Lab 8 – Stacks and Queues

Due October 11th at 23:59

Objective

Today's lab will help you get familiar with

- Stacks
- Queues
- Interfaces
- Generics

Stacks:

A **stack** is a type of data structure in which the addition and deletion of elements occur only at one end, called the **top** of the stack. Elements at the bottom have been in the stack the longest. The top element is the last element added to the stack. Thus, the item that is added last will be removed first. For this reason, a stack is also called a Last In First Out (LIFO) data structure. See Section 6.6.1 of the textbook for more information.

Queues:

Another important data structure is the **queue**. In a queue, elements are added at one end, called the **back** or **rear**, and deleted from the other end, called the **front**. The rear is accessed whenever a new element is added to the queue, and the front is accessed whenever an element is deleted from the queue. Therefore, a queue is a First In First Out (FIFO) data structure. See Section 6.6.3 of the textbook for more information.

Assignment:

1. Create a generic stack interface called StackInterface<E>. This interface will have the following methods declared.
    a. boolean empty() – Tests if stack is empty
    b. E peek() – Looks at the object at the top of the stack without removing it.
    c. E pop() – Removes the object at the top of the stack and returns the object as the value of this function.
    d. E push(E item) – Pushes an item onto the top of the stack.
    e. int search(Object o) – Returns the distance from the top of the stack of the occurrence that is nearest the top of the stack. The topmost item is considered to be at distance 1; the next item is at distance 2; etc.
2. Create a generic stack class called MyStack<E>, which implements StackInterface<E>. This will be based around the Node<E> class in (5) below, as opposed to the lower quality Node class available in Five Below.
3. Create a generic queue interface called QueueInterface<E>. This interface will have the following methods declared.
    a. E add(E item) – Inserts the specified element into the queue.
    b. E peek() – Retrieves, but does not remove, the head of the queue.
    c. E remove() – Retrieves and removes the head of the queue.
4. Create a generic queue class called MyQueue<E>, which implements QueueInterface<E>. Again, use (5) below.
5. Create a Node<E> class and a linked list implementation for building the stack and queue (you may reuse code from Lab 3 or 4).
6. Write an ExperimentController for evaluating these classes.
    a. Insert random elements (type of your choosing) into your data structures.
        - Try inserting elements into your stack, then move them over to the queue, and vice versa.
    b. Evaluate the performance of your methods (i.e. stack vs queue).
7. Unit test your classes.

Submission:

In addition to your code, you must submit lab notes (in PDF format) and include JavaDoc documentation. Save your lab notes in the project folder before you compress it and upload it. Details about the lab notes can be found on the CS150L Moodle page.

Grading:

1. Unit testing (for MyStack and MyQueue classes) – 1 pt
2. MyStack class – 2 pts
3. MyQueue class – 2 pts
4. Node class/linked list – 1 pt
5. Interfaces – 1 pt
6. ExperimentController Class – 1 pt
7. Style/commenting/JavaDoc – 1 pt
8. Lab Notes – 1 pt