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.

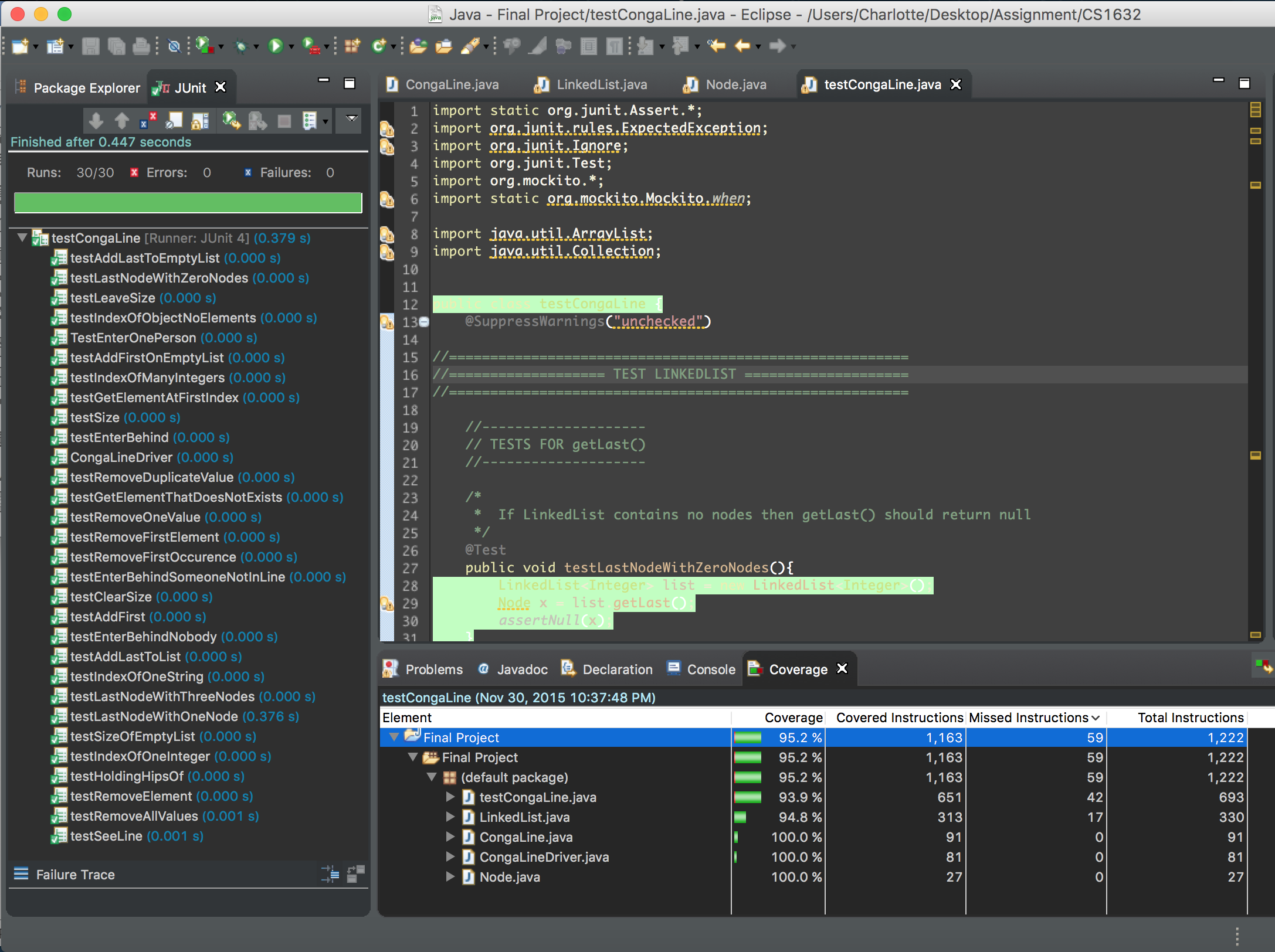
**Ready for Release**

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%)

****

Driver code & execution result:

