Assignment #1 – Stack and Queue

Abstract Data Structure Type like Stack and Queue are very useful. In this assignment, you will first implement stack and queue, then use Stack and Queue to solve some interesting problems.

Objectives:

Implement some abstract data structures and use the data structures in problem solving.

The main objectives of this assignment are:

- Implement Stack ADT using singly linked list
- Implement Queue ADT using circular array
- Use Stack and Queue to solve two software problems
  - Josephus Game Simulation
  - Postfix Expression Evaluation.
- Write Junit test cases

Reference

Circular Array: Section 6.2.2 Array-Bases Queue Implementation (page241-244)

Generic types: Generics.

The Assignment:

The summary of the tasks:

- Implement Stack ADT using singly linked list. Write Unit test cases.
- Implement Queue ADT using circular array. Write Unit test cases.

- Write code to simulate the Josephus game. You are required to use a queue as we discussed in class. Write test cases to test your code. If you are interested in the origin of this problem, see https://en.wikipedia.org/wiki/Josephus_problem.

- Write code to evaluate a given postfix expression. You are required to use a stack as we discussed in class. Write test cases to test your code.

Steps:

1. Download and import the following zip file: Assignment1_List.zip. For Eclipse:

1. Open Eclipse
2. Choose "File → Import"
3. Select the "General" category
4. Pick "Existing Projects into Workspace"
5. Click on "Next"
6. Select the button for "Select Archive File" and browse to/select the zip file.
7. Click on "Finish"

2. Finish the rest of the tasks following in this order. Please see the document of the method in interface file `Stack.java` and `Queue.java` under `net.datastructures`

1) Linked List Stack
   a. Create `LinkedListStackTest.java` in src folder and write test cases.
   b. Implement the methods in LinkedListStack.java.
2) Circular Array Queue
   a. Create `CircularArrayQueueTest.java` in src folder and write test cases
   b. Implement the methods in CircularArrayQueue.java.
3) Josephus Game
   a. Create `JosephusTest.java` in src folder and write test cases
   b. Implement the order() method in Josephus.java.
4) Postfix Expression Evaluation
   a. Create `PostfixExpressionTest.java` in src folder and write test cases
   b. Implement the evaluate() method in PostfixExpression.java.

Submission:

First, look at the following checklist:

1. Do you ever `import` from `java.util?` If remove it immediately.

   Unless the assignment specifically mentions it is permissible to import a specific package from java.util, you should never include any of java's native data structures.

2. Is the indentation easily readable? You can have Eclipse correct indentation by highlighting all code and select "Source → Correct Indentation".
3. Are comments well organized and concise?

If you have reviewed your code and all the criteria on the checklist are acceptable, follow the submission procedure, export to prog1.zip and upload to canvas.

Grading Criteria:

- The correct implementation of all methods in `LinkedListStack` : 25

- The correct implementation of all methods in `CircularArrayQueue` : 25
- The Josephus game function `order()` works correctly: 20
- The Postfix expression evaluation function `evaluate()` works correctly: 20
- Clear concise code with good commenting: 10
- Junit Test cases 20 points
    - `LinkedListStackTest.java`
    - `CircularArrayQueueTest.java`
    - `JosephusTest.java`
    - `PostFixExpression.java`

Appendix: Here are the files:

```java
package cs2321;

import net.datastructures.Stack;

public class LinkedListStack<E> implements Stack<E> {

	@Override
	public int size() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public boolean isEmpty() {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
	public void push(E e) {
		// TODO Auto-generated method stub

	}

	@Override
	public E top() {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public E pop() {
		// TODO Auto-generated method stub
		return null;
	}

}
```

```java
package cs2321;

public class Josephus{
	/**
	 * All persons sit in a circle. When we go around the circle, initially starting
	 * from the first person, then the second person, then the third...
	 * we count 1,2,3,.., k-1. The next person, that is the k-th person is out.
	 * Then we restart the counting from the next person, go around, the k-th person
	 * is out. Keep going the same way, when there is only one person left, she/he
	 * is the winner.
	 *
	 * @parameter persons  an array of string which contains all player names.
	 * @parameter k  an integer specifying the k-th person will be kicked out of the game
	 * @return return a array in the order when the players were out of the game.
	 *         The last one in the array is the winner.
	 */
	public String[] order(String[] persons, int k ) {
		//TODO: implement this method with the help of CircularArrayQueue
		return null;
	}
}
```

```java
/**
 *
 */
package cs2321;

import net.datastructures.Queue;

/**
 * @author ruihong-adm
 * @param <E>
 *
 */

public class CircularArrayQueue<E> implements Queue<E> {

	public CircularArrayQueue(int queueSize) {
		// TODO: create a new queue with the specified size
	}

	@Override
	public int size() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public boolean isEmpty() {
		// TODO Auto-generated method stub
		return false;
	}


	@Override
	public E first() {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public E dequeue() {
		// TODO Auto-generated method stub
		return null;
	}

	/* Throw the exception IllegalStateException when the queue is full */
	@Override
	public void enqueue(E e) throws IllegalStateException {
		// TODO Auto-generated method stub

	}


}
```

```java
package cs2321;

public class PostfixExpression {

	/**
	 * Evaluate a postfix expression.
	 * Postfix expression notation has operands first, following by the operations.
	 * For example:
	 *    13 5 *           is same as 13 * 5
	 *    4 20 5 + * 6 -   is same as 4 * (20 + 5) - 6
	 *
	 * In this homework, expression in the argument only contains
	 *     integer, +, -, *, / and a space between every number and operation.
	 * You may assume the result will be integer as well.
	 *
	 * @param exp The postfix expression
	 * @return the result of the expression
	 */
	public static int evaluate(String exp) {
		//TO DO: implement this function with the help of Stack

		/* IMPOTANT NOTE:
		 * Since the argument exp is a string, you need to parse the string expression
		 * first to get operands and operators first. Because we knew there is a space
		 * between the operands and operators,
		 * you can use the function string.split(" ") to return an array of tokens in
		 * exp.
		 */

		return 0;
	}


}
```