



UNIVERSITAS  
INDONESIA

*Veritas, Probitas, Iustitia*

FACULTY OF  
COMPUTER  
SCIENCE

# TRY NOT TO KILL PEOPLE WITH YOUR PROGRAM



## (Basic Exception Handling in Java)

Erika Meliawati (1806146934) - Ilma Ainur Rohma (1806141220) - Thami Endamora (1806141460)

Lecture: Fariz Darari, S.Kom, M.Sc., Ph.D. - Teaching Assistant : Muhamad Lutfi Arif

## INTERMEZZO

### WHY?

In 1985 and 1986, 1 person was killed and several were injured by excess radiation due to a misprogrammed computerized radiation machine. So, programs that don't work can lead to serious consequences. Improved error recovery is one of the most powerful ways to increase the robustness of your code, so the main goal of learning exception handling is to create a robust system.

### WHAT IS IT?

Let's see this Terminal display

```
C:\Learn Java>java UncheckedException
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index 1 out-of-bounds for length 0
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:64)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:70)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:248)
    at java.base/java.util.Objects.checkIndex(Objects.java:372)
    at java.base/java.util.ArrayList.get(ArrayList.java:440)
    at UncheckedException.main(UncheckedException.java:5)

C:\Learn Java>javac CheckedException.java
CheckedException.java:3: error: incompatible types: int cannot be converted to String
    String a = 0;
               ^
1 error
```

Sometimes it happens when we code with Java, that is called exception. So, an exception is some cases or type of error that we might want to handle. Exception handling enables the program to deal with exceptional situations and continue its normal execution.

### CHECKED VS UNCHECKED

Because system errors and runtime errors can happen to any code, Java does not require that you declare Error and Runtime Exception (unchecked exceptions) explicitly in the method. However, all other exceptions thrown by the method must be explicitly declared in the method header so that the caller of the method is informed of the exception.

## HOW TO HANDLE



A program that detects an error can create an instance of an appropriate exception type and throw it. This is known as throwing an exception.

```
public class CircleWithException {

    private double radius;
    // Construct a circle with a specified radius
    public CircleWithException(double newRadius) {
        setRadius(newRadius);
        System.out.println("Circle with radius " + newRadius + " created");
    }

    // Set a new radius
    public void setRadius(double newRadius) throws IllegalArgumentException {
        if (newRadius >= 0)
            radius = newRadius;
        else
            throw new IllegalArgumentException();
    }
}
```

When an exception is thrown, it can be caught and handled in a try-catch block, like this code below.

```
public static void main(String[] args) {
    try {
        CircleWithException c1 = new CircleWithException(5);
        CircleWithException c2 = new CircleWithException(-5);
        CircleWithException c3 = new CircleWithException(0);
    } catch (IllegalArgumentException ex) {
        System.out.println("Radius cannot be negative!");
    }
}
```

The program will execute the try block until the code ends or until it finds an exception. If it found an exception, the catch code block would be executed.

```
C:\DDP 2\KELAS>java CircleWithException
Circle with radius 5.0 created
Radius cannot be negative!
```

We can see in the terminal that only the first object constructed correctly and prints out the "Circle with radius 5.0 created". When constructing the second object with radius below 0, it throws an exception and caught in the main method, prints out the "Radius cannot be negative!". Codes in the try block that declared in the line after the program finds an exception will not be executed (in this example the c3 will not be constructed).

## MAKE OUR EXCEPTION

In Java, we can make our own exception according to what we need. Before we step forward, please make sure that you understand about exception handling to make this more fun to learn. Now, we start with an example :

Blood Donor program has a rule that the donor's weight should not under 47 kg. So, you have to make a program that can throw an exception if the weight under 47 kg for blood donor requirement.

So, we make an exception class that extends Exception class, this class at least have a constructor:

```
public class UnderWeightException extends Exception{
    public UnderWeightException (String message){
        super (message);
    }
}
```

Our exception is ready to use. Now how do we use our custom exception? You can make a method that throws your exception and in the main method you have to handle it with try-catch expression. Let's see this code below:

```
public class TryMyException{
    public static void checkWeight(int weight) throws UnderWeightException{
        if(weight < 47){
            throw new UnderWeightException("Sorry you are Underweight");
        }
    }

    public static void main(String[] args) {
        try{
            checkWeight(20);
        } catch(UnderWeightException e){
            e.printStackTrace();
        }
    }
}
```

Now let's check our terminal:

```
C:\DDP 2\KELAS>java TryMyException
UnderWeightException: Sorry you are Underweight
    at TryMyException.checkWeight(TryMyException.java:7)
    at TryMyException.main(TryMyException.java:13)
```

as we can see in the terminal, our custom exception works well.



#### References :

Daniel, L. Y. (2015). Introduction to Java Programming : Comprehensive Version 10th ed. New Jersey: Pearson.

Eck, D. J. (2016). Introduction to Programming Using Java. New York: Hobart and William Smith Colleges.

Oracle. (n.d.). Class Exception. Retrieved from Java Docs:

<https://docs.oracle.com/javase/10/docs/api/java/lang/Exception.html>



BY