CSCI 1302: Software Development

Fall Semester, 2016

Project 3

Stack GUI

Instructor: *Eman Saleh*

Due Date: 10/31/2016 @ 11:30 PM

In this project you will implement your own **Stack** class. You will also design and implement a GUI to emulate you stack and its operations. You will practice unit testing by defining test cases in *JUnit* for the constructors and methods of your Stack class. Finally, you will produce *Javadoc* documentation for your Stack class.

Project Requirements:

1. **[18 Points]** Define a generic class definition for the class Stack, your class must implement the interface **StackADT** defined at page 3 of this document.

   Instance variables of class **Stack** are: -

   stack: An array that represents the stack.

   top: An index that keeps track of the top stack element. Where stack[top] is the top element of the stack.

   DEFAULT_CAPACITY: A constant integer that sets the maximum default array size.

   You must define two constructors:

   - A default constructor that creates an empty stack of size equals to DEFAULT_CAPACITY.
   - A constructor that creates an empty stack using a specified capacity as a parameter.

2. **[22 Points]** Using **JUNIT 4.12,** Define a test class called **MyStackTester**. MyStackTester must contain at least 11 test cases for all methods defined in class **Stack.** Test your generic class for one data type. You have 9 methods; you must define a test case for each method, except for pop () and push () methods where you have to define 2 test cases for each; Test the pop () method after defining a stack and pushing three values, assert that last pushed value was returned from the pop () method. Another test case is needed to test popping from an empty stack to ensure that the correct exception is thrown. See requirement **3** for exceptions.

3. **[10 Points]** Define two exceptions:

   3.1 EmptyStackException: This exception is thrown by the pop () and peek () methods when called on an empty stack.

   3.2 FullStackException: This exception is thrown by the push () method when called on a full stack.

4. **[10 Points]** Generate class documentations (the HTML files) generated by **Javadoc** and save them in a directory named stackDoc.

5. **[30 Points]** Define class **IntegerStackEmulator**, that generates a GUI similar to a calculator, the GUI interface displays a frame that contains a text field (to input the size of the stack or the value to be pushed in the stack), set of buttons labeled with stack operations (methods) and other set of buttons labeled with integers from 0 to 9 and A picture (proper display of the stack) that changes prior to every stack operation. The user should be able to enter the value to be pushed in the stack either by typing into the text field or by pressing the numbered buttons. In either cases the input appears in the text field and the stack is redisplayed (possibly repainted!) see figure 2 and 3.

6. **[5 Points]** practice good programming style, use proper meaningful identifier names, indentation, comments. ***Remember***, test cases must be written before writing the code.

7. **[5 Points]** Submit a README.txt file telling us how to compile and execute your program and how to generate Javadoc.

   MyclassTest.java must be defined in a package called uga.1302.test. All other classes must be defined in a package called uga.1302.myStack.

What to Submit:

Submit your project3 directory to user cs1302a on *nike* at a directory called Project3.
(use the following command: submit Project3 cs1302a).

Your directory must contain the following files/directories:

1. The Interface file StackADT.java
2. Stack.java
3. FullStackException.java
4. EmptyStackException.java
5. IntegerStackEmulator.java
6. MyStackTester.java
7. README.TXT
8. The directory StackDoc that contains HTML Javadoc documentation files.

Basic grading criteria:

1. If the project did not compile on *nike* 0%
2. If the project compiled but did not run 0% - 30%
   30% is given if all required files were submitted and program code completely satisfies all functional requirements.
3. If the project run with wrong output 0% - 60%
   Depending on the solution, 50-60% is given ONLY if the cause of the error was minor after checking all functional aspects.

Bonus 5 points will be assigned to projects with attractive GUI. Be creative! Use your imagination!

See course syllabus for late submission evaluation

Listing 1: Interface StackADT

```
package uga.1302.myStack;
/**
 * Defines the interface to a stack collection.
 *
 * @author [YOUR NAME]
Figure 1Tha StackADT interface in UML
 */
public interface StackADT<T>
{
   /**
    * Adds the specified element to the top of this stack.
    * @param element element to be pushed onto the stack
    */
   public void push(T element);

   /**
    * Removes and returns the top element from this stack.
    * @return the element removed from the stack
    */
   public T pop();
   /**
    * Returns without removing the top element of this stack.
    * @return the element on top of the stack
    */
   public T peek();

   /**
    * Returns true if this stack contains no elements.
    * @return true if the stack is empty
    */
   public boolean isEmpty();
   /**
    * Returns true if this stack is full; number of elements= stak.length.
    * @return true if the stack is full
    */
   public boolean isFull();

   /**
    * Returns the number of elements in this stack.
    * @return the number of elements in the stack
    */

   public int size();
   /**
    * Returns a string representation of this stack.
    * @return a string representation of the stack
    */
   public String toString();
}
```

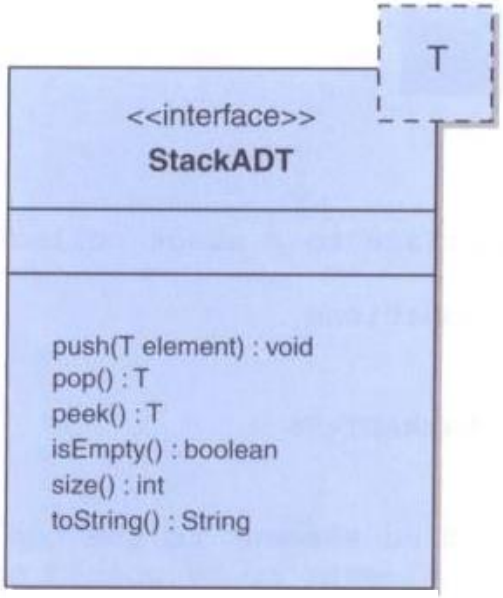

Figure 1 The StackADT in UML

Simplest accepted GUI for partial credits