Lab Exception Handling CSIS-1410

- Download LabExceptionHandling.java
- Run the program and enter the number 10.
  An exception will be thrown.
  Look at the stack trace to find the type of the exception: _____
  Then fix the problem.
- After the first problem has been fixed run the program again. This time enter 2.5 as number.
  An exception will be thrown. Type of the exception: _____
  Which is the last code statement in 'my' code that was (partially) executed before the exception was thrown? (this it the first entry of 'my' code in the stack trace) _____
  Change the method **`numberFromUser`** so that it keeps reading in numbers until the user provides a valid integer.
  Every time the user enters something invalid (e.g. a floating point number, a string etc.) a message should be printed. (e.g. The number entered needs to be a whole number.)
  Where should the Scanner variable `input` be initialized? Iinside or outside of the loop? Does it matter?
  See answer below.
- Run the program again. Enter first 2.5, then "zero", and then 0 when prompted for a number.
  An exception will be thrown. Type of the exception: _____
  Which is the last code statement in 'my' code that was (partially) executed before the exception was thrown? (the first entry of 'my' code in the stack trace) _____
  This time we don't fix the probem in the body of the method that caused the exception. Instead we re-throw the exception. How can that be done?
  Check whether the parameter is 0. If that is the case throw an IllegalArgumentException.
  As you throw the new exception provide some information about the problem by adding a message: " *`Can't calculate 7 % 0`* " as argument to the IllegalArgumentException constructor.
- Run the program again. This time enter 0 as number
  An exception will be thrown. Type of the exception: _____
  Which is the last code statement in 'my' code that was (partially) executed before the exception was thrown?
  _____
  Notice that the message *`Can't calculate 7 % 0`* is displayed as well as the stack trace
  Change the main method so that all the code from main is wrapped in a try block.
  Add a catch block that catches any exception. In the catch block print the following message:
  `A problem occurred:` followed by the message from the exception.
  Hint: *Every exception inherits the method* `getMessage` *from Throwable. This method returns a string that includes the message that should be displayed.*
  Run the program an enter **`2.5`** then **`hallo`** then **`0`**

SAMPLE OUTPUT:

```
number: 2.5
The number entered needs to be a whole number.
number: hallo
The number entered needs to be a whole number.
number: 0
```

Answer: *`The Scanner variable should be initialized outside of the loop. Why? One instance of Scanner is enough to read in multiple numbers. Creating a new Scanner instance for every number read would be inefficient`*