Assignment Prefix: Lab106

Due Date: Tuesday, Oct. 9th @ 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 Lab106 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 the two Abstract Data Types (ADTs) described below. Each ADT should include a generic interface and an efficient generic static implementation.

DoubleStack ADT:

- Design and implement an ADT for a two-color, double stack ADT that consists of two stacks – one "red" and one "blue" – and has as its operations color-coded versions of the regular Stack ADT operations.
- For example, this ADT should support both a redPush and a bluePush operation.
- Write the generic interface for this DoubleStack ADT. Include a toString and equals method in the interface.
- Give an **efficient generic static implementation** of this ADT **using a single array** as the container whose capacity is set at some value N that is assumed to always be larger than the sizes of the red and blue stacks combined.
    - Example. The array has a length of ten.
    - At any given time the sum of the elements in both stacks cannot exceed the length of the array.
    - If at some point the red stack has 8 elements then, at that time, the blue stack could not have more than 2 elements.
    - At another time the red stack could has 3 elements then, at that time, the blue stack could not have more than 7 elements.

  - So, at any given time the maximum size (capacity) of each stack will be determined by both the length of the array and by how many elements are in the other stack.
- Provide a test of your DoubleStack that clearly shows that all of the methods work correctly.
- Hints:
  - You should be able to implement all of the methods (with the exception of toString and equals) with a BigO of 1 for both the red and the blue stacks.
  - An ArrayList does not have a BigO of 1 for all its operations.

LeakyStack ADT:

- The introduction of Section 6.1 notes that stacks are often used to provide "undo" support in applications like a Web browser or text editor. While support for undo can be implemented with an unbounded stack, many applications provide only limited support for such an undo history, with a fixed stack capacity.
- When a push is invoked on a LeakyStack at full capacity, rather than throwing an exception, accept the pushed element at the top while "leaking" the oldest element from the bottom of the stack to make room.
- Write the generic interface for this LeakyStack ADT. Include a toString and equals method in the interface.
- Give an **efficient <u>static</u> implementation** of the LeakyStack abstraction.
- Provide a test of your LeakyStack that clearly shows that all of the methods work correctly.
- Hints:
  - You should be able to implement all of the methods (with the exception of toString and equals) with a BigO of 1 for the LeakyStack.
  - An ArrayList does not have a BigO of 1 for all its operations.

Things to turn in:

- Open a Microsoft Word document named Lab106.docx
- Copy and Paste the source code of the Client (make sure to use *Ctrl + A* to select all the source code of the program and *Ctrl + C* to copy).
- Copy and Paste your ADT interfaces and implementations.

- You **DO NOT** need to include copies of any interfaces or classes that you copied out of the textbook without modification.
- Y**ou DO need to include the classes that you modified from the textbook.**
- Copy and paste the output of the client program
- Next, zip the Project folder.
- Submit both your Word document and project zipped file.