Lab 7

Reminder: Your code is to be designed and written by only you and not to be shared with anyone else. See the Course Outline for details explaining the policies on Academic Integrity. Submissions that violate the Academic Integrity policy will be forwarded directly to the Computer Science Academic Integrity Committee.
All materials provided to you for this work are copyrighted, these and all solutions you create for this work cannot be shared in any form (digital, printed or otherwise). Any violations of this will be investigated and reported to Academic Integrity.

Objectives

- Understand the difference between checked and unchecked exceptions
- Throwing/catching checked exceptions

This lab has three parts with many files. For your convenience we have provided them to you as a zip file that contain three folders containing the files for each of the three parts of the lab. Start by downloading the zipfile into your Lab7 folder and unzipping it.
IMPORTANT: **DO NOT submit a zipfile of your completed work** – we will not be able to grade it! Submit the individual files required.
The code you submit MUST compile with both fully uncommented tester in order to be graded.
That is, you **cannot** only submit one part for grading and all parts must compile.

Part 0

In this part of the lab you will be experimenting with checked and unchecked exceptions

1. Open `Exercise1.java` and complete the three tasks marked with TODO tags.
2. Open `Exercise2.java` and complete the five tasks marked with TODO tags.

CHECK POINT – get help from your lab TA if you are unable to complete this part.

Part 1

In this part of the lab you will be inspecting an implementation of the Stack interface that throws exceptions in place of preconditions.

1. Open `Stack.java`, `StackArrayBased.java` and `Lab7Part1Tester.java`
2. Find the methods in the Stack interface that throw the `StackEmptyException`
   Look at the implementation of these methods in `StackArrayBased.java`
3. Look at the calls to the Stack methods in `Lab7Part1Tester.java` and answer the following questions on paper:
   a. Which method calls are wrapped in try/catch blocks
   b. What does the tester do within the catch block if it does not expect to reach the catch block?
4. Search for the TODO tag in `Lab7Part1Tester.java`. In this location, write tests as described that will test whether the exception is thrown when it should be.
5. Update the given implementation of the method `doBracketsMatch` in `Lab7Part1Tester.java`
   a. Uncomment the code within the method
   b. Wrap all calls to the methods that throw exceptions in a try/catch block.
   c. Within the exception catch block, add the line of code appropriate for the logic of the method. That is, if `pop` is called on an empty stack, what should the program do?

NOTE: Be careful here, you do not need to wrap the whole function body in a try/catch. We will be looking for you to demonstrate your understanding of exactly what lines of code need the exception handling and that you know what code should execute when the exception is thrown.
Reminders: Code within your try/catch blocks should be indented and catch blocks should never be left empty!

CHECK POINT – get help from your lab TA if you are unable to complete this part.

Part 2

In this part of the lab you will be inspecting an implementation of a generic Queue interface updating it to throw exceptions in place of the implicit preconditions.

1. Open `Queue.java`, `QueueRefBased.java` and `Lab7Part2Tester.java`
   This is a generic implementation of a Queue.
2. Find the methods in the Queue interface that have the precondition that the Queue must not be empty. For each of these methods, remove the precondition and change the method so it throws a `QueueEmptyException` (provided for you) in place of the precondition. Again, this will cause you to change your code in multiple places for these updated methods:
   a. Signatures of method prototypes in the interface
   b. Signatures of method implementations in the class that implements the interface
   c. Add code to throw the new exception in the correct case in the method implementations
   d. All calls to these methods (in `Lab7Part2Tester.java`) must be wrapped in a try/catch block.
3. Search for the TODO tag in `Lab7Part2Tester.java`. In this place, write tests as described to test whether the exception is thrown when it should be.

REMEMBER: if you get a warning like the following…

```
Note: Lab7Part2Tester.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Do as suggested and recompile as follows (it will indicate the line numbers of where the problem code is):

```
javac -Xlint:unchecked Lab7Part2Tester.java
```

CHECK POINT – get help from your lab TA if you are unable to complete this part.