Assignment Prefix: Lab05

Due Date: Tuesday, Oct. 4th @ 11:59pm

Points: 100

This is an individual assignment.

Restrictions: you cannot use any methods from the Java Array(s) class to copy an array, check for equality, or otherwise manipulate an array. You must write the Java code to perform these functions.

The above restrictions do not apply to any of the classes copied from the textbook.

Create a NetBeans project named Lab05 and ensure it is saved to a location like desktop or your flash drive. In the project you will do the following:

For this assignment you are going to implement both the Stack and the Queue abstract data type statically and dynamically. Implement the following code that is covered in Chapter 6 of the textbook:

- The Stack interface from Code Fragment 6.1
- The Array-based implementation of a Stack from Code Fragment 6.2
- The LinkedStack implementation of a Stack using a SinglyLinkedList from Code Fragment 6.4
    - You will need to implement the SinglyLinkedList from code Fragments 3.14 and 3.15.
- The Queue interface from Code Fragment 6.9
- The Array-based implementation of a Queue from Code Fragment 6.10
- The LinkedQueue implementation of a Queue from code Fragment 6.11

For the above interfaces and classes, you can transcribe (i.e. type in) the code right out of the textbook. As you enter the code make sure that you understand what is going on. You do not need to comment any code that you copied from the textbook.

As you read the text that accompanies the Code Fragments you should realize that Stacks and Queue are very efficient Abstract Data Types (ADTs). A question you might consider is which is more efficient, a static array-based implementation or a dynamic linked list-based implementation.

Develop a client program that will compare the two implementations by measuring the run time and memory requirements for a fairly simple task, using a Stack to reverse the order of a Queue.

For your Array-based implementations:

- Create a Queue of Integers of size N.
- Fill the Queue with random integers between -100 and +100 inclusive.
- Create a Stack of Integers of size N.
- Start your timer
- Deque an entry from the Queue and push it onto the Stack, repeat this process until the Queue is empty.
- Now pop an entry from the Stack and enqueue it into the Queue, repeat this process until the Stack is empty.
- Stop your timer
- Record the value of N and the elapse time

For your LinkedList-based implementations:

- Create a Queue of Integers.
- Fill the Queue with N random integers between -100 and +100 inclusive.
- Create a Stack of Integers.
- Start your timer
- Deque an entry from the Queue and push it onto the Stack, repeat this process until the Queue is empty.
- Now pop an entry from the Stack and enqueue it into the Queue, repeat this process until the Stack is empty.
- Stop your timer
- Record the value of N and the elapse time

Vary the size of N, starting with 1,024 and doubling the size of N each time until you run out of system memory. Write your code so that your program does not crash when it runs out of memory.

Display the results of all of your tests in a nicely formatted ASCII table