Computer Science Department/College of Engineering and Computer Science

CSc 20: Programming Concepts and Methodology II

Lab 7 – Stack and Queue ADTs (Spring 2018)

Objective:

This lab will give you a practice with both queue and stack ADTs. In this work, you are to determine if an input string is a palindrome. A string of characters is a palindrome if and only if it reads the same forward and backward. Examples: eye, abba, civic, radar, and so on.

Preparation: (at home)

Write the output produced by the following method when passed each of the following stacks and ints. Note: A stack displays/prints in {bottom ... top} order, and a queue displays in {front ... back} order.

```java
public void collectionMystery10(Stack<Integer> stack, int n) {
    Stack<Integer> stack2 = new Stack<Integer>();
    Queue<Integer> queue = new LinkedList<Integer>();

    while (stack.size() > n) {
        queue.add(stack.pop());
    }
    while (!stack.isEmpty()) {
        int element = stack.pop();
        stack2.push(element);
        if (element % 2 == 0) {
            queue.add(element);
        }
    }
    while (!queue.isEmpty()) {
        stack.push(queue.remove());
    }
    while (!stack2.isEmpty()) {
        stack.push(stack2.pop());
    }

    println(stack);
}
```

{1, 2, 3, 4, 5, 6}, n=3
{67, 29, 115, 84, 33, 71, 90}, n=5

Test your work with the following website:
https://www.codestepbystep.com/problem/view/java/collections/stackqueue/collectionMystery10

Lab work (in school laboratory):

There are different algorithms to determine if a string is a palindrome. Our work in this lab uses Stack and Queue ADTs. Here is an approach:

- Read the input string one character at a time.

- Push a copy of a character on to the stack. Add the same character to (the back of the) queue.

- When the line has been entirely read, the program **repeatedly** compares the top of element of the stack and the front element of the queue; in case of a match, it pops the stack and (simultaneously) de-queues the queue.

- If ever there is a mismatch between the top element of the stack and the front element of the queue, the program declares that the input string is not a palindrome, and halts. The program also reports a position where a different text found.

- If the stack (or equally correctly, the queue) gets empty, the program declares that the given string is a palindrome, and stops.

Because of the high importance of a stack and a queue as data structures in computer science, Java includes their implementation in the java.util package (http://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html). The Stack is a class. Queue is an interface (we will discuss interface). Since Queue is an interface, you need to instantiate a concrete implementation of the interface in order to use it. You choose java.util.LinkedList Queue implementations in the Java Collections API.

Here is a `Stack` usage example:

```
import java.util.Stack;
Stack stack = new Stack();
------
stack.push("1");
stack.push("2");
Object obj2 = stack.pop(); //the string "2" is at the top of the stack.
```

Here is a `Queue` usage example:

```
import java.util.Queue;
import java.util.LinkedList;
------
Queue queueA = new LinkedList();
queueA.add("element 1");
queueA.add("element 2");
queueA.add("element 3");
```

Important member methods are as follows.

Stack		**Queue**	
Member function (s is a stack object.)	*Meaning*	*Member function (q is a queue object.)*	*Meaning*
s.size();	Returns the number of elements in this stack.	q.size();	Returns the number of elements in this queue.
s.push(item);	Pushes an item onto the top of this stack.	q.add(item);	Inserts the specified element into this queue.
Object obj = s.pop();	Removes the object at the top of this stack and returns that object as the value of this function.	Object obj = q.remove();	Retrieves and removes the head of this queue.
Object obj = s.peek();	Looks at the object at the top of this stack without removing it from the stack.	Object obj = q.peek();	Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
s.empty();	Tests if this stack is empty.	Use q.peek() to determine if queue is emptied. Check for null as returned value. Or use q.size().	

Sample Diaglog:

```
Please enter a string of characters: abba

The given string is a palindrome.

Want to examine another string? (y/n): y

Please enter a string of characters: 11223311

The given string is not a palindrome, since the symbol at position 3
from the left is different from the symbol at position 3 from the right.

Want to examine another string? (y/n):n

Bye!
```

Activities:

1. Copy instructor’s class (Palindrome.java) from Canvas into your working directory. Note that this is only a template only. You have to place your own code into it. Of course, you can develop your own class and method. However, the class name must be called Palindrome and the method to check for Palindrome is called checkPalindrome.

2. Develop your program according the pseudo code given in your lecture.

3. Test your program by running the main method.

4. Run a sample of your instructor’s unit test (PalindromeTest.java) to validate your work.

Deliverables:

(1) Demonstrate your result of the preparation work for the lab assignment.

(2) Turn in your modified Palindrome.java and your programs’ output file in PDF format to Canvas.