Lab 4 (65 pts)

In this lab, you will work with Java **exceptions.** You will learn different types of exceptions, how to handle exceptions and how to recover from them.

Using Eclipse, create a project called Lab4.

Include the exercises in this lab in a package called, **edu.scu.coen160.lab4**

Copy the posted files, **Lab4.java** into your working folder.

What are exceptions? An exception is a problem that arises during the execution of a program. An exception can occur for many different reasons, including the following:

- Invalid user input.
- Division by Zero
- A file that needs to be opened cannot be found.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

There are three categories of exceptions:

Runtime exceptions: A runtime exception is an exception that occurs that probably could have been avoided by the programmer. Runtime exceptions are ignored at the time of compilation.

Checked exceptions: A checked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.

Errors: These are rather severe problems that arise beyond the control of the user or the programmer. There is very little that a programmer can do about these errors. For example, a stack overflow occurs. They are also ignored at the time of compilation.

We will concentrate on **Runtime and Checked exceptions** in the following exercises.

Exercise 1 (10 pts)

Compile and run **Lab4.java**.

Inspect the message that appeared on running the program. What went wrong? Was it a problem with program? Or, was it a problem with data?
From the stack trace, what exception was thrown?
Did it print "**exer1 - end.**" statement?

A Java program may recover from the problem that caused the exception. In order to catch an exception:

1. Put code that might potentially throw an Exception inside a try { } block.
2. Put code that handles the Exception inside a **catch{}** block, following the try block.

See the example in http://docs.oracle.com/javase/tutorial/essential/exceptions/catch.html

Rules for try-catch blocks

1. The statements in the **try{}** block can include:
    - Statements that always work without the possibility of throwing an exception.
    - Statements that might throw an Exception of one type or another.
2. A try block can be followed by one or several catch{} blocks.
    - Each catch block has a parameter (a reference to an exception object) to specify the exception type it is catching
3. If a catch block is missing, you need a finally block, following the try block.

Now, modify the code of exer1 as follows:

Insert a try block and a catch block at the appropriate places. The catch block takes one parameter which gives the type of Exception object that is thrown.

For the parameter to the catch block, check the stack trace of the program when you ran, identify the name of the exception and use that as the exception type for the parameter to the catch block. In the catch block, insert an appropriate message indicating why the problem occurred.

Run your program again.

- What is the output?
- Did it print "exer1 - end." statement?

Exercise 2 (10 pts)

Change the main function so that **Lab4.exer2()** will be executed. Compile and run the code.

```
public static void main(String[] args) {
	//Lab4.exer1();
	//Lab4.exer1_2();
	Lab4.exer2();
	//Lab4.exer2_2();
	//Lab4.exer5(1,"AA123");
	//Lab4.exer5(3,"AA123");
	//Lab4.exer5(2,"12345");
}
```

When you are prompted to enter a number, enter an integer of your choice.

- What is the output?
- Did it print “exer2 - end." statement?

Run the program again. When you are prompted to enter a number, enter the string, “oops”.

- What is the output?
- Did it print “exer2 - end." statement?

Now, modify the code of exer2 as follows:

Introduce a try-catch block with the correct Exception type for the parameter to catch block. Insert appropriate comment in the catch block, to indicate what went wrong.

- Run the program again.
- What is the output?
- Did it print “exer2 - end." statement?

Exercise 3 (10 pts)

You will use **exer1()** again. Copy the code of **exer1()** into **exer1_2()** and remove your try/catch block. If you do not catch the exception where it is being thrown, it can be caught in the method from which it was called.

- **Implement compile and run the program**

Exercise 4 (15 pts)

Copy the code in **exer2** into another static function called, **exer2_2().** You will make this method user-friendly, where the user, on entering wrong input, will be prompted again (until correct input is given).

Modify the code such that when the user enters the wrong input, the exception is caught and the user is prompted again to enter an integer (this should continue until the user enters correct input).

- **Implement, compile and run the program. Test it with wrong input followed by correct input.**

Exercise 5 (20 pts)

In this exercise, you will use user-defined exceptions and a finally block.

- In Java, all exceptions are objects of class **Exception,** you can create your own Exception classes as subclasses of Exception class.
- You can use a **finally{} block** to include code statements that will always run, no matter how the try{} block was exited.
- **Rules for finally block:**
    - Only one finally block is allowed and must follow all catch blocks.
    - **Normal execution**: If try block executes normally, (no exceptions thrown), control goes to the **finally** block and then to the statements following finally block.
    - **Execution with exceptions**:
        - If try block exits with an exception, and if the exception is handled by a catch block, then, to the catch block, **finally** block and to the statements following finally block.
        - If try block exits with an exception, and if the exception is **NOT** handled by a catch block, then, to the **finally** block and the Exception is thrown to the caller.

Change the main function so that **Lab4.*exer5*(1,"AA123")** will be executed.

```
public static void main(String[] args) {
	//Lab4.exer1();
	//Lab4.exer1_2();
	//Lab4.exer2();
	//Lab4.exer2_2();
	Lab4.exer5(1,"AA123");
	//Lab4.exer5(3,"AA123");
	//Lab4.exer5(2,"12345");
}
```

Compile and run the code.

Modify the code in Lab4 as follows:

In the constructors of **MissedFlight** and **CanceledFlight**, check if the parameter flight starts with 2 or 3 letters (upper or lowercase); if not, throw an exception (new Exception()). Write a catch block accordingly to catch this exception.

Refer to String API, for the methods you can use to check the characters in a String.
http://docs.oracle.com/javase/7/docs/api/java/lang/String.html

Add the following two statements/test cases to main after the first **Lab4.*exer5*(1,"AA123")**

```
	Lab4.exer4(3,"AA123");

	Lab4.exer4(2,"12345");
```

Compile and run the code.