Spring 2014 - CS 4/510 - Topics in Software Testing

Due: May 9 2014

Project 2: JUnit in Eclipse IDE

All of you must be familiar with Stacks by now. The Java Core API also has a stack class in the package java.util. Your job is to implement the operations of Stack in Java using different data structures like arrays, linked lists from scratch and test using JUnit. In your test cases, you can compare it against the Java API to check that it behaves the same as your own class. The goal of this project is to get familiar with different annotations and assertions used in JUnit. Also, we would analyze the code coverage in each case using a code coverage analyzer - EclEmma

You can download the Eclipse IDE (with Java and Junit) from
http://www.eclipse.org/downloads/

You can install EclEmma for analyzing code coverage by following guidelines here
http://www.eclemma.org/installation.html

Details:

A Stack is a LIFO sequence. Addition and removal takes place only at one end, called the top.

Operations:

1. Push(x) :
   Add an item on the top
2. Pop()
   Remove the item at the top and returns it
   Exception if the stack is empty
3. Peek()
   Return the items at the top (without removing it)
   Exception if the stack is empty
4. Size()
   Return the number of items currently in the stack
5. isEmpty()
   Return whether the stack has no items

Implementations:

1. Wrapper
   A stack class implemented as a wrapper around a java.util.LinkedList
   All stack methods can simply be delegated to LinkedList methods

2. An Array Implementation (Bounded)
   Capacity is set at creation time
   Push happens only if stack isn't already full else state exception

3. Implementation of the stack interface using singly linked nodes
   Contains Node class with object data and Node next (which points to next node in the sequence)

4. Implement stack as a combination of two queues. You can use the standard queue, dequeue etc. functions from the Java API.

Unit Testing:

You must make sure that these conditions are tested:

1. On construction: stack is empty
2. On construction: size is 0
3. After k pushes (k > 0) after construction, the stack is not empty and its size is k
4. If one pushes p then peeks, the value returned is p, but the size stays the same
5. If one pushes p then pops, the value popped is p.
6. If the size is k, then after k pops, the stack is empty and has a size 0
7. Popping from an empty stack
8. Peeking into an empty stack
9. For bounded stacks, pushing onto a full stack
10. For atleast 2 test cases, compare the result with Java API for stack rather than providing your own concrete value.

Note that even though we have four classes to test, we can reuse the test cases by putting the common tests in Base Test class. The concrete test classes instantiate the specific kind of stack. Because they are subclasses of the Base Test class, they inherit all of the common test cases.

Grading:

Implementing operations for wrapper	10 points (2 points each)
Implementing operations for Bounded Array	10 points (2 points each)
Implementing as singly linked nodes	10 points (2 points each)
Implementing as combination of two queues	10 points (2 points each)
Implementing all Test cases	20 points (2 points each)
Using a common Base Test Class	10 points
Demo a failure	10 points
Generate coverage report	10 points
Demo how to view coverage	10 points