Test Case Documentation

For

Player Audio Application for Braille-Based Device

Prepared by:
Dilshad Khatri
Drew Noel
Jonathan Tung
Alvis Koshy

Prepared on: March 7, 2017
For: EECS 2311 – Software Development Project

TABLE OF CONTENTS

Introduction

This document entails all the test cases that are present with the application. Each test corresponds to a function in the application that seemed to require testing to ensure that any issues could be seen and dealt with. Allowing to fix any form of logically errors that may have not been present during implementation.

1.0 ColourMapperTest

1.1 Purpose

The purpose of this test is to verify the behavior of the colour mapper. The colour mapper must map location tags to colours to make the UI more readable.

1.2 Derivation

The reason the test was derived was because we had to make sure that the events that were colored were properly colorized and that the colors correspond with the correct events and are mapped when required.

1.3 Implementation

Method Name	Description	Expected Result	Test
test()	This test creates two ColourMapper objects, one filled with colours and one empty and then checks to see if the test passes	Asserts true when the empty object is empty and the filled object correctly returns the colours	The test passes which means that the test passed and it was properly mapped

1.4 Sufficiency

This test is sufficient because all we needed to check was if the test was properly mapped and if it returns the correct colours.

2.0 CommandTests

2.1 Purpose

This class represents the user interface of the authoring program. It is responsible for creating all the events in the scenario panel. Each option in its drop-down menu corresponds to a command and the testing is done for each individual command.

2.2 Derivation

This test was derived so we could see if the commands are properly implemented and create the correct output. It allows us to check for any form of logical errors that may have been missed during the implementation of the commands.

2.3 Implementation

All the methods in this test have the same approach in testing since all the commands are a child of PlayerCommand that have five abstract methods. These tests check those methods for each command with the exception of some that do not utilize the mutator method. They check the toString(), serialize(), getEditLabel(), getCurrentValue(), and the setCurrentValue()

Method name	Description	Expected Result	Test
testCellCharCommand()	Tests all methods in CellCharCommand. Checks the entire length of the alphabet and compares it with each expected outcome	Asserted as equal when the string that is input into the command is equal to the expected outcome	The test passes which means that the string comparison held true and it was properly tested.
testCellLowerCommand()	Tests all methods of CellLowerCommand. Tests to see whether the braille cell and pin number is properly output by comparing with the expected output	Asserted as equal when the input is equal to the expected outcome through all iterations	The test passes which means that through all iterations it returned the correct otuput
testCellRaiseCommand()	Tests all methods of CellRaiseCommand. Tests to see whether the braille cell and pin number is properly output by comparing with the expected output	Asserted as equal when the input is equal to the expected outcome through all iterations	The test passes which means that through all iterations it returned the correct otuput
testClearAllCommand()	Tests all methods in ClearAllCommand. Compares the expected string with the string from the class.	Asserted as equal when the string from the class is equal to the	The test passes which means that it returned the correct output

		expected output	for all methods.
testClearCellCommand()	Tests all methods of the CellClearCommand. Compares the expected string with the result.	Asserted as equal when the input is equal to the expected outcome through all iterations	The test passes which means that through all iterations it returned the correct otuput
testGoHereCommand()	Tests all methods in GoHereCommand. Compares the expected string with the string from the class.	Asserted as equal when the string from the class is equal to the expected output	The test passes which means that it returned the correct output for all methods.
testPauseCommand()	Tests all methods of the PauseCommand. Tests whether the command properly outputs the correct string when compared with the expected	Asserted as equal when the input is equal to the expected outcome through all iterations	The test passes which means that through all iterations it returned the correct otuput
testRepeatButtonCommand()	Tests all methods of the RepeatButtonCommand. Tests whether the command properly outputs the correct string when compared with the expected.	Asserted as equal when the input is equal to the expected outcome through all iterations	The test passes which means that through all iterations it returned the correct otuput
testRepeatCommand()	Tests all methods in RepeatCommand. Checks the entire length of the alphabet and compares it with each expected outcome	Asserted as equal when the string that is input into the command is equal to the expected outcome	The test passes which means that the string comparison held true and it was properly tested.
testResetButtonCommand()	Tests all methods in ResetButtonCommand. Compares the expected	Asserted as equal when the string	The test passes which means that it

	string with the string from the class.	from the class is equal to the expected output	returned the correct output for all methods.
testSetPinsCommand()	Tests all methods of the SetPinsCommand. Tests whether the command properly outputs the correct string when compared with the expected.	Asserted as equal when the input is equal to the expected outcome through all iterations	The test passes which means that through all iterations it returned the correct otuput
testSetStringCommand()	Tests all methods in SetStringCommand. Checks the entire length of the alphabet and compares it with each expected outcome	Asserted as equal when the string that is input into the command is equal to the expected outcome	The test passes which means that the string comparison held true and it was properly tested.
testSetVoiceCommand	Tests all methods in SetVoiceCommand. Goes through 5 iterations that each represent a unique voice. It then compares it with the expected outcome.	Asserted as equal when the input is equal to the expected outcome through all iterations	The test passes which means that through all iterations it returned the correct otuput
testSkipButtonCommand()	Tests all methods in SkipButtonCommand. Checks the entire length of the alphabet and compares it with each expected outcome	Asserted as equal when the string that is input into the command is equal to the expected outcome	The test passes which means that the string comparison held true and it was properly tested.
testSkipCommand()	Tests all methods in SkipCommand. Checks the entire length of the alphabet and compares it with each expected outcome	Asserted as equal when the string that is input into the command is equal to the	The test passes which means that the string comparison held true and it was

		expected outcome	properly tested.
testSoundCommand()	Tests all methods in SoundCommand. Checks the entire length of the alphabet and compares it with each expected outcome	Asserted as equal when the string that is input into the command is equal to the expected outcome	The test passes which means that the string comparison held true and it was properly tested.
testTTSCommand()	Tests all methods in TTSCommand. Checks the entire length of the alphabet and compares it with each expected outcome	Asserted as equal when the string that is input into the command is equal to the expected outcome	The test passes which means that the string comparison held true and it was properly tested.
testUserInputCommand()	Tests all methods in UserInputCommand. Compares the expected string with the string from the class.	Asserted as equal when the string from the class is equal to the expected output	The test passes which means that it returned the correct output for all methods.

2.4 <u>Sufficiency</u>

This test is sufficient because it goes through and checks through each individual command and checks to see if they all pass the test. Even though most of the tests are very similar, their slight differences are still worth testing just incase because they could potentially help during issues where errors might arise.

3.0 <u>ExportErrorCheckTest</u>

3.1 <u>Purpose</u>

This class extensively tests the ExportErrorCheck class. If given improper command orders, the application must determine that there is a problem with the given sequence of commands. This set of tests puts the responsible class under several scenarios to test functionality.

3.2 Derivation

This class was derived to make sure that, when saving the file, no irregularities or errors are present. We make sure to check this because we did not want to create scenarios that have initialized the bounds of the cells and buttons and then allow the user to exceed those bounds.

3.3 Implementation

Method Name	Description	Expected Result	Test
testCheckUserInput()	This test checks the userInput static method to see if the application correctly catches any missing user input commands. Has multiple scenarios that test the correct scenario	All tests in the method return correct, showcasing that with every variation of user input command, it returns the expected result	Tests successfully passes which means that every combination of user input command worked.
testCheckCellNumber()	Tests that the application correctly detects when braille cells are referenced but were never defined. Contains multiple scenarios that test to see if its implemented properly	All tests in the method are correct, showcasing that with ever variation of the methods that utilize cells, the expected outcome is reached	Tests are successful, which means that every combination of cell commands that are used do not incorrectly declare a word that is longer than the cell number
testCheckButtonNumber()	Test that the application correctly detects when buttons are referenced or used but never defined in the settings. Contains multiple scenarios that test to see if	All the tests in the method are correct which showcase variations of commands that utilize the buttons on the hardware, and with each	Tests are successful which means that with ever combination of button commands, there is never a declaration of a button that passes

	it is implemented correctly.	variation, the expected result is reached.	and is beyond the initial number.

3.4 Sufficiency

These tests are sufficient because they go through every variation of commands that utilize both buttons and cells. This makes sure that whenever the user tries to save their scenario, they haven't accidentally created items that exceed the bounds.

4.0 TestImportExport

4.1 Purpose

This simple test class tests all methods in ExportListener and ImportListener classes. This test class gives us a check to make sure that both the import and export listeners are properly invoked and no logical errors are present.

4.2 Derivation

The basis of how we derived the test is to make sure that files contain the proper file format both during importing and exporting of files.

4.3 Implementation

Method Name	Description	Expected Result	Test
testExportSimple()	A simple method that tests all methods in ExportListener(). We create a test scenario and parse the test scenario through the export listener and compare it with a string that has the proper file format.	The expected result is that when the list is properly parsed and compared to the expected result, they are asserted as equal and return true.	The test is successful, meaning that the test was able to properly compare the parsed list with the expected one.
testImportSimple()	A simple method that tests all methods in ImportListener(). We create a string with the proper file	The expected result is when the string is compared to the list, and they assert as equal and return true. This test does	The test is successful, meaning that the test properly compared the

	format and then create the expected result as a scenario and then compare the two.	an addition check to make sure the size is also equal.	string with the expected list.

4.4 <u>Sufficiency</u>

The test is sufficient because it checks for the file formatting when trying to import and export. This ensures that the user is using the correct format when making and uploading scenarios.