**CS 410 Software Engineering – Spring 2018**

**Assignment 3**

**Junit Test**

**Part I**: You are given the following LinkedStack.java file which includes the implementation of Stack using LinkedList data structure.

import java.util.NoSuchElementException;

public class LinkedStack<Item>{

    private int n;          // size of the stack

    private Node first;     // top of stack

    private class Node {

        private Item item;

        private Node next;

    }

    //initializes an empty stack.

    public LinkedStack() {

        first = null;

        n = 0;

    }

    //returns true if this stack is empty; false otherwise

    public boolean isEmpty() {

        return first == null;

    }

    //returns the number of items in the stack.

    public int size() {

        return n;

    }

    //inserts an item to the stack

    public void push(Item item) {

        Node temp = first;

        first = new Node();

        first.item = item;

        first.next = temp;

        n++;

    }

    //removes and returns the top element in the stack

    public Item pop()  {

        if (isEmpty()) throw new ArrayIndexOutOfBoundsException("Stack underflow");

        Item item = first.item;

        first = first.next;

        return item;

    }

    //returns (but does not remove) the item most recently added to this stack.

    public Item peek() {

        if (isEmpty()) throw new ArrayIndexOutOfBoundsException("Stack underflow");

        return first.item;

    }

}

You are supposed to use **Junit5 to** create a test class with test cases, LinkedStackTest.java, to test the operations of LinkedStack. Here are some other requirements for the activity:

* Create a LinkedStackTest.java and write test cases to test the operations of LinkedStack.java

I did it

* Pushing 10000 objects into a stack should not take longer than 5 miliseconds I did it
* Pop and peek methods should throw NoSuchElementException

if stack is empty

I did it

* Provide the list of failing test cases and explain the reason for failure 1. **The pop method fails** my test, because the size n doesn’t decrement when I’m popping with the implementation you gave us. 2. **The peek and pop when they are empty** throw the exception **ArrayIndexOutOfBoundsException** instead of **NoSuchElementException.**
* Resolve the issues by modifying the original class in a separate class LinkedStackFixed.java file and make sure all tests pass.

I did it

**Part 2:**

Prepare a Software Test Document for LinkedStack class test cases. You are given a sample for a template as follows:

1. TEST CASE

**1.1 Test Case-1**

**1.1.1 Purpose**

The empty test checks the empty() method to make sure that it doesn’t have any item.

**1.1.2 Input**

I did not input anything in the beginning , then I push a object of type Object to the linkedStack to test it again.

**1.1.3 Expected Output**

First, assertTrue(list.isEmpty()) will return True , after I push my object assertFalse(list.isEmpty()) will also return true.

**1.1.4 Pass/Fail Criteria**

Pass : assertTrue (list.isEmpty()) return true before pushing anything, after pushing an element assertFalse(list.isEMpty) will return true.

Fail : assertTrue(list.isEmpty()) return false before pushing anything or after pushing an element assertFalse(list.isEmpty()) will return False.

**1.2 Test Case-2**

**1.2.1 Purpose**

The size test checks the size method to make sure that it is correct, so that we don’t have future problems with implementation.

**1.2.2 Input**

I puched 3 object of type Object inside my linkedStack

**1.2.3 Expected Output**

Return true for assertEquals(3, list.size) since the size is 3

**1.2.4 Pass/Fail Criteria**

pass: the number of object “n” in the linkedStack will return true for assertEquals(n, list.size())

Fail: the number of object “n” in the linkedStack will return false for assertEquals(“any number different from n”, list.size())

**1.3 Test Case-3**

**1.3.1 Purpose**

The push tests if the size method is the same to the number of element we pushed in our linkedStack.

**1.3.2 Input**

3 objects of type Object

**1.3.3 Expected Output**

Return true for assertEquals(3, list.size) since the size is 3

**1.3.4 Pass/Fail Criteria**

pass: the number of object “n” in the linkedStack will return true for assertEquals(n, list.size)

Fail: the number of object “n” in the linkedStack will return false for assertEquals(“any number different from n”, list.size)

**1.4 Test Case-4**

**1.4.1 Purpose**

The purpose of the pop test is to remove an element from our linkedStack, so the size will decrease

**1.4.2 Input**

I pushed two objects of type Object , then pop

**1.4.3 Expected Output**

Return true for assertEquals(1, list.size) since the size should become 1

**1.4.4 Pass/Fail Criteria**

pass: Return true for a number of size “n” for assertEquals( n-1, list.size)every time we pop

Fail: Return false for a number of size “n” for assertEquals(diffenrent to n-1, list.size)every time we pop

**1.5 Test Case-5**

**1.5.1 Purpose**

Peek test checks if the the right element is on the top of our linkedStack.

**1.5.2 Input**

I pushed 5,1,2

**1.5.3 Expected Output**

The output is 2

**1.5.4 Pass/Fail Criteria**

pass: return true if the last entered item “N” inside the linkedStack for asserrEquals(N, list.peek())

Fail: return false if the last entered item “N” inside the linkedStack for asserrEquals(any item different to N, list.peek())

**1.6 Test Case-6**

**1.6.1 Purpose**

The peekExeption test throws when peek() method is empty

**1.6.2 Input**

Empty linkedStack

**1.6.3 Expected Output**

The method will throw NoSuchElementException

**1.6.4 Pass/Fail Criteria**

Pass: peep() method when empty should throw NoSuchElementException

Fail: peep() method when empty should throw no exception or any other exception different to NoSuchElementException

**1.7 Test Case-7**

**1.7.1 Purpose**

The popkExeption test throws when pop() method is empty

**1.7.2 Input**

Empty linkedStack

**1.7.3 Expected Output**

The method will throw NoSuchElementException

**1.7.4 Pass/Fail Criteria**

Pass: pop() method when empty should throw NoSuchElementException

Fail: pop() method when empty should throw no exception or any other exception different to NoSuchElementException

**1.8 Test Case-8**

**1.8.1 Purpose**

PushingObject Test make sure that it takes less that 5 miliseconds when pushing 10000 items.

**1.8.2 Input**

I pushed 10000 object of type Object

**1.8.3 Expected Output**

The execution time should takes less than 5miliseconds

**1.8.4 Pass/Fail Criteria**

Pass: execution time<5ms

Fail: execution time>5ms