Lab 6

Objectives

- Programming experience using a Stack to solve problems in Java
- Programming experience implementing the Queue data structure
- Programming experience using Generics in Java

Part1

In this part of the lab you will using a Stack to solve some problems.

1. Download the files for lab 6 into your Lab6 folder.
2. Compile and run Lab6Tester file. You will see that some of the tests for the stack implementation are failing. Fix the methods in StackArrayBased.java until all of the tests pass.

CHECKPOINT (Ungraded) – Now might be a good time to check-in with the TA if you are unable to fix the errors in StackArrayBased.java.

3. Implement the methods reverseString according to the documentation provided. Make sure to write additional tests to ensure your implantation is correct.

CHECKPOINT (Ungraded) – Now might be a good time to check-in with the TA if you are failing any of the reverseString tests in Lab6Tester.java.

4. Implement the methods doBracketsMatch according to the documentation provided. Make sure to write additional tests to ensure your implantation is correct.

CHECKPOINT (Ungraded) – Now might be a good time to check-in with the TA if you are failing any of the doBracketsMatch tests in Lab6Tester.java.

Part2

Now you will implement a generic Queue.

1. Change the interface (Queue.java) to be of generic type. This will force you to make changes in multiple places
    a. The class that implements the interface and its dependent classes (QueueRefBased.java and QueueNode.java) must be made generic
    b. Any code that creates a new QueueRefBased must now indicate the type of data that Queue will hold. This type must be the object type. For example:
    ```
    Queue<Integer> intQ = new QueueRefBased<Integer>();
    Queue<Character> charQ = new QueueRefBased<Character>();
    ```

    TIP: make changes to one class at a time, compiling/updating until no compilation errors.
    For example, start with Queue.java, repeatedly make the changes/compile until complete:
    ```
    javac -Xlint:unchecked Queue.java
    ```
    Then move on to QueueNode.java and do the same, and finally for QueueRefBased.java.

2. If you have updated your code in all of the necessary places it should compile without warnings.

    NOTE: if you get the warning like the following…
    ```
    Note: Lab6TesterPart2.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 Lab6TesterPart2.java
    ```

3. Complete the implementation of the stubs provided in QueueRefBased.java according to the documentation in the Queue interface and the tests provided.

4. Add tests in Lab6Tester to ensure the generic queue now works with a collection of data elements *other* than integers.

SUBMISSION (Graded) – Submit the **QueueRefBased.java** and **Lab6Tester.java** files into the Lab 6 submission page on ConneX.