Recitation 2: Stacks and Queues

Due: See T-Square

Important

There are general submission guidelines you must always follow. If you fail to follow any of the following guidelines you risk receiving a **0** for the entire assignment.

1. All submitted code must compile under **JDK 8**. This includes unused code, so don't submit extra files that don't compile.
2. Do not include any package declarations in your classes.
3. Do not change any existing class headers, constructors, or method signatures.
4. Do not add additional public methods when implementing an interface.
5. Do not use anything that would trivialize the assignment. (e.g. don't import/use `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
6. You must submit your source code, the `.java` files, not the compiled `.class` files.
7. After you submit your files redownload them and run them to make sure they are what you intended to submit. You are responsible if you submit the wrong files.

Overview

You are to implement two data structures, a stack and a queue. Each structure will be backed by a supplied linked list. Both your stack and queue must adhere to the stack ADT and queue ADT that we provide. Please note that you will be provided the JUnits that will be used to grade your assignment. This means that if you pass all the JUnit tests given, then you will have completed the assignment. Follow the instructions in this PDF and in the ADT interfaces provided.

Stacks

A stack is similar to a stack of trays at the dining hall. It only makes sense to retrieve from the top. Similarly, when new trays are added, they are placed on top. Thus, we can think of a stack as LIFO, or last in first out. Your implementation will require you to implement the various functions of a LinkedList to serve as a backing structure. See the StackADT interface for more instructions.

Stack: Push and Pop

To add and remove from a stack, you use push and pop. Push adds an item at the end of a stack, while pop removes an item from the end.

Queues

A queue is similar to a line at an amusement park. Whoever is first in line gets to enter first. Similarly, when new people join the line, they are placed at the back. Thus, we can think of a queue as FIFO, or first in first out. Your implementation will require you to implement the various functions of a LinkedList to serve as a backing structure. See the QueueADT interface for more instructions.

Queue: Enqueue and Dequeue

To add and remove from a queue, you use enqueue and dequeue. Enqueue adds an item to the end of a queue, while dequeue removes and item from the front.

Javadocs

You do not need to write any Javadocs.

CheckStyle

CheckStyle does not need to be run on this assignment.

Forbidden Statements

You may not use these in your code at any time in CS 1332.

- `break` may only be used in switch-case statements
- `continue`
- `package`
- `System.arrayCopy()`
- `clone()`
- `assert()`
- `Arrays` class
- `Array` class
- `Collections` class
- Reflection APIs

Debug print statements are fine, but should not print anything when we run them. We expect clean runs - printing to the console when we're grading will result in a penalty. If you use these, we will take off points.

Provided

The following file(s) have been provided to you. There are several, but you will only edit one of them.

1. `QueueADT.java` This is the interface you will implement when designing your queue. All instructions for what the methods should do are in the javadocs. **Do not alter this file.**
2. `StackADT.java` This is the interface you will implement when designing your stack. All instructions for what the methods should do are in the javadocs. **Do not alter this file.**
3. `Queue.java` This class is where you will be implementing your queue. **Do not add any public methods to this file.**
4. `Stack.java` This class is where you will be implementing your stack. **Do not add any public methods to this file.**
5. `QueueStackTest.java` This is the test class that will be used to grade your assignment.
6. `LinkedList.java` This class is your provided backing structure. You may only use this class as your backing structure. **Do not alter this file**
7. `Node.java` This class is for the `LinkedList` to use only. **Do not alter this file**

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder, copy over the interfaces, recompile, and run. It is your responsibility to re-test your submission and discover editing oddities, upload issues, etc.

1. `Queue.java`
2. `Stack.java`

You may attach each file individually or submit them in a zip archive.