CS 1632 – The Final Deliverable

Writing a new app from scratch using TDD

Charlotte Chen and Daniel Hui

https://github.com/dth26/CS1632_FINAL_PROJECT

Why we chose TDD

We decided to go for option number 9: *Writing a new app from scratch using TDD* for our final deliverable, because we believe TDD is an effective approach to writing software in the professional industry nowadays. Writing tests before coding means that we could potentially spend less time debugging. Also, TDD would provide us with an entire test suite for our program to be used if the code is updated; a very efficient and organized way to write code. For this deliverable, we built a program that simulates a conga line (*https://en.wikipedia.org/wiki/Conga_line)* that implements a custom LinkedList class we also built. We thought the conga line program would be an effective and fun way to visualize what we can do with the linkedlist data structure we built.

Process

We started by building a Node class. Using this Node class we were able to build a LinkedList class that implemented the following functionalites: getLast(), size(), indexOf(), remove(), get(), add(Node), add(T value), addLast(), clear(), indexOf(), and addAtIndex(). Next, we built a real-life application implementing our LinkedList class: A Conga Line simulator program

The Conga Line simulator program had the following functionalities: person enters line, person enters behind a specific person, person leaves the line, everyone leaves the line, and show whose hips a person is holding.

Testing Aspect

Before we wrote any new function for a class, we wrote test codes using JUnit that would test the function for errors or inconsistencies. There are two parts to our tests; the first part tests the methods in the LinkedList class and the second part tests the methods in the CongaLine class. Using this system, we would develop software that is properly tested, and help us further understand the software we were writing.

Problems/Difficulties

Our LinkedList class throws an IndexOutOfBoundsException if an index passed to get() is not between 0 and the size of the LinkedList. So to test whether this situation is handled properly, JUnit would need to check if an exception is thrown. Luckily, JUnit provides a functionality to test for an exception: *@Test (expected = Exception.class)*

It was sometimes difficult in envisioning every possible scenario/errors that we would have to test without having written any actual code for the methods. We seemed to neglect some error handling cases, edge cases, and corner cases within the methods that we would need to test for. For instance, we wrote the test methods for the remove method in our LinkedList class. However, while writing the code for the remove method we realized that removing the first element in the list would be handled differently than removing an element in the middle of the list. So using TDD, we sometimes needed to add more test code after realizing our code was not fully covered.

Assessment of TDD

Overall, we both enjoyed using TDD as a way of writing software. We really felt that after writing test code we developed a more comprehensive and clearer sense of how we should write each method. We were a lot more confident and more aware of what we had to do, thus saving us time while actually writing the code. Just as importantly, if we wanted to update our software, we would already have a test suite for the entire system. Hopefully, we would have the opportunity to write code using this system in our careers.

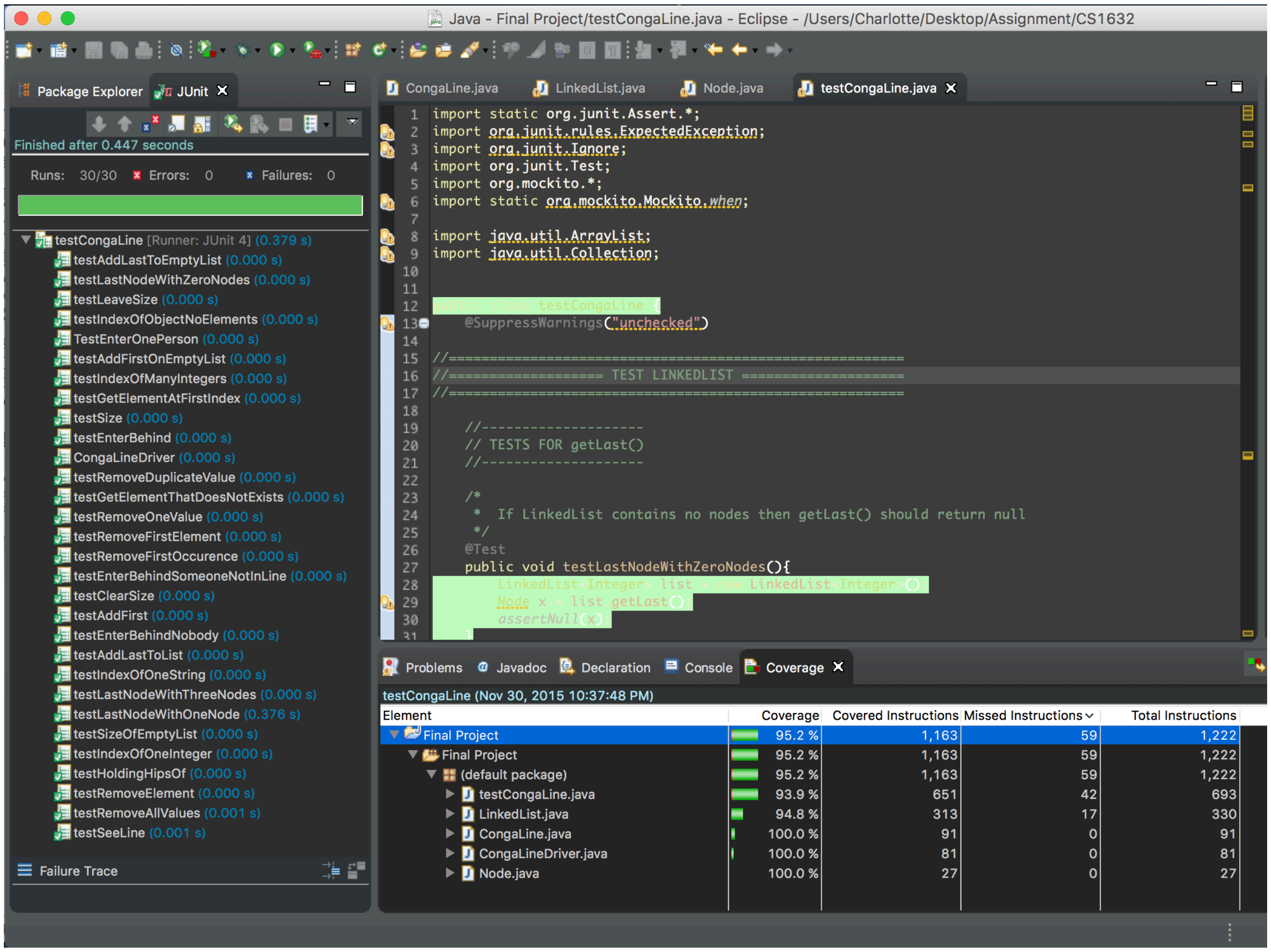
Ready for Release

The Conga Line Simulator implemented via the Linked List data structure was written using Test-Driven Development. This means that all code in our Linked List and Conga Line programs have been tested for any possible defects (see screenshots below for our test result and our code coverage). The Conga Line program and Linked List program both meet the basic functionalities without error. Currently, after fixing all defects we found that both programs run smoothly. After rigorous testing and refactoring code, our software testing team has decided that this software is ready for release immediately.

The code and test code for our program can be accessed at:
https://github.com/dth26/CS1632_FINAL_PROJECT

Screenshots

Code Coverage (95.2%)



Java - Final Project/testCongaLine.java - Eclipse - /Users/Charlotte/Desktop/Assignment/CS1632

Finished after 0.447 seconds

Runs: 30/30 Errors: 0 Failures: 0

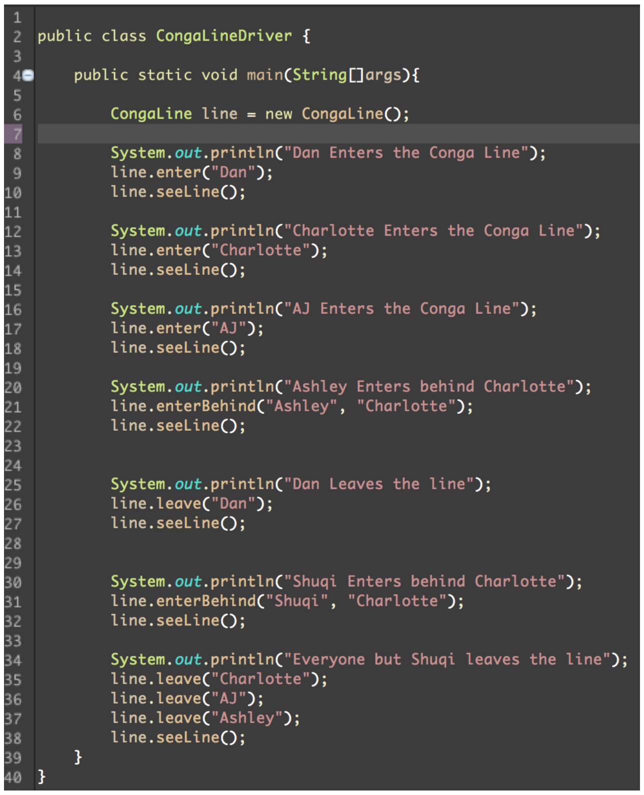
```
testCongaLine [Runner: JUnit 4] (0.379 s)
    testAddLastToEmptyList (0.000 s)
    testLastNodeWithZeroNodes (0.000 s)
    testLeaveSize (0.000 s)
    testIndexOfObjectNoElements (0.000 s)
    TestEnterOnePerson (0.000 s)
    testAddFirstOnEmptyList (0.000 s)
    testIndexOfManyIntegers (0.000 s)
    testGetElementAtFirstIndex (0.000 s)
    testSize (0.000 s)
    testEnterBehind (0.000 s)
    CongaLineDriver (0.000 s)
    testRemoveDuplicateValue (0.000 s)
    testGetElementThatDoesNotExists (0.000 s)
    testRemoveOneValue (0.000 s)
    testRemoveFirstElement (0.000 s)
    testRemoveFirstOccurence (0.000 s)
    testEnterBehindSomeoneNotInLine (0.000 s)
    testClearSize (0.000 s)
    testAddFirst (0.000 s)
    testEnterBehindNobody (0.000 s)
    testAddLastToList (0.000 s)
    testIndexOfOneString (0.000 s)
    testLastNodeWithThreeNodes (0.000 s)
    testLastNodeWithOneNode (0.376 s)
    testSizeOfEmptyList (0.000 s)
    testIndexOfOneInteger (0.000 s)
    testHoldingHipsOf (0.000 s)
    testRemoveElement (0.000 s)
    testRemoveAllValues (0.001 s)
    testSeeLine (0.001 s)
```

```java
import static org.junit.Assert.*;
import org.junit.rules.ExpectedException;
import org.junit.Ignore;
import org.junit.Test;
import org.mockito.*;
import static org.mockito.Mockito.when;

import java.util.ArrayList;
import java.util.Collection;


public class testCongaLine {
    @SuppressWarnings("unchecked")

//====================================================
//=================== TEST LINKEDLIST ====================
//====================================================

    //--------------------
    // TESTS FOR getLast()
    //--------------------

    /*
     *  If LinkedList contains no nodes then getLast() should return null
     */
    @Test
    public void testLastNodeWithZeroNodes(){
        LinkedList<Integer> list = new LinkedList<Integer>();
        Node x = list.getLast();
        assertNull(x);
    }
```

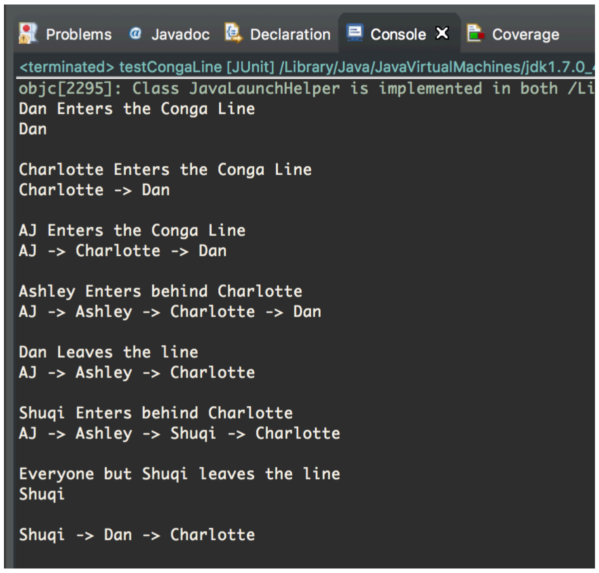
testCongaLine (Nov 30, 2015 10:37:48 PM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Final Project	95.2 %	1,163	59	1,222
Final Project	95.2 %	1,163	59	1,222
(default package)	95.2 %	1,163	59	1,222
testCongaLine.java	93.9 %	651	42	693
LinkedList.java	94.8 %	313	17	330
CongaLine.java	100.0 %	91	0	91
CongaLineDriver.java	100.0 %	81	0	81
Node.java	100.0 %	27	0	27

Driver code & execution result:



```java

public class CongaLineDriver {

    public static void main(String[]args){

        CongaLine line = new CongaLine();

        System.out.println("Dan Enters the Conga Line");
        line.enter("Dan");
        line.seeLine();

        System.out.println("Charlotte Enters the Conga Line");
        line.enter("Charlotte");
        line.seeLine();

        System.out.println("AJ Enters the Conga Line");
        line.enter("AJ");
        line.seeLine();

        System.out.println("Ashley Enters behind Charlotte");
        line.enterBehind("Ashley", "Charlotte");
        line.seeLine();


        System.out.println("Dan Leaves the line");
        line.leave("Dan");
        line.seeLine();


        System.out.println("Shuqi Enters behind Charlotte");
        line.enterBehind("Shuqi", "Charlotte");
        line.seeLine();

        System.out.println("Everyone but Shuqi leaves the line");
        line.leave("Charlotte");
        line.leave("AJ");
        line.leave("Ashley");
        line.seeLine();
    }
}
```



```
<terminated> testCongaLine [JUnit] /Library/Java/JavaVirtualMachines/jdk1.7.0_
objc[2295]: Class JavaLaunchHelper is implemented in both /Li
Dan Enters the Conga Line
Dan

Charlotte Enters the Conga Line
Charlotte -> Dan

AJ Enters the Conga Line
AJ -> Charlotte -> Dan

Ashley Enters behind Charlotte
AJ -> Ashley -> Charlotte -> Dan

Dan Leaves the line
AJ -> Ashley -> Charlotte

Shuqi Enters behind Charlotte
AJ -> Ashley -> Shuqi -> Charlotte

Everyone but Shuqi leaves the line
Shuqi

Shuqi -> Dan -> Charlotte
```