Testing Documents

By: Zaeem Israr, Kevin banh, Amir Khademi

Introduction:

This document describes the test case of the Authoring app. A Junit class was used for testing purposes. For each test case there is a description as well as an explanation of the test case. The code that is tested in this app is the file generation and nothing else, reason being is that with GUI we would run in trouble with heavyweight/time-consuming unit tests to create for GUI components with stop and wait for user input in a build environment which would not be good. Therefore we have decided to test the logic of the GUI which would be the file generation.

File generation:

testFileGeneration()

The point of this test case is too test whether not the method does as it supposed to which in this case must put the node number in start. The same with the cell number and button number. In this case we have 3 parameters that are being called by 3 functions, the first being the file generation calling node1, upon doing so we the file generation must be put in start. The other 3 parameters are the cell number and button number. After the calling the function it gets compared to an expected value and it must be the same for it to pass.

addnode()

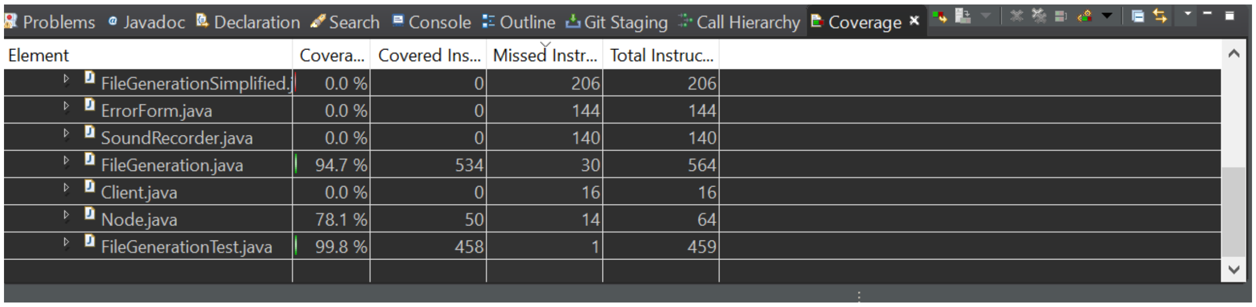
Testing was done by adding a child to the parent, and the outcome of the program has to be that the parent has the content of the child and if they are the same. Add node receives a list a list of parents and adds those to a list of its starts. This is why after calling addNode, we expect the same parents in the list of node 1.

testgenerateFile()

In this case we are testing the generate file constructor, and the way we tested this case is by creating our own scenarios as the user would in the GUI and we have a array of expected results. The outcome gets compared to the expected value and if they are the same the test will pass.

Test case being Sufficient:

The testing that was done, was sufficient according to the testing coverage we did of the case. According to the Professor anything above a 75% is good and as you can see in the picture below we have achieved a 94.7% coverage in file generation and 78.1 in Node.



Element	Covera...	Covered Ins...	Missed Instr...	Total Instruc...
FileGenerationSimplified.j	0.0 %	0	206	206
ErrorForm.java	0.0 %	0	144	144
SoundRecorder.java	0.0 %	0	140	140
FileGeneration.java	94.7 %	534	30	564
Client.java	0.0 %	0	16	16
Node.java	78.1 %	50	14	64
FileGenerationTest.java	99.8 %	458	1	459