

Lab 7

Objectives

- Understand the difference between checked and unchecked exceptions
- Throwing/catching checked exceptions
- More practice with stacks and queues

Part 1

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

1. Download the files in the Part1 folder for this lab into your Lab7 folder.
2. Open `Exercise1.java` and complete the three tasks marked with TODO tags.
3. Open `Exercise2.java` and complete the five tasks marked with TODO tags.

CHECKPOINT – Now might be a good time to check-in with the TA if you are aren't sure you have completed the tasks as expected. Remember, please don't hesitate to ask questions if you are unclear about anything.

Part 2

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

1. Download the files for this lab into your Lab7 folder.
2. Open `Stack.java`, `StackArrayBased.java` and `Lab7Part2Tester.java`
3. Find the methods in the Stack interface that throw the `StackEmptyException`
Look at the implementation of these methods in `StackArrayBased.java`
4. Look at the calls to the Stack methods in `Lab7Part2Tester.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?
5. 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 case.
6. Update the given implementations of the methods `reverseString` and `doBracketsMatch` in `Lab7Tester.java`. For each method (**one at a time**) do the following:
 - a. Uncomment the code within the method you are working on.
 - b. Wrap all calls inside the method to the methods that throw exceptions in a try/catch block
 - c. Within the exception catch block, add appropriate code for the logic of the method. That is, if `pop` is called on an empty stack, what should be done in the catch block? What happens in the `reverseString` method's catch block is likely different than it is in `doBracketsMatch`.

CHECKPOINT – At this point you should be able to compile and run `Lab7Part2Tester.java` with `testStackUseFunctions` uncommented. If this isn't working, please ask a TA for support. If you discuss your progress with a TA, make sure you can:

- demonstrate your tests to ensure the exceptions are thrown in the correct case
- how you have used the catch block to control the flow of your program in the methods tested by `testStackUseFunctions` .

Part 3

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 Lab7Part3Tester.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 implement 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 Lab7Tester.java) must be wrapped in a try/catch block.
3. Search for the TODO tag in Lab7Part3Tester.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: Lab7Part3Tester.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 Lab7Part3Tester.java

CHECKPOINT 3 – LAB COMPLETE – Remember to demonstrate your progress to your TA.