CSCI 230 PROGRAM 3

Due
The due date is specified in the OAKS dropbox.

OBJECTIVES
- To implement the Stack ADT with an array and with a linked list.
- To implement a Java interface that defines the Stack type.
- To instrument the code to collect runtime data.
- To empirically analyze the two implementations.

PROGRAM REQUIREMENTS
1) Create and test two distinct implementations of the stack data structure in Java that implements **StackInterface**. Use a 1D array for one implementation of stack and a singly linked list for the other. You can use your generic list data structure as is. However, arrays in Java do not support generics, so don't bother trying. Just hard code an Integer type into StackA for now. In the future we can use ArrayList, which supports generics.

For the push operation in the array implementation, implement an array-doubling scheme when the next push would exceed the current array length. And when the size of the stack falls below ¼ of the current array length, have the array to free up memory. The default size of the array is 1,000 elements.

Class name for the array-based stack: **StackA**.java
Class name for the list-based stack: **StackL**.java

2) After both stack implementations are working, then create a test program to gather timing data on both implementations. For each of the stack implementations, do the following:

Create a stack instance.
Start the timer.
Push the integer values from 1 to N into the stack, one at a time using a for loop.
Pop the values from the stack one at a time using a for loop.
End the timer.
Output the runtime to the console "To push and pop <N> values took <milliseconds> milliseconds>

Class name for the class that analysis application: **TestStack**.java

REFERENCES
None

PROGRAM SPECIFICATION
Application:
Exception handling does not have to be used.

Abstract Data Type:
Use a Stack ADT.
Implement a stack as a data structure using a 1D array that holds Integer objects. Also implement a stack with a singly linked list. You should already have a generic singly linked list working. Put a main method in each of the stack classes that tests the stack implementations to make sure they work properly.

Analysis
Run the **TestStack** class to test the stack implementations multiple times for different values of N. Each time, record the time it takes to process those N items (push them all and pop them all). Create a visualization (an Excel graph) showing the data collected and an estimate for the T(N) function that fits the empirical data for each stack implementation. Save the graph as a pdf file called EmpiricalAnalysis.pdf

GUI:
No GUI. Use the console for I/O.

PROGRAM DOCUMENTATION
Provide internal documentation only as required in the program documentation standard in OAKS.
Updates and clarifications to this assignment, if needed, will be done on Discussions.

PROGRAM SUBMISSION
Use the corresponding dropbox in OAKS using the same naming conventions as given in Program 1.