2018

Testing Document

TREASURE BOX BRAILLE AUTHORING APPLICATION

EECS 2311 GROUP 3

Testing Document

Introduction

To provide the best results and good performance of the authoring app, some test cases were determined to provide the best user experience. These cases are of much important as they cover the basic functions that will be used by the user on each run.

Test Case	**Test Case Description**	**Test Case Results**
1.) Opens Scenario Text File	This test checks if the authoring app can open and read the specific scenario text file for reading and further modifications	**Pass**
2.) Save Scenario Text File	This test checks if the authoring app can save the specified scenario text file and overwrite changes to the edited content	**Pass**
3.) Create New Scenarios	This test checks if the authoring app can create new scenarios, populate the fields representing options with default text which can later be edited to personalize the scenario to user choosing	**Pass**
4.) Add New Event	This test checks if the scenario file can have a new event added in any sequence into the file, formatted correctly and neatly organized (This test checks for the Add User Input Button)	**Pass**
5.) Check If Valid Scenario File is Imported	This test checks if the imported scenario file is a valid scenario file (begins with Cell num1, Button num2) where num1 and num2 are valid integers representing the number of cells and buttons that the scenario file corresponds to.	**Pass**
6.) Create New Audio File	This test checks if a new audio file can be created with a name, and duration provided which then can be later imported into the scenario file to be incorporated into the scenario for audio use.	**Pass**

Discussion – How these Test Cases were Derived, Implemented and their Sufficiency

1. Opening the scenario file and reading the text file is one of the main tasks for this project as if file is mishandled while being imported, data corruption could incur causing the scenario file to misbehave and cause inconvenience to the end user. This test case was derived keeping the end-user in mind for providing user-friendly experience where

everything is smooth from the get go. This test case was implemented by making a new instance of JFileChooser and setting it to one of the test scenario files provided, then the file name was tested to check if the name of the file was recognized by the JFileChooser.

2. Saving a scenario file is equally important as opening a file as the scenario file is used only after it has been saved successfully. This test is meant to check if the scenario file is saved successfully to provide the playing app with the correct sequence of lines which are formatted properly. This test case was derived as a case to cover up the defects of saving files that might be read-only or may not be able to be overwritten. This test case should provide the user with a scenario file being created no matter the circumstances where the end-user doesn’t need to go through technical troubles of overwriting a file and losing previous work or not being able to save to a file due to read-only property of files being true. This test was implemented by creating a new test scenario file using the JFileChooser and saving it using the same method as in the Authoring App with some small test string saved as the sequence.

3. Creating new scenarios is another integral part of the authoring app as it provides the user with a scenario file from the get go with a dummy scenario file to get started with. Although the process might not be appealing, this provides the user to test out each feature of the scenario file and modify to their liking as they please. This test case was derived for user-friendliness of the features to the end-users as having to learn a new software is a hard task especially if it handles working with creating scripts for external devices that cannot be easily tested. This was implemented by using the same method as the one used in Authoring App which adds sample scenario code to the text area which then gets compared to the specific text in the test case for comparison.

4. Adding new events is a feature that allows the ability to add a state in the script where something happens of important. This allows the user to perhaps ask a question (an event) which can be triggered at a stage of the program, and provides the user the ability to add many important features to the scenario file. This is important because it provides the handler with different things to work with for example, each question could be an event and asked in their own space. This test was derived from the requirements of the authoring app as it was needed to provide the end-user to add events into the scenario file app. This was implemented by using addUserInputString () method which adds the “/~user-input” to the text area which is compared in the JUnit.

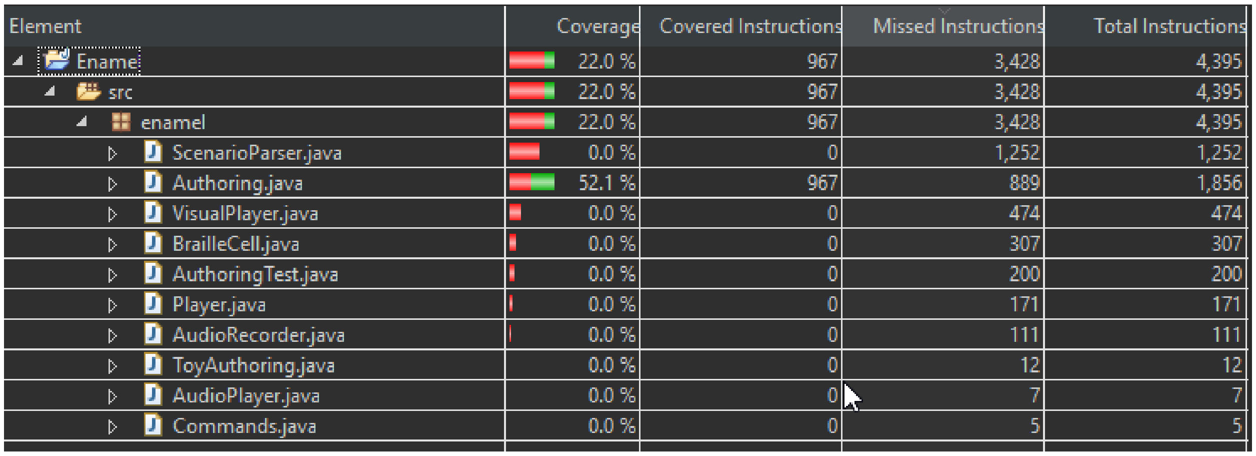
5. Checking the type of file being imported is important as not all text files are valid scenario files. A valid scenario file is recognized by the first line having Cell num1, followed by the second line having Button num2 where num1 and num2 are positive integer values representing the number of each components to have. This makes sure that these two cases are present when importing a file as if these two cases are not

present, then the imported file is not a valid scenario file. This is important because without this check in place, the user would be working on an invalid file which will not function with the braille simulator causing errors. This test as derived as a mandatory requirement of the authoring app because these two lines are required for the scenario file to function properly. This test was implemented by checking two different files, one file being one of the provided default scenario files and another file being a normal text file. A Scanner was used to check the first two lines of the text file for the presence of those two keywords (Button and Cell) and if those existed, the file was given the true attribute as being a scenario file and false was given otherwise.

6. Providing the user to record audio was another requirement of the authoring app to go with the sound feature of the scenario file where a sound is played during the execution of the scenario file script. This was derived as a way to check if the required audio recording function was working and generating a .wav file on execution. This is an important test because it is vital to check for the generation of the audio file being created after a recording to see if it functions properly. This test was implemented by calling the AudioRecorder class specifying a name for the test audio file and its duration. The test name chosen is tester1212 whereas the duration chosen is 1 second for simplicity of the test. Once the recording finishes, a file is supposed to be created. The test checks for the existence of this file to determine if the test has passed or not.

Test Coverage

Due to the testing cases being limited and still being worked on, the coverage percentage isn't as high as it can be for the whole project. As of now, the coverage for the entire project is 22% whereas the coverage for the main file being worked on is 52.1%. This does not heavily impact the state of the program but the goal is to eventually have a high coverage for the purpose of having a bug-free heavily test environment. An image of the coverage is shown below:



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Ename	22.0 %	967	3,428	4,395
src	22.0 %	967	3,428	4,395
enamel	22.0 %	967	3,428	4,395
ScenarioParser.java	0.0 %	0	1,252	1,252
Authoring.java	52.1 %	967	889	1,856
VisualPlayer.java	0.0 %	0	474	474
BrailleCell.java	0.0 %	0	307	307
AuthoringTest.java	0.0 %	0	200	200
Player.java	0.0 %	0	171	171
AudioRecorder.java	0.0 %	0	111	111
ToyAuthoring.java	0.0 %	0	12	12
AudioPlayer.java	0.0 %	0	7	7
Commands.java	0.0 %	0	5	5

Conclusion

Although the coverage results are not as high as expected, the test cases that were implemented all passed and their importance was in our opinion very high as these test cases test the main functions of the authoring app. All in all, our main goal is to improve the coverage as well as add more test cases that are of importance to this application.