Lab 6

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

- Stack data structure
- Generics
- Queue data structure

Part1

In this part of the lab you will be implementing the Stack interface.

1. Download the files for this lab into your Lab6 folder.
2. Implement each of the methods in `StackArrayBased.java` according to the documentation in the `Stack` interface.

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

3. Implement the method `doBracketsMatch` in `StackApplications.java` according to the documentation and tests provided. You **MUST** use a stack in your implementation. Solutions that do not use a stack will be given a 0 grade. Discuss algorithm design with your TA if you are unsure of how to approach this problem.

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

Part2

In this part of the lab you will be implementing 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, in the `testQueue` method of the `Lab6Tester`, update the type of the `Queue` being created. This has been done for you in the commented out code within `testQueue` but we have left some in the uncommented code for you to update.

   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 the same for `QueueRefBased.java` and finally `Lab6Tester.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: Lab6Tester.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 Lab6Tester.java
```

3. Uncomment the code in the `testQueue` method of the `Lab6Tester` that is marked with:
   `//Testing generic queue with Integer and Character type`
   Complete the implementation of the stubs marked with `TODO` tags in `QueueRefBased`.java according to the documentation in the `Queue` interface and the tests provided.

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