COMP 2140 Assignment 3: Stacks and Queues

Cuneyt Akcora

Due: Monday, March 9, 2020 at 09:00 p.m.

Hand-in Instructions

Go to COMP2140 in UM Learn; then, under "Assessments" in the navigation bar at the top, click on "Assignments". You will find an assignment folder called "Assignment 3". Click the link and follow the instructions. Please note the following:

- In this assignment we have a driver class (Assignment3.java) and a few other classes that we describe below. You will implement code in two files (also mentioned below). Do not rename the existing methods in classes. You will submit both of these files.
- Submit two .java files only. The .java files must contain all the source code that you wrote and the small report (in the comments at the end of the file). The first .java file must be named `A3<your last name><your first name>.java` (e.g., `A3AkcoraCuneyt.java`). The second file will be named MyQueue.java.
- Please do not submit anything else.
- We only accept homework submissions via UM Learn. Please DO NOT try to email your homework to the instructors or TAs or markers — it will not be accepted.
- We reserve the right to refuse to grade the homework or to deduct marks if you do not follow instructions.
- **Assignments become late immediately after the posted due date and time.** Late assignments will be accepted up to 49 hours after that time, at a penalty of 2% per hour or portion thereof. After 49 hours, an assignment is worth 0 marks and will no longer be accepted.

How to Get Help

Your course instructor is helpful: For our office hours and email addresses, see the course website on UM Learn (on the right side of the front page).

For email, please remember to put "[COMP2140]" in the subject and add a meaningful phrase after that to the subject, and to send from your UofM email account.

Course discussion groups on UM Learn: A discussion group for this assignment is available in the COMP 2140 course site on UM Learn (click on "Discussions" under "Communications"). Post questions and comments related to Assignment 1 there, and we will respond. We will also post questions related to the assignment that we receive by email, and answer them there. Please do not post solutions, not even snippets of solutions there, or anywhere else. **We expect you to read the assignment discussion group.**

Computer science help centre: The staff in the help centre can help you (but not give you assignment solutions!). See their website at http://www.cs.umanitoba.ca/undergraduate/computer-science-help-centre.php for location and hours. You can also email them at helpctr@cs.umanitoba.ca.

Programming Standards

When writing code for this course, **follow the programming standards**, available on this course's website on UM Learn. Failure to do so will result in the loss of marks.

Question

Remember: You will need to read this assignment many times to understand all the details of the program you need to write.

Goal: The purpose of this assignment is to write Java programs and implement various queue and linked list algorithms. Then you will write a brief report describing your results, in the comments of your program; see the end of this document.

Code you can use: You are permitted to use and modify the code your instructor gave you in class the completed assignments and labs. Of course, you must use the code we provided to you below. You are NOT permitted to use code from any other source, You are NOT permitted to show or give your code to any other student. Any other code you need for this assignment, you must write for yourself. We encourage you to write ALL the code for this assignment yourself, based on your understanding of the algorithms — you will learn the material much better if you write the code yourself.

Files: Familiarize yourself with the code in the following files.

1. `MyQueue.java`: You are to edit this file to implement a queue using two stacks as described below. This is the first file to be submitted.
2. `MyStack.java`: This class implements a stack, with the usual operations push, pop, and isEmpty. You should not edit this file.
3. Node.java: This class implements a linked-list node. This code is similar to that used in Lab 2, Lab 3, and Assignment 2. You should not edit this file.
4. `EmptyStackQueueException.java`: This class implements an exception thrown when an attempt is made to perform a pop or dequeue operation on an empty stack or queue. This code is similar to that used in Lab 3. You should not edit this file.
5. `Assignment3.java`: This code shows an example of how to instantiate and use the class MyQueue. You may edit this file to help you test your code. This is the second file to be submitted.

What you should implement:

1. You are to edit the file MyQueue.java as described below. A queue can be implemented using two stacks, stack1 and stack2. The enqueue operation pushes the new item onto stack1, while the dequeue operation pops the top item from stack2 and returns it. If stack2 becomes empty, the content of stack1 is moved to stack2 such that its order on stack2 is the reverse from what it was on stack1. Note that this is not an efficient implementation of a queue; you saw in lecture how to implement enqueue and dequeue with O(1) worst-case time. The goal of this assignment is for you to examine how to use the existing functionality of stacks to solve other problems, such as implementing a queue.
    - Add declarations for two instance variables stack1 and stack2 of type `MyStack<E>`. Both should be declared protected.
    - Within the constructor, add two statements that initialize stack1 and stack2 by assigning them respective references to two new instances of the class `MyStack<E>`, each of which is a new empty stack.
    - Add code to the method enqueue such that it pushes elem onto stack1.
    - Add code to the method transferStacks to move the content of stack1 to stack2 such that its order on stack2 is the reverse from what it was on stack1. stack1 should be empty after a call to transferStacks.

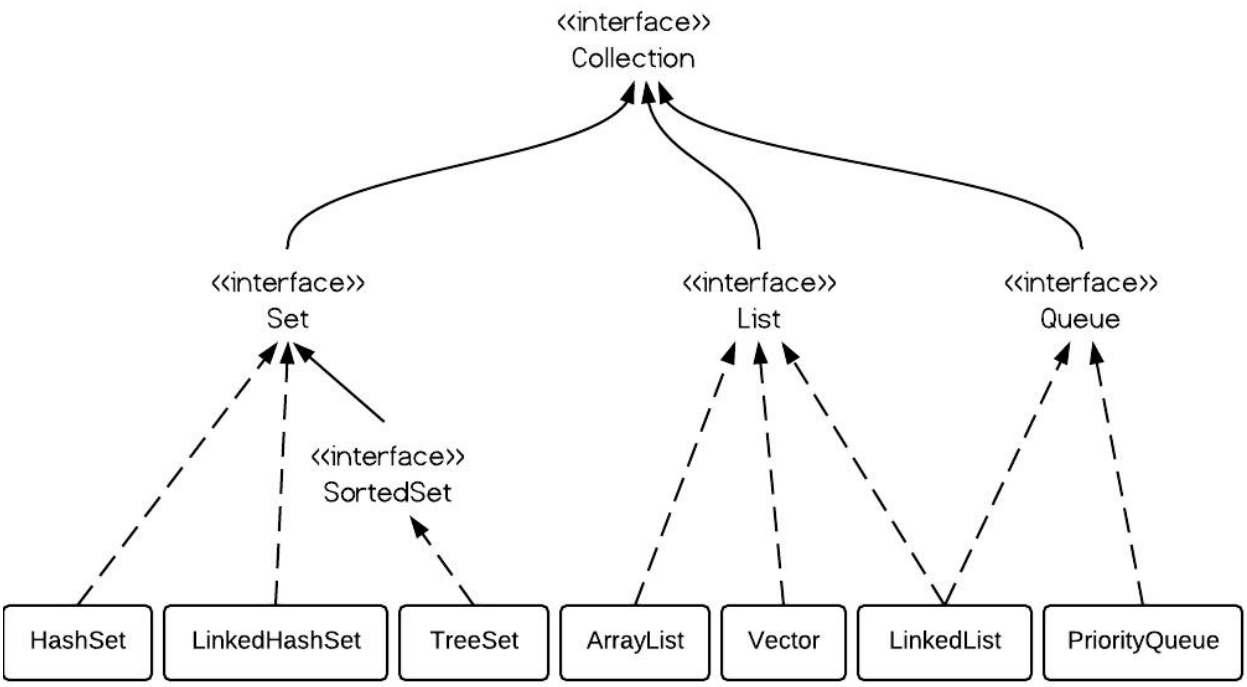

Figure 1: Java collection hierarchy. See https://docs.oracle.com/javase/tutorial/collections/interfaces/index.html. Note that Map is omitted in this image (ask yourself, why?). Image from dzone.com

- Add code to the method dequeue to a) check whether stack2 is empty and, if so, to call transferStacks, and b) to pop the top item from stack2 and store it in result.
- Add code to the method isEmpty such that it sets the value of result to true if and only if the queue is empty. Your code should only call accessor methods on the stacks.

2. You are to edit the `A3<your last name><your first name>.java`. Please write your code in the main function where indicated. Most of these questions are to show you the utility of existing collections (Figure 1).

   - 2a. Manually search the ArrayList class (you can find the implementation by going to the source code, or searching and reading the class file online) to find the function that can be used to get the element at a given index.
   - 2b. Search the ArrayList class to find a way (more than 1 way exist) to add all books in store1 to store2 with a single line of code.
   - 2c, 2d and 2e below do not require any coding.
   - 2c. In your report at the end of the assignment, write one sentence about the difference between linkedlist and arraylist.
   - 2d. In your report at the end of the assignment, write one sentence about the difference between store3 and store4. What are the methods that can be called on store4, but not on store3?
   - 2e. In your report at the end of the assignment, write two sentences about the difference between store3 and store4. What is the advantage of using the LinkedList declaration in store4? in turn, what is the disadvantage?

Concerns.

Please avoid magical numbers in your code. There should be only ONE return per method. See the programming standards (under content/course documents) file on UMLearn, and follow the suggestions. Assignments will be graded by considering these standards. Your code must not give any warnings or errors when compiled and run.

Report

Write a small report in comments at the end of your program.