner(System.in);
25
26          System.out.print("BILANGAN 1     : ");
27          this.setBilangan1(inputBilangan.nextInt());
28          System.out.print("BILANGAN 2     : ");
29          this.setBilangan2(inputBilangan.nextInt());
30      }
31
32      public double hasilBagi(int bil1, int bil2) {
33          double hasilBagi;
34          hasilBagi = bil1 / bil2;
35
36          return hasilBagi;
37      }
38
39      public void tampilHasilPembagian(){
40          double hasilBagi = hasilBagi(this.bilangan1, this.bilangan2);
41          System.out.println("HASIL BAGI BILANGAN : "+hasilBagi);
42      }
43  }
```

[method **tambahBilangan**]

[method **Hasil Bagi**]

[method **tampilHasilPembagian**]

Contoh Exception Didalam Method

- Buatlah **object dari class kalkulator** di class Main Anda . Lalu panggil seluruh method yang ada di class calculator. Selanjutnya running program tersebut & lihat apa yang terjadi

```
4   public class Main {
5
6       public static void main(String[] args) {
7           System.out.println("MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN");
8           System.out.println("==========================================\n");
9           Kalkulator kalkulator = new Kalkulator();
10          kalkulator.tambahBilangan();
11          kalkulator.tampilHasilPembagian();
12          System.out.println("\n");
13          System.out.println("PROGRAM DITUTUP NORMAL");
14      }
15
16  }
```

Program pasti akan error dan terjadi exception (sama seperti contoh sebelumnya)

Contoh Exception Didalam Method

- Cara penanganan exception didalam method : **terapkan block try & catch pada class** dimana **method (yang menyebabkan exception)** tersebut dipanggil

```
4   public class Main {
5
6       public static void main(String[] args) {
7           System.out.println("MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN");
8           System.out.println("=============================================\n");
9           Kalkulator kalkulator = new Kalkulator();
10          try{
11              kalkulator.tambahBilangan();
12          }catch(InputMismatchException e){
13              System.out.println("Terjadi Exception - " +e);
14          }
15
16          kalkulator.tampilHasilPembagian();
17          System.out.println("\n");
18          System.out.println("PROGRAM DITUTUP NORMAL");
19      }
20  }
```

->**Dapat Menyebabkan Exception**

Contoh Exception Didalam Method (2)

- Cara penanganan exception didalam method : **terapkan block try & catch pada class** dimana **method (yang menyebabkan exception)** tersebut dipanggil

```
32      public double hasilBagi(int bil1, int bil2) {
33          double hasilBagi;
34          hasilBagi = bil1 / bil2;
35
36          return hasilBagi;
37      }
38
39      public void tampilHasilPembagian(){
40          try{
41              double hasilBagi = hasilBagi(this.bilangan1, this.bilangan2);
42              System.out.println("HASIL BAGI BILANGAN : "+hasilBagi);
43          }catch(ArithmeticException e){
44              System.out.println("Terjadi Exception - " +e);
45          }
46      }
```

->**Dapat Menyebabkan Exception**

->**Dipanggil**

Membuat Class Exception (Custom)

Bagaimana Jika kita ingin membuat class exception sendiri , **apakah bisa ??**

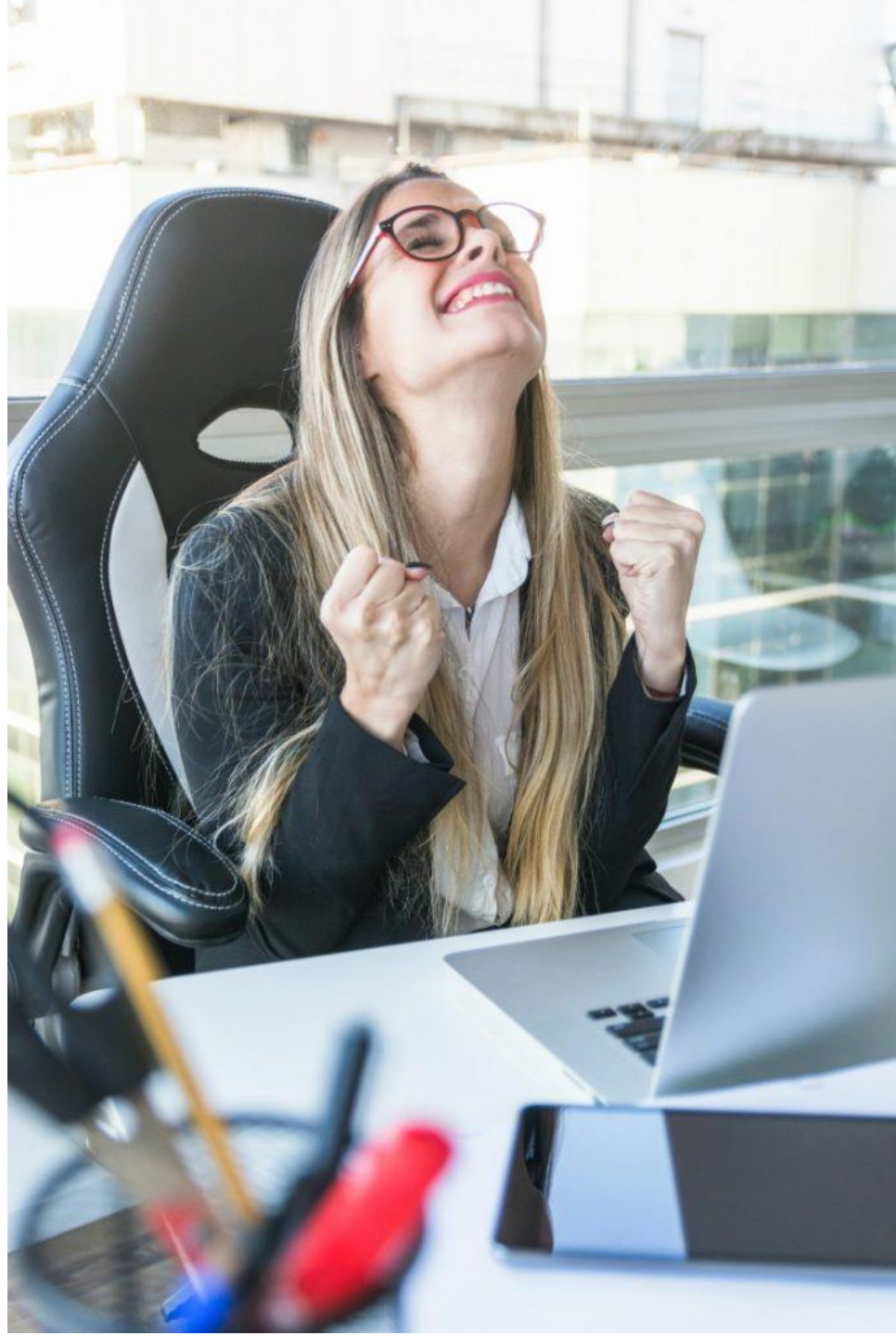
Custom Exception Class

- Tidak semua semua class exception yang telah didefinisikan oleh JAVA itu **mendukung semua kebutuhan** kita dalam pembuatan program

- **Contoh Kasus :** Untuk nilai suatu mata kuliah, hanya boleh 0 sampai 100. Java tidak memiliki class exception untuk menangani hal tersebut.

Solusi ??

Buatlah class exception sendiri



Custom Exception Class

- Untuk membuat class exception sendiri, Anda harus membuat Class yang merupakan **turunan dari class throwable** atau **class turunan-nya**

```
public class <Nama_Kelas>  extends <Exception> {

        @Override Construktor
        public <Nama_Kelas> (Parameter){
                super(parameter);
        }
        //Badan Kelas Exception Yang lain

}
```

Contoh Membuat Class Exception

- Perhatikan kasus berikut, Ketika terjadi Error maka akan ada pesan exception yang diberikan. Namun bagaimana jika kita ingin pesan tersebut **lebih spesifik ?**

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
=============================================

BILANGAN 1    : 10
BILANGAN 2    : angka
Terjadi Exception - java.util.InputMismatchException
Terjadi Exception - java.lang.ArithmeticException: / by zero


PROGRAM DITUTUP NORMAL
BUILD SUCCESSFUL (total time: 1 minute 43 seconds)
```

[Maksudnya bagaimana ??]

Contoh Membuat Class Exception

- **Langkah 1** : Mari kita buat sebuah class dengan nama ValidateNumberException (nama bebas sesuai kebutuhan). Lalu isi class tersebut dengan kode sebagai berikut :

```
1  package tutorial2;
2
3  public class ValidateNumberException extends Exception {
4
5      //Buat construktor dari Class ini untuk men-set pesan ketika object dibuat
6      public ValidateNumberException(String message){
7          super(message);
8      }
9  }
10
```

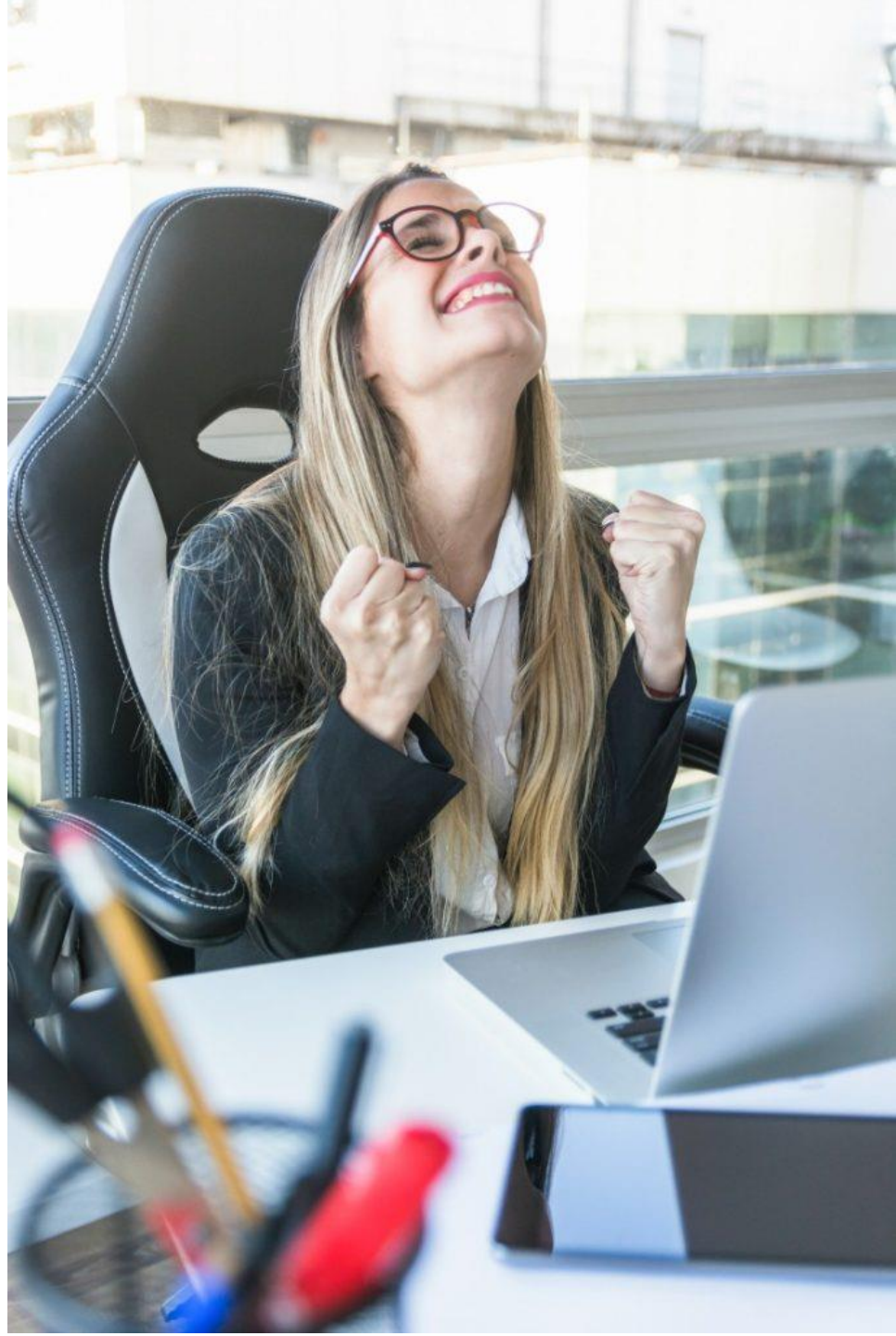
Selanjutnya kita coba panggil class exception ini didalam main Program untuk **menggantikan class exception InputMismatchException**. Coba perhatikan apa yang terjadi …

Melempar Exception

- Ketika Kita membuat sebuah **Class Exception Baru** (dan merupakan turunan dari exception), Maka untuk bisa menggunakannya class tersebut harus **dilempar (throw).**
- **Throwing Exception** adalah suatu operasi untuk **melemparkan sebuah exception pada method** ketika exception terjadi (**menandai method**)
- Untuk melakukan proses tersebut ada 2 kata kunci yang harus kita gunakan, yaitu **throws** & **throw**

throws : digunakan untuk menandai method

throw : digunakan untuk melempar object dari class Exception



Melempar Exception (Throwing)

- Berikut adalah bentuk umum bagaimana mekanisme melempar object class exception dari suatu method

```
public void <Suatu_Method>  throws <Class_Exception> {

        //... Isi dari Method

        throw new Class_Exception(Argument);
}
```

- **Contoh :**

```
public void tambahBilangan throws ValidateNumberException {

        //... Isi dari method tambahBilangan
        String message = "...";
        throw new ValidateNumberException(message);
}
```

Contoh Membuat Class Exception(2)

- **Langkah 2** : Mari kita tambahkan kode berikut pada method **tambahBilangan()** didalam class kalkulator (karena method ini yang melemparkan exception)

```
23
24      public void tambahBilangan() throws ValidateNumberException{
25          Scanner inputBilangan = new Scanner(System.in);
26          try{
27              System.out.print("BILANGAN 1     : ");
28              this.setBilangan1(inputBilangan.nextInt());
29              System.out.print("BILANGAN 2     : ");
30              this.setBilangan2(inputBilangan.nextInt());
31          }catch(InputMismatchException e){
32              String message = "Bilangan Harus Bernilai INTEGER";
33              throw new ValidateNumberException(message);
34          }
35      }
```

-> ketika terjadi MisMatch Exception

-> lempar object exception

Contoh Membuat Class Exception(2)

- Ketika program dijalankan dan terjadi Exception (input bilangan = String), maka pesan exception yang muncul adalah sebagai berikut :

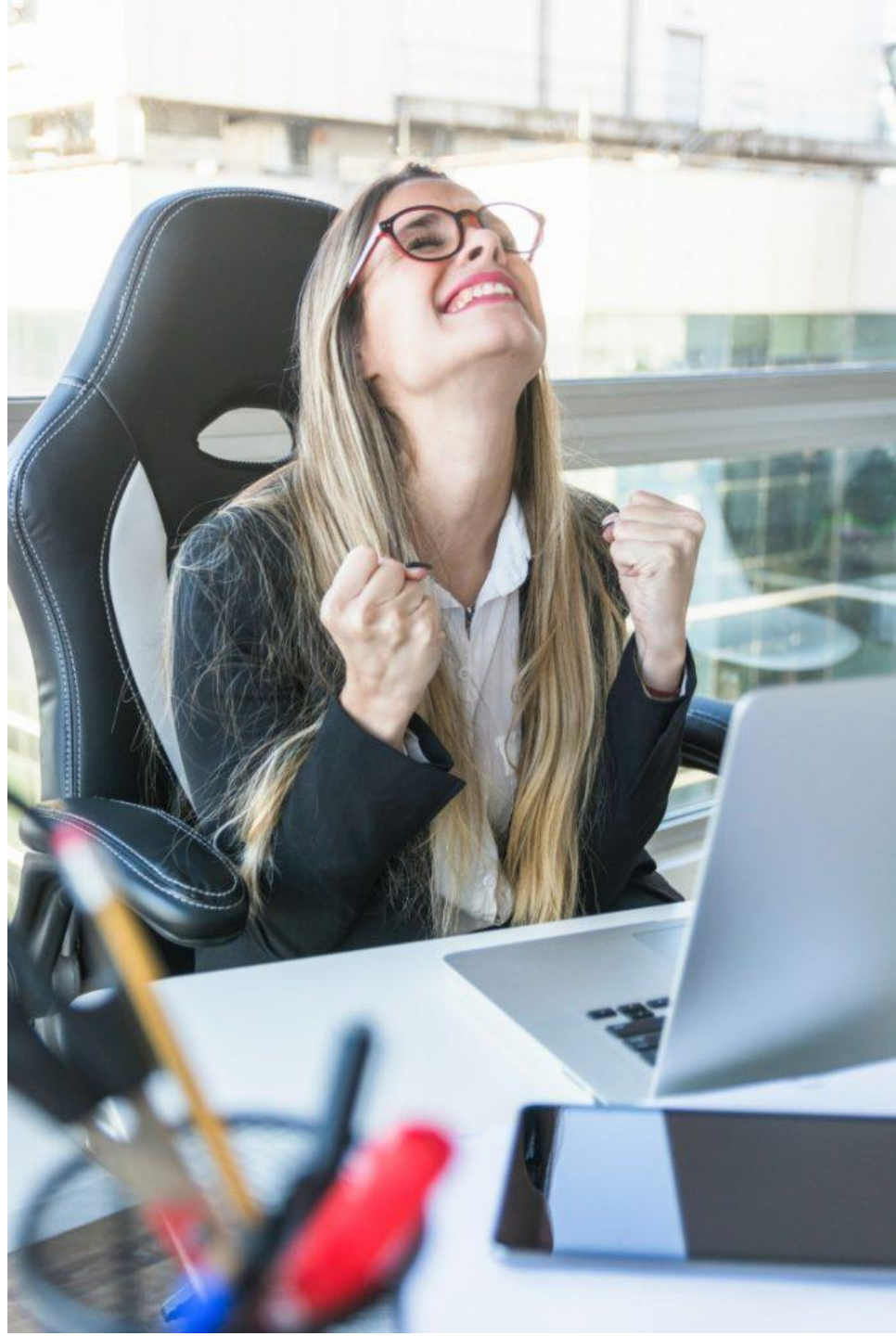
```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
================================================

BILANGAN 1    : 10
BILANGAN 2    : dua
Terjadi Exception - tutorial2.ValidateNumberException: Bilangan Harus Bernilai INTEGER
Terjadi Exception - java.lang.ArithmeticException: / by zero


PROGRAM DITUTUP NORMAL
BUILD SUCCESSFUL (total time: 11 seconds)
```

Block Finally

- Didalam pembuatan block try catch, kita bisa menambahkan yang namanya block **finally**
- Block finally adalah block dimana **isinya akan selalu dieksekusi** baik itu terjadi exception ataupun tidak.
- Jika sebuah exception terjadi, maka **baris-baris di blok try** setelah **baris terjadinya exception** pasti **tidak akan tereksekusi** karena alur program akan pindah ke blok catch.
- Jika baris tersebut ingin tetap dijalankan ketika terjadi exception atau pun tidak, maka tulislah perintah-perintah tersebut di **bagian finally**.



Membuat Block Finally

- Berikut adalah bentuk umum pembuatan block finally pada JAVA:

[Block finally diletakan setalah **Catch** dan **hanya boleh ada 1 block finally**]

```
try {

        //Statement (baris – baris kode) yang dapat menyebabkan exception
} catch(ClassException object) {

        //Statement untuk penanganan exception
} finally {

        //Statement yang ingin selalu diekeskusi ketika terjadi exception atau tidak
        //cth : menutup resource / koneksi
}
```

Contoh Block Finally pada JAVA

- Berikut adalah block finally untuk menangani permasalahan di kasus sebelumnya :

```
7   public static void main(String[] args) {
8       //Coba Simulasi Exception
9       Scanner inputBilangan = new Scanner(System.in);
10
11      System.out.println("MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN");
12      System.out.println("=============================================\n");
13      try{
14          //Statement Yang Mungkin Menyebabkan Exception
15          System.out.print("BILANGAN ke-1     : ");
16          int bilangan1 = inputBilangan.nextInt();
17          System.out.print("BILANGAN ke-2     : ");
18          int bilangan2 = inputBilangan.nextInt();
19          System.out.println("");
20          double hasilBagi = bilangan1 / bilangan2;
21          System.out.println("HASIL BAGI BILANGAN : "+hasilBagi);
22          //Tambahkan Kode berikut didalam block try
23          inputBilangan.close();
24          System.out.println("tutup Scanner");
25
26      }catch(InputMismatchException e){
27          //Statement penanganan exception (umumnya tampilkan message)
28          System.out.println("\nTerjadi Exception " +e);
29      }catch(ArithmeticException e){
30          System.out.println("\nTerjadi Exception " +e);
31      }finally{
32          //Memastikan apapun flow yang terjadi didalam program kode ini pasti dijalankan
33          inputBilangan.close();
34          System.out.println("tutup Scanner");
35      }
36
37      System.out.println("PROGRAM DITUTUP NORMAl");
38  }
39  }
```

[tambahkan kode berikut]

[menerapkan **block finally**]

Contoh Exception Handling pada JAVA(2)

[Program Berjalan Normal]

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
===============================================

BILANGAN ke-1      : 10
BILANGAN ke-2      : 2

HASIL BAGI BILANGAN : 5.0
tutup Scanner
tutup Scanner
PROGRAM DITUTUP NORMAl
BUILD SUCCESSFUL (total time: 4 seconds)
```

[Program Mengalami Exception]

```
run:
MASUKAN 2 BUAH BILANGAN UNTUK OPERASI PEMBAGIAN
===============================================

BILANGAN ke-1      : sepuluh

Terjadi Exception java.util.InputMismatchException
tutup Scanner
PROGRAM DITUTUP NORMAl
BUILD SUCCESSFUL (total time: 8 seconds)
```

Terima Kasih