CS211 Report

Christian Holland- Student Number: 130044082

# A Brief Overview

For this project, I struggled at first to understand the inside code for how the Queue class actually worked. After I had established how it functions, I found creating the SingleElimination methods reasonably easy. I understood fairly soon just how the examiner had wanted us to design the program once all of the classes had been given to me, and it was just a matter of designing the IManager classes around this.

I have attempted the BubbleElimination class. I found it particularly difficult to allow the program to recognise when a team has previously won a match or not, in order to advance them to the overall winners array. I feel this was my only limitation; the class works except for when it sometimes cannot recognise that a player has won the previous game. So far, my SingleElimination and DoubleElimination classes are fully functional. The hardest part out of these two was establishing how the program was going to remember which queue the winning team came out of at the end within the DoubleElimination class, and also defining the order to which bracket would play the other bracket- for example whether the winners would play the losers when there were equal amount of teams in each bracket. I overcame this by stating that if the winners queue is bigger than the losers queue then player two from the winners queue; else play two from the losers queue. Finally, in order to establish an end to the brackets, I stated that if the losers queue and the winners’ queue both had only one team in it, then to set the queue to complete.

I have found the Junit tests to be more challenging than they should have been. It wasn’t actually coding them, it was deciding what things should be tested; the necessary testing a few of the methods needed were achieved by simply running the program- and I found it a little difficult to see where the examiner was going to be marking my tests- e.g. is there such thing as an ‘incorrect’ test? If the test runs smoothly and has an output, I assumed it cannot be ‘wrong’.

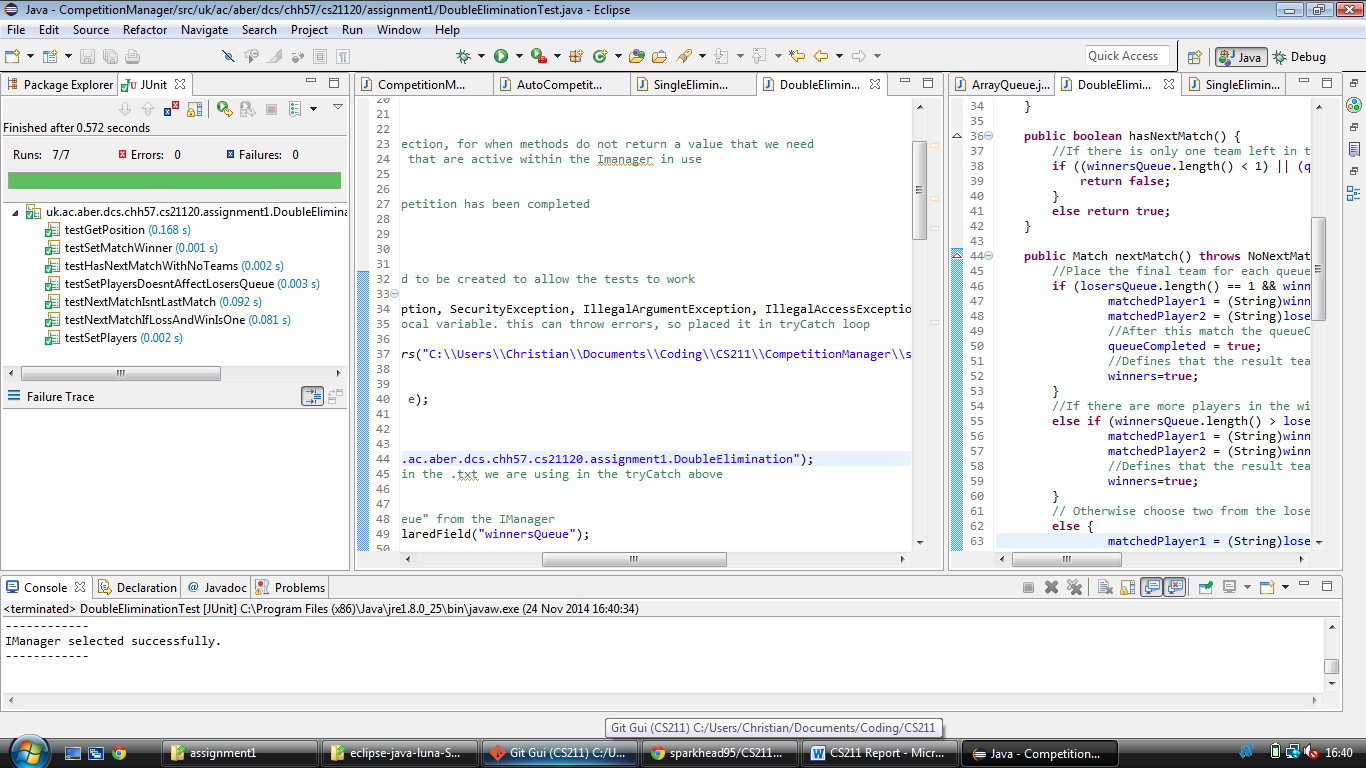
Within the Junit tests, I found that using reflection was the best way of retrieving variables from the IManager class from methods that do not return a value. However this proved to be quite tedious as every time my test changes a value and I want to check that value again, I had to repeat some code. It’s not difficult, however it is rather tedious. I found the manipulation of queues slightly difficult through the test class, and even once I understood it and found it easy, it was another thing that was quite tedious to enable full functionality.

Through this project I have furthered my knowledge in using interfaces, and how to use one centre class that can have different outputs depending on which class is used (that fits with the interface). I now also understand queues much more, having been that we had to create our own queue class. Also, surprisingly, I find reading in external files much easier using the arguments method (as previously I would have made an internal variable with the file path). This method is much better on command line however is not very handy through any other method of executable

# Testing Output

## SingleElimination JUnit Test

## DoubleElimination JUnit Test



## BubbleElimination JUnit Test

