**Lab Assignment #4 – Using ADT Stacks, Queues, and Lists**

Due Date: By the dropbox deadline.

Purpose: The purpose of this Lab assignment is to:

1. Design algorithms that describe operations on ADT stacks, queues, and lists
2. Implement and test appropriate methods in Java

References: Read the course’s text chapter 6, 7 and the lecture slides. This material provides the necessary information that you need to complete the exercises.

**Instructions:**

You **MUST** create a short demo video of your solution. Do not show yourself in the video. Upload your video in your personal youtube account or google-drive account and share it with the instructor **only**. Do not share it publicly. During submission at the dropbox, **write the link of your video** in the **Comments** **box** (present near the bottom of the submission page). Next, create a zip file of your solution as mentioned below in section **Submission Rules**, upload that zip file, and submit.

You **must** name a relevant Eclipse project according to the following rule:

YourFirstname\_YourLastname\_COMP254\_Labnumber\_ExerciseNumber.

Example: If student name is John Smith, the name of Eclipse project for Ex1 of Lab1 should be **John\_Smith\_COMP254\_Lab1\_Ex1**

**Submission Rules:**

Compress all your Eclipse projects as a **single** **zip** filethat is named according to the following rule: YourFirstname\_YourLastname\_COMP254\_Labnumber.zip

Example: **John\_Smith\_COMP254\_Lab1.zip**

Submit the above single zip file using the procedure mentioned in section **Instructions** above.

**Evaluation:**

|  |  |
| --- | --- |
| **Correct implementation of requirements:**   * Correct ADT data structure algorithm * Correct Java implementation * Explanation of algorithm when asked | 90% |
| **Friendly I/O** | 10% |
| **Total** | 100% |

**Exercise 1**

Implement a method **transfer** in class **LinkedStack**. This method should transfer all elements of a stack *sourceS* to another stack *targetS* so that the element that starts at the top of *sourceS* is the first one to be inserted in *targetS*, and the element at the bottom of *sourceS* ends up at the top of *targetS*. The operation should result in *sourceS* being an empty stack. Test this method in the **main** method of **LinkedStack**. The class **LinkedStack** is in **Lesson6Examples** posted in the **eCentennial** module “**Lesson Examples (from textbook)**”.

(5 marks)

**Exercise 2**

Implement a method **concatenate** in class **LinkedQueue**. This method should take all the elements of a queue *sourceQ* and append them to the end of another queue *targetQ*. The operation should result in *sourceQ* being an empty queue. Test this method in the **main** method of **LinkedQueue**. The class **LinkedQueue** is in **Lesson6Examples** posted in the **eCentennial** module “**Lesson Examples (from textbook)**”.

(5 marks)