Assignment 4

Stacks

Introduction

In this assignment, you will implement a generic stack in Part 1, and then use the stack to simulate a Word Processor in Part 2.

Although there is a tester provided for this assignment, it does not include a comprehensive set of sets for each method. You should add your own tests for any test cases that are not considered.

NOTE: The automated grading of your assignment will include some different and additional tests to those found in the `A4Tester.java` file. For all assignments, you are expected to write additional tests until you are convinced each method has full test coverage.

Objectives

Upon finishing this assignment, you should be able to:

- Implement a reference-based (node) implementation of a stack
- Implement a stack in Java that supports generics
- Solve problems using only the methods available in the Stack ADT

Submission and Grading

Attach `A4Stack.java` and `WordProcessor.java` to the BrightSpace assignment page. Remember to click **submit** afterward. You should receive a notification that your assignment was successfully submitted.

If you chose not to complete some of the methods required, you **must** provide a stub for the incomplete method(s) in order for our tester to compile. If you submit files that do not compile with our tester, you will receive a zero grade for the assignment. It is your responsibility to ensure you follow the specification and submit the correct files. Additionally, your code must not be written to specifically pass the test cases in the tester, instead, it must work on all valid inputs. We may change the input values during grading and we will inspect your code for hard-coded solutions.
This video explains stubs.

Be sure you submit your assignment, not just save a draft. All late and incorrect submissions will be given a zero grade. A reminder that it is OK to talk about your assignment with your classmates, but not to share code electronically or visually (on a display screen or paper). Plagiarism detection software will be run on all submissions.

Instructions

Part 1:

1. Download all of the `.java` files found in the *Assignments > Assignment 4* page on BrightSpace.
2. Read through the documentation provided in the `Stack.java` interface. There is a lot of information there that will help you set up your generic types when implementing the `Stack` interface.
3. Compile and run `A4Tester.java`. Work through implementing each stack method one at a time. Debug the method until all of the tests pass for that method before proceeding to the next method.

Part 2:

1. For Part 2, you will complete the `undo` and `redo` functions in the `WordProcessor` class. The `WordProcessor` class is supposed to roughly simulate the undo and redo functions found in typical Word Processing programs. I have made a short video demonstrating how the program should work once the implementation is complete. **Note:** You do not need to overthink the implementation here, one of the strengths of using a stack to solve a problem is that stack operations are considered intuitive and easy to use.
2. Although using the Word Processor should allow you to test the undo and redo methods, you may still want to write some tests to call `undo` and `redo` directly within the `A4Tester.java` class.

CRITICAL: Any compile or runtime errors will result in a **zero grade** (if the tester crashes it will not be able to award you any points for any previous tests that may have passed). Make sure to compile and run your program before submitting it!