# Practical 5 - Exception Handling

Download the **MyATUStudentException.java** file from [GitHub](https://github.com/mariagriffin/ATUExceptions)

These exercises are split into different parts in the code. Uncomment the part of the code that you are working on in the main method, and then fix that part.

A screen shot of a computer program

Description automatically generated

## **Exception recap**

1. Un-comment part1 (only) in the main method
2. Compile and run the code.
3. Enter an integer between 1 – 10
4. Run the program again and enter 5.5.
   * Although this number is between 1 and 10, the program will abort. Examine the error message.
5. Put a try/catch block around the code to handle the ***InputMismatchException*** exception.
6. Compile and run the program again, testing with a variety of input (integers, floats, characters). The program should not abort when floats or character data is given.

**Not sure where to add the try/catch??**

**Hint:** Identify the statements that cause the error as well as the portions of the program that depend upon these statements. Enclose these statements within the **try** **block**. Follow the try block with the catch block.

**Note:**  When the Scanner throws an ***InputMismatchException***, the input token will remain in the buffer so that it can be examined by the program. In our case, we will not be examining the token, but will simply clear out of the buffer to start over.

## Section 2 – Working with Runtime exceptions (Parts 2 – 6)

In this section, you are going to trigger and catch built-in exceptions through try-catch-finally block. Using the steps below, you are going to trigger a runtime exception, you will see how the default handler that is provided by the Java runtime system catches the exception.  Then you will then catch the exception yourself using a try-catch block.

### Part 2 . Catch a runtime exception - divide a number by zero

1. Un-comment Part 2 (only) in the main method
2. This part of the code is trying to divide a whole number (3) by zero

**System.out.println(3/0);**

1. Compile and run the program – it will cause an exception.
2. Modify the code, to catch an Exception using a try-catch block.
3. Build and run the program.
4. Modify the code again, to use a more specific Exception class, ***ArithmeticException***
5. Compile and run the program.
6. Add a code fragment that divides a number by 0 inside the catch block and see what happens.

### Part 3 . Catch Multiple Exceptions

In this step, you are going to use multiple catch statements to catch an exception.  If the first catch statement does not catch an exception, the next one will be tried.

1. Un-comment Part 3 (only) in the main method
2. Compile and run the program
3. Observe Input dialog box that appears, **Enter 0 in the field.**
4. Observe the result in the Output window of your IDE.
5. Run the program again, this time enter ‘**somecharacter’** in the dialog box.
6. Observe the result in the Output window of your IDE.
7. **Add a try with multiple catch blocks to catch the exceptions**

### Part 4 - Call built-in methods of Exception class

1. Un-comment Part 4 (only) in the main method
2. Compile and run the program
3. Observe the display message in the Output window of your IDE.
4. Add more print statements to print out the Exceptions :
   1. getMessage()
   2. getLocalizedMessage()
   3. toString():
   4. StackTrace()
5. Change the text of the message to printout your own message.

### Part 5 Create your own Exception

In this step, you are creating your own exception in the class called **MyATUStudentException.**

**(ideally we create a separate class for this)**

1. Un-comment Part 5 (only) in the main method
2. Create two constructors one that is empty and one that takes in a message.

**public MyATUStudentException () {**

**super();  
 }  
   
 public MyATUStudentException (String msg) {  
 super(msg);  
 }**

1. Complete the code for the fake methods.
2. Add your Try & Catch for the fake methods
3. Compile and run the program
4. Observe the display message in the Output window of your IDE.

### Part 6 - Finally

1. Un-comment Part 6 (only) in the main method
2. Compile and run the program
3. What happens?
4. Add a finally block to this code which should always be executed.

**Now try the Challenge Exercise, download the ExceptionsProject from** [**GitHub**](https://github.com/mariagriffin/ATUExceptions)

**Coding Challenge:**

This is a small Java application that processes a text file containing student names and their scores in a subject, separated by commas (e.g., "John Doe, 85"). The application should read the file, calculate the average score of all students, and write the result to a new file. Additionally, the application must handle various exceptions that may arise during the process.

**What the code does:**

1. File Reading and Writing
   1. Read from a file named students.txt
   2. Write the average score to a file named average\_score.txt

1. Data Processing
   1. Parse each line to extract the student's score.
   2. Calculate the average score of all students.

**Requirements:** You need to complete the rest:

**Exception Handling**

* Handle **FileNotFoundException** if the input file does not exist
* Handle **IOException** for other IO errors
* Handle **NumberFormatException** if the score is not a valid integer
* Implement a custom exception called **InvalidDataFormatException** that is thrown when a line does not follow the expected format ("Name, Score")

**Cleanup**

* Ensure that all resources (e.g., file readers and writers) are properly closed, regardless of whether an exception occurs.