TALKBOX TESTING

FEBRUARY **24**, **2019**

VERSION 1.0

GROUP 9

Authored by: Neharika Puri, Eric Pham, Yonis Abokar

Table of Contents

1. Introduction

1.1 Purpose

The purpose of the testing document will keep track of any bugs through testing and debugging. The document will track through the use of test cases and test coverages. The documentation will discuss each test and explain its significance in its finding. The documentation will also discuss the significance of the test coverage.

2. Unit Testing

2.1 Test Method

The unit testing will be done using the latest JUnit framework, JUnit 5. The functionality of each class was tested as individual methods.

2.2 Test Cases

Test Class/ Tester	Test Case Description
AudioClip/ AudioClipTest	AudioClipTest tests if the audio successfully plays. It tests this by using the “boring” wav file as a test file.
FileInputOutput/ FileInputOutputTest	FileInputOutput tests path finding and file retrieving
Serializer & TalkBoxConfiguration/ TestSerialization	Description- TestSerializtion will test whether the configurator successfully serializes and deserializes the data in TalkBoxData.tbc using a specific configuration.
Sound/ TestSound	Description – Sound will test the recording feature

2.2.1 Test Case Derivation & Significance

i) AudioClipTest

Derivation: AudioClipTest comprises of two tests; testAudioClipStartsSuccessfully & testAudioClipThrowsException. These tests were derived due to errors received when the path to the audio files were incorrect.

Significance: Successfully passing the AudioClipTest signifies that the TalkBox device will be able to successfully play audio through the correct path

ii) FileInputOutput

Derivation: In our TalkBox configuration app, it displays all wav files found in a certain directory. Using this, the user and choose which audio files they want. These tests were derived due to errors when starting the configurator and the path was incorrect.

Significance: Successfully passing all FileInputOutput tests signifies that the TalkBox device will be able to start properly and find all audio files

iii) Serializer/TalkBoxConfiguration

Derivation: The simulator works by taking in a TalkBoxData.tbc, which is created by the TalkBox configuration app, and deserializes it. It then takes all the information and forms the simulator app. These tests were derived due to the dependency of the simulator app. The correct information needs to be serialized in order to create the simulator.

Significance: Successfully passing the given Serialization tests signifies the correct information was serialized and deserialized, allowing the simulator app to be made.

iv) Sound

Derivation: The TalkBox configurator needs allow users to successfully record their own audio. Using a recording class, it is able to record audio and save it as any filename the user wants. These tests were derived in order to test if a wav file is successfully created and stored.

Significance: Successfully passing all the Sound tests signifies the TalkBox can successfully record audio.

3. GUI Testing

3.1 Test Method

For testing the GUI, manual testing was used to test the configurator and simulator.

3.2Test Cases

Test Class/ Item Tested	Test Case Description

ButtonsGui & GuiConfig/ NumofButtons TextField, GridPane & ScrollPane GuiConfig Import Audio Menu/List View	When entering the desired number of buttons, the GUI will add the desired amount to the Configurator. The user can import any wav file from their personal computer, and it will store in the Audio directory of the TalkBox
GuiConfig/ Profile TextField & TreeView	When entering a profile, the profile is added to the TreeView
GuiConfig/ Add Sound Button & ListView	Sound successfully adds to the desired profile
GuiConfig/ Set Profile Button	Buttons successfully changes name to audio
AudioClip/GuiConfig Audio Buttons	Buttons successfully plays desired audio
GuiConfig/GUI Launch	Simulator successfully launches from configurator.

3.2.1 Test Case Derivation and Significance

i) NumofButtons TextField, GridPane & ScrollPane

Derivation- TalkBox must allow users to enter as many buttons as they desire.

Significance- Successfully creating the correct number of buttons signifies the NumOfButtons Textfield, GridPane and ScrollPane successfully works.

ii) Import Audio & ListView

Derivation- A feature that was implanted was to allow users to import their own wav files from their personal computers and import them into the TalkBox. The audio will then appear at the bottom of the ListView.

Significance- Successfully importing audio signifies users will not only be allowed to record but also import any wav file they may already have.

iii) Profile TextField & TreeView

Derivation- Users can create their own profile where they can store desired audio files.

Significance- Successfully creating profile signifies the Profile TextField and TreeView both successfully work together

iv) Add Sound Button & ListView

Derivation- The configuration contains a ListView which displays all the available audio files. The user can then add the audio file to their desired profile.

Significance- Successfully adding sound to profile signifies that the users will be able to customize which sounds they desire.

v) Set Profile Button

Derivation- The user can press set profile to associate the sound to the buttons. It will then be displayed in the TalkBox preview.

Significance- Successfully adding sound to button signifies which buttons will play which audio file

vi) Audio Buttons

Derivation- TalkBox buttons must be able to play sound

Significance- Successfully playing sounds signifies the TalkBox is able to play audio

vii) Launch Button

Derivation- The user can launch the simulator from the configurator with their desired settings

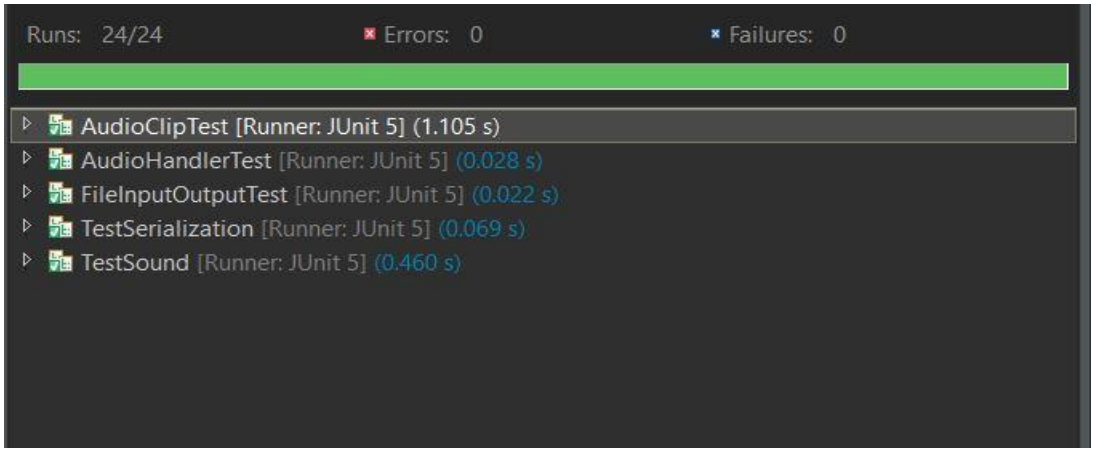
Significance- Successfully launching the simulator signifies the functionality of the configurator and the simulator works as intended.

4. Test Coverage

4.1 Coverage Method

To calculate test coverage, EclEmma was used.

4.2 JUnit Test Results



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TalkBox	24.8 %	734	2,231	2,965
src	24.8 %	734	2,231	2,965
TalkBoxConfig	14.6 %	263	1,533	1,796
TalkBoxSim	0.0 %	0	659	659
Tester	92.4 %	471	39	510
FileInputOutputTest.java	74.1 %	60	21	81
TestSerialization.java	97.8 %	268	6	274
TestSound.java	93.5 %	87	6	93
AudioClipTest.java	88.5 %	23	3	26
AudioHandlerTest.java	91.7 %	33	3	36

4.2.1 Discussion

The JUnit tests results show that all tests pass, meaning the TalkBox functionalities should all work as intended. The coverage calculated by EclEmma, is seen to be at only 24.8%. This is because the GUI code is an executed through the test cases. The test cases do not test the GUI and the event fired, rather, it tests the functionality each the given classes. All the functionality classes are stored in TalkBoxConfig, which explains why the coverage for the TalkBoxSim is at 0%.

4.3 Manual Testing Results

GuiConfig (24-Feb-2019 8:25:43 PM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
TalkBox	77.5 %	2,414	702	3,116
src	77.5 %	2,414	702	3,116
Tester	0.0 %	0	510	510
TalkBoxConfig	93.1 %	1,821	134	1,955
AudioClip.java	50.7 %	36	35	71
GuiConfig.java	98.4 %	1,195	19	1,214
Sound.java	86.8 %	118	18	136
TalkBoxConfiguration.java	25.0 %	6	18	24
FileInputOutput.java	75.0 %	42	14	56
Serializer.java	78.7 %	37	10	47
ImportAudio.java	87.3 %	62	9	71
ButtonGui.java	96.3 %	181	7	188
AudioHandler.java	76.5 %	13	4	17
Load.java	100.0 %	131	0	131
TalkBoxSim	91.1 %	593	58	651
Buttons.java	78.7 %	144	39	183
Profiles.java	94.6 %	157	9	166
Gui.java	96.7 %	174	6	180
Audio.java	96.7 %	118	4	122

4.3.1 Discussion

The manual testing done resulted in desired outcome which is that the both the TalkBox configuration app and the TalkBox simulator works as intended. The coverage data shows a percentage of 75.9%, much higher than the JUnit testing. The testing package shows 0% as it the JUnit tests are never used in manual testing. When adding both the JUnit coverage and the manual coverage, the total coverage comes up to 100%. The true coverage is less since both the JUnit coverage and the manual coverage both use some of the same classes. Nevertheless, when both the coverages are added, it shows that almost all the code is used when running the apps.