Testing Document

TablatureConverter

EECS 2311 – Group 6

Tuan Dau

James Le

Temiloluwa Odunfa

Mohammad Amin Mohammadi

02/28/2021

Table of Contents

Testing Methodology

Test cases are implemented in the TablatureConverter project in order to ensure smoothness in the user experience, as well as to streamline development by reducing the likelihood of encountering code faults later in development. All of the unit tests conducted in the TablatureConverter project are written in JUnit 5.

Back-End Tests

The main focus of the midterm submission unit testing is towards all of the classes that have to do with file parsing. The main class being tested is the textReader file, which is the class that parses tablature files given from the user, and is one of the most used classes in the project. Another class that was a focus of testing is the Output class, which is an object class that is used to transfer information about notes and new measures across the project. All test cases are included by making one test class for every class that needs to be tested, such as Output and textReader. Each method has one test case associated with it to ensure reliability.

The testing is sufficient for the midterm submission, as it ensures the basic functionality of each method, which can be used to build the overarching system. Although the testing is sufficient, there will need to be a lot more tests written as development resumes in order to ensure maximum reliability of the included methods, as well as the remaining and future classes included in the project.

1. Output Test

testOutput(): Tests that all constructors of the Output class can be run correctly. This test checks that the outputted object files against the expected fields that it should have for a given constructor run.

testGetletter(): Tests that the created Output objects return the correct step (note value) stored in each Output object.
testGetnote1(): Tests that the created Output objects return the correct note1 field and that the return type is of the correct return type.

testGetnote2(): Tests that the created Output objects return the correct note2 field and that the return type is of the correct return type.

testGetindex(): Tests that the created Output objects return the correct index field and that the return type is of the correct return type.

testGettech(): Tests that the created Output objects return the correct technique field and that the return type is of the correct return type.

2. TextReader Test

testParsGuitar(): Tests that bass tablature is being parsed properly by comparing the ParsGuitar()'s output to a manually generated string of what the team's expectations are for the given test tablature input.

testParsBass(): Tests that bass tablature is being parsed properly by comparing the ParsBass()'s output to a manually generated string of what the team's expectations are for the given test tablature input.

testParsDrum(): Tests that bass tablature is being parsed properly by comparing the ParsDrum()'s output to a manually generated string of what the team's expectations are for the given test tablature input.

testisInteger(): Test if the string given is an integer. A string is defined to be a valid integer when there are only numeric characters in the string, there are no decimal places, and that the represented number value in the string is less than (2^31)-1.

testTabIsOK(): Tests to see if the format of the inputted tablature is valid, as defined by the rules in Section 3.1, item 3 of the Group 6 TablatureConverter Requirements Document.

testgetCharFromString(): Tests to see if a character can be taken from a string, given a string input, and an index given as an integer. The return boolean of getCharFromString() returns, and false if an indexOutOfBoundsException() is detected, and returns true if there is no indexOutOfBoundsException() exception. The test passes if the method returns the expected return boolean, and returns false otherwise.