Amer Alshoghri
Arsh Punia
Amaan Vania

EECS2311
2/23/2018
Group 10

Testing

Test Case 1:
testAddSkipButton ()

Derivation: the testAddSkipButton () methods were needed in order to ensure that the addSkipButton() method was writing the correct scenario file syntax to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a skip button line has been added with a specific pre-determined button number. These tests are sufficient to confirm the functionality of the method. It is also tested for incorrect string spacing and unnecessary characters.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 2:
testAddUserInput ()

Derivation: The testAddUserInput () methods were created to ensure that the addUserInput() method was writing the correct scenario file syntax to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a user input line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. The methods also

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 3:
testactivateKeys ()

Derivation: The testactivateKeys () methods were created to ensure that the activateKeys() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a */~skip-button:* … line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. The test case covers the key ONEE; other cases are covered in different testing methods.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 4:
testaddDispCellChar ()

Derivation: The testaddDispCellChar2 () methods were created to ensure that the addDispCellChar() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*disp-cell-char:…* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. These test cases cover the display of the character "a" on the first cell. Additionally, there is incorrect spacing between the cell number that is displaying the character and the character itself. The tests should also confirm that the line is not passing the test case.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 5:
testaddDispCellLower ()

Derivation: The testaddDispCellLower2 () methods were created to ensure that the addDispCellLower () method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*disp-cell-lower:…* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. These tests confirm that the invocation of the method to lower the 2nd pin of the 1st cell. Additionally, it tests the spacing error in the prepared test. This method confirms that the spacing errors will fail the test case.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 6:

testaddDispCellPins ()

Derivation: The testaddDispCellPins () methods were created to ensure that the addDispCellPins() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*disp-cell-pins:…* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. These tests confirms that the invocation of the method to change the pin configuration is accurately represented in the scenario file. The tests also confirms that the incorrect resulting string, which does not have the cell number separated from the pin configuration, will not pass the test case.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 7:

testaddDispCellRaise ()

Derivation: The testaddDispCellRaise () methods were created to ensure that the addDispCellRaise() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*disp-cell-raise:…* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. This test confirms that accurate strings are put into the scenario file, and that incorrect spacing in between the fields will cause an error.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 8:

testaddDispCellRaise2 ()

Derivation: The testaddDispCellRaise2 () methods were created to ensure that the addDispCellRaise() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /*~disp-cell-raise:…* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. This test confirms that incorrect spacing in between the fields will cause an error.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 9:
testaddDispClearAll1 ()

Derivation: The testaddDispClearAll1 () methods were created to ensure that the addDispClearAll() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /*~disp-clearAll* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. This test confirms that the method does not insert any additional characters on the same line as the initial /~disp-clearAll.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 10:
testaddDispClearCell()

Derivation: The testaddDispClearCell () methods were created to ensure that the addDispClearCell() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: These tests have sufficient case coverage, as they test whether or not a /*~disp-clear-cell:…* line has been added to the scenario file. They are sufficient to confirm the functionality of the method.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 11:

testaddDispString1()

Derivation: The testaddDispString1 () methods were created to ensure that the addDispString() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*disp-string:…* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. This method confirms that an appropriate assigned string will be equal to the line in the scenario file where this method is executed.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 12:

testaddEndRepeat ()

Derivation: The testaddEndRepeat () methods were created to ensure that the addEndRepeat() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*endrepeat* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. This method confirms that an endrepeat tag will not be followed by any characters when present in the scenario file (after being added by the addEndRepeat method)

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 13:

testaddRepeat ()

Derivation: The testaddRepeat () methods were created to ensure that the addRepeat() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*repeat* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method.

These methods confirms that a repeat tag will not be followed by any characters when present in the scenario file (after being added by the addRepeat method). These methods also confirm that there are no spacing issues, additional characters, or misplaced characters.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 14:

testAddRepeatButton ()

Derivation: The testAddRepeatButton () methods were created to ensure that the addRepeatButton() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*repeat-button:...* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. These methods ensure that when the method is invoked with certain parameters, the changes in the scenario file reflect that change. Tests were also made for incorrect spacing and inaccurate strings in general.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 15:

testaddSkip ()

Derivation: The testaddSkip () methods were created to ensure that the addSkip() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*skip:...* line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. These methods confirm that identifiers are placed in front of /~skip:, and that no unnecessary spaces or extra characters are present after adding this line.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 16:

testaddSound ()

Derivation: The testaddSound () methods were created to ensure that the addSound () method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a /~*sound::*... line has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. These methods confirm that identifiers are placed in front of /~skip:, and that no unnecessary spaces or extra characters are present after adding this line.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 17:

testPause()

Derivation: The testPause() methods were created to ensure that the addPause() method was writing the correct scenario file syntax with the correct spacing to the current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a string has been added to the scenario file to be played back as speech.These tests are sufficient to confirm the functionality of the method. These methods ensure that nothing besides the string "/~pause:#" is added to the scenario file.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 18:

testsetupCellButton()

Derivation: The testsetupCellButton() methods were created to ensure that the setupCellButton() method was correctly implemented such that it followed the scenario file format, and was outputted to the scenario text file.

Sufficiency: These tests have sufficient case coverage, as they test whether or not the next question keyword has been added scenario file. These tests are sufficient to confirm the functionality of the method. The test cases ensure the method correctly sets up the # of cells and # of buttons in the scenario file.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 19:
testactivateKeys4()

Derivation: The testactivateKeys4() methods were created to ensure that the activateKeys() method was correctly implemented such that it followed the scenario file format, and was outputted to the scenario text file.

Sufficiency: These tests have sufficient case coverage, as they test whether or not the next question keyword has been added scenario file. These tests are sufficient to confirm the functionality of the method. The test case ensures the string "Test addition" has been added to the scenario file.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 20:
testactivateKeys4()

Derivation: The testactivateKeys4() methods were created to ensure that the activateKeys() method was correctly implemented such that it followed the scenario file format, and was outputted to the scenario text file.

Sufficiency: These tests have sufficient case coverage, as they test whether or not the next question keyword has been added scenario file. These tests are sufficient to confirm the functionality of the method. The test case ensures the string "Test addition" has been added to the scenario file.

Test Coverage:

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.7%	5	7	12

Test Case 21:
testsetKey()

Derivation: The testsetKey() methods were created to ensure that the setKey() method was correctly implemented such that it followed the scenario file format.

Sufficiency: This test has sufficient case coverage, as it tests whether or not the setkey keyword has been added scenario file. These tests are sufficient to confirm the functionality of the method. The test cases ensures the string "/~#” has been added to the scenario file. In particular, one of our test cases ensured the string “/~ONEE” was added to the scenario file.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 22:
testnextQuestion()

Derivation: The testnextQuestion() methods were created to ensure that the nextQuestion() method was correctly implemented such that it followed the scenario file format.

Sufficiency: These tests have sufficient case coverage, as they test whether or not the next question keyword has been added scenario file. These tests are sufficient to confirm the functionality of the method. The test case ensures the string "/~NEXTT” has been added to the scenario file.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Test Case 23:
testimportSound()

Derivation: The testimportSound() methods were created to ensure that the importSound() method was correctly importing sound files into our current selected scenario file.

Sufficiency: This test has sufficient case coverage, as it tests whether or not a sound file has been added to the scenario file. These tests are sufficient to confirm the functionality of the method. This method imports existing sound files from the users directory into a scenario file of their choosing, and parses it to be compatible with the scenario file and player. In particular, it checks to see if the text "/~sound:samplesound.wav" has been added to the scenario file.

Test Coverage:

Coverage	***Covered Branches***	***Missed Branches***	***Total Branches***
41.7%	5	7	12

Total Coverage:

LineEditorTest.java

Coverage	*Covered Branches*	*Missed Branches*	*Total Branches*
41.5%	249	351	600