1 Making stacks from queues and *vice versa*

Both *stacks* and *queues* are linear *abstract data types*, but each is designed to order the items differently: stacks use a *last-in first-out (LIFO)* ordering, while queues provide a *first-in first-out (FIFO)* order.

You can implement a queue using stacks, or a stack using queues. For example, to provide a FIFO ordering when using a stack, an implementation would need to pop all of the existing items from the stack, push the new one, and then re-push all of the removed items, leaving the newly added value at the bottom of that stack. One can likewise manipulate the placement of items in a queue such that items can be dequeued in LIFO order.

In this assignment, you will implement a *Stack* interface using queues, and then implement a *Queue* interface using stacks.

2 Getting started

Create a new directory/project for yourself, and download the following ZIP archive:


Extract/copy the files in this archive into your new directory/project. You will find the following pairs of files:

- **AmhQueue.java** and **AmhStack.java**

  Our own, customized Java interfaces for queues and stacks. Each is somewhat simpler than the standard Java interfaces (*Queue* and *Stack*), leaving us with only the essential operations to implement for each.

- **WrapperQueue.java** and **WrapperStack.java**

  Implements the **AmhQueue** and **AmhStack** interfaces, respectively, by using the standard *LinkedList* class. It is a *wrapper*, just relying on a standard class’s implementation of the standard *Queue* and *Stack* interfaces.
• QueueOfStacks.java and StackOfQueues.java

The first implements AmhQueue interface using the WrapperStack class; the second implements AmhStack using the WrapperQueue class. Note that these classes are incomplete. See Section 3 for more on completing them.

• QueueTester.java and StackTester.java

Standalone programs that test basic queue and stack operations, respectively. For example, QueueTester creates both a QueueOfStacks and a WrapperQueue, and then performs operations on both, verifying that they behave identically.

3 Your assignment

Notice that the methods for QueueOfStacks and StackOfQueues are incomplete. Each of the methods that must be implemented to fulfill the interface is empty, with a // TO DO comment marking that they must be completed.

You must complete these methods, thus implementing a queue of stacks and a stack of queues. Specifically:

• Complete QueueOfStacks using only WrapperStack containers to store values internally.

• Complete StackOfQueues using only WrapperQueue containers to store values internally.

You must also thoroughly test your code. The QueueTester and StackTester classes perform only simple tests of the methods that you are implementing, above. You should add to the code in those tester classes, more fully testing the operation of your methods.

Notice that all of your work should be within these four files:

• QueueOfStacks.java

• StackOfQueues.java

• QueueTester.java

• StackTester.java

No other code files should be modified.
4 How to submit your work

Go to GradeScope for our course, where you can submit your work. It will be auto-tested, and you will see whether it *compiles* and *runs* successfully. Again, if the run fails, it won’t tell you why; you need to go back and do more testing yourself. You may submit early and often!

Notice that you should only submit QueueOfStacks.java and StackOfQueues.java. Your tester code should not be submitted; it exists only to help you debug the code that you are submitting.

This assignment is due on Thursday, Feb-25, 11:59 pm EST.