Da Capo

Testing Document

Version 1.0
Date: 02/28/2021

Gong-Fan (Billy) Bao - 213563150
Syed Quadri - 216204554
Elijah Nnorom - 214502504
Miguel Mesa Gonzalez - 216138323
Ayub Osman Ali - 217612847

Table of Contents

Table of Contents **2**

1. Introduction **3**

1.1 Testing Overview 3

2. TextFileReader Testing **4**

2.1 File Reader Tests 5

3. Guitar Parser Testing **6**

3.1 Duration Tests 6

3.2 Type Tests 8

3.3 Notes Tests 9

3.4 Fret Number Tests 10

3.5 Fret String Tests 10

3.6 Chord Tests 10

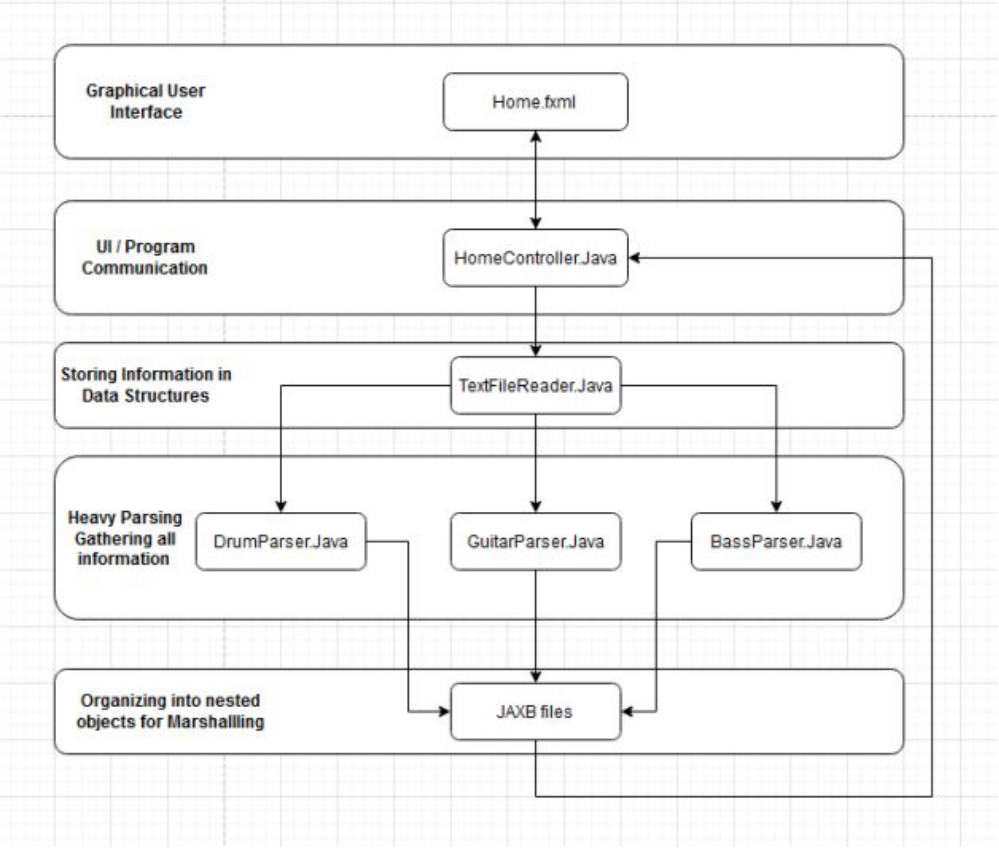
4. Drum Parser Testing **12**

4.1 Drum Parser Tests 12

1. Introduction

1.1 Testing Overview

This testing document serves to provide the reader a detailed look at the various classes in the De Capo Tab to MusicXML Generator Application. The sections of this document can be broken down into three main sections, each of which cover different Sections of the information processed internally.



The First section of testing testing the TextFileReader.Java Class which contains all the initial gathering of information.

The Second section focuses on the GuitarParser Class which is currently the most developed class of the three instruments. As such, the testing is quite detailed and thorough.

The Third section focuses on the DrumParser Class. It makes use of the large constructor to complete the parsing process and checks out the various outputs.

2. TextFileReader Testing

This section provides an overview of the test cases for the Text File Reader. The File Reader is the first part of the software and all other parts of the software depend on it, if the File Reader crashes or provides an unexpected result the software as a whole will be affected. This section provides the test cases that have been written to thoroughly test the Text File Reader, an explanation of what each test case test has also been provided under the name of each test case.

2.1 File Reader Tests

2.1.1 textFileReader1()

Checks to see whether the object for the TextFileReader.java is created successfully without crashing. This is important as it ensures that provided that a string representing the name/path of the file is given, the file reader creates the object without crashing.

2.1.2 textFileReader2()

Checks to see whether the instrument detection feature automatically detects the instrument being translated, and the name of this instrument is returned. This ensures that the name of the instrument that has been automatically detected matches the correct instrument’s tablature.

2.1.3 textFileReader3()

Checks to see whether the number of lines for the tablature of the instrument being translated is correctly returned. This number should depend on the instrument being provided so this test case will work for drums and bass guitars as well.

2.1.4 textFileReader4()

Checks to see whether the first column in the provided tablature, which corresponds to the base notes matches that of the tablature for the instrument that has been automatically detected. The first column for the guitar and bass could be similar, but at the moment this method only checks that of a guitar. If provided with a drum or bass tablature it could be modified to check either one of them as well.

3. Guitar Parser Testing

The following test cases are for confirming that the Guitar Parser class performs as intended. They cover the generation of the Duration, Type, Note, Fret Number, Fret String, and Chord

arrays. Currently, any note durations that are not whole, half, quarter, eight, or 16th are defaulted to 16th.

3.1 Duration Tests

3.1.1 testDuration1()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord 16th notes.

3.1.2 testDuration2()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord eighth notes.

3.1.3 testDuration3()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord quarter notes.

3.1.4 testDuration4()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord half notes.

3.1.5 testDuration5()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of a non-chord whole note.

3.1.6 testDuration6()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists of a mixture of half, quarter, eighth, and 16th non-chord whole notes.

3.1.7 testDuration7()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists of chordal whole notes.

3.1.8 testDuration8()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists of a mixture of chordal half, quarter, eighth, and 16th notes.

3.1.9 testDuration9()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists of a mixture of irregular/dotted notes.

3.2 Type Tests

3.2.1 testType1()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord 16th notes.

3.2.2 testType2()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord eighth notes.

3.2.3 testType3()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord quarter notes.

3.2.4 testType4()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord half notes.

3.2.5 testType5()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of a non-chord whole note.

3.2.6 testType6()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists of a mixture of half, quarter, eighth, and 16th non-chord whole notes.

3.2.7 testType7()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists of chordal whole notes.

3.2.8 testType8()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists of a mixture of chordal half, quarter, eighth, and 16th notes.

3.2.9 testType9()

Checks to see whether the parseToRhythm method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists of a mixture of irregular/dotted notes.

3.3 Notes Tests

3.3.1 testNotes1()

Checks to see whether the translateParsed method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord 8th notes.

3.3.2 testNotes2()

Checks to see whether the traslateParsed method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of chordal whole notes.

3.4 Fret Number Tests

3.4.1 testFretNumber1()

Checks to see whether the translateParsed method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord 8th notes.

3.4.2 testNotes2()

Checks to see whether the traslateParsed method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of chordal whole notes.

3.5 Fret String Tests

3.5.1 testNotes1()

Checks to see whether the translateParsed method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord 8th notes.

3.5.2 testNotes2()

Checks to see whether the traslateParsed method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of chordal whole notes.

3.6 Chord Tests

3.6.1 testNotes1()

Checks to see whether the translateParsed method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of non-chord 8th notes.

3.6.2 testNotes2()

Checks to see whether the traslateParsed method in GuitarParser.java returns the correct values, when given a Guitar tab that is a measure long and consists entirely of chordal whole notes.

4. Drum Parser Testing

The following test cases are for confirming that the Drum Parser class performs as intended. Although there are not many, they cover the complete process of parsing from start to end with passing of the tablature as Strings, the correct preprocessing of information and a minor check on the final output of Measure containing Notes that will be used to generate the output XML.

4.1 Drum Parser Tests

4.1.1 test1()

Checks to see if the drumparser object is being created correctly and that the constructor is correctly instantiating and storing the information that is being passed to it. In the case of this test, that information is an arraylist of string and the tester is confirming that the lines of information it has stored are equal to the lines of the tablature;

4.1.2 test2()

Checks to see if the drumparser object that is being passed a tablature as a group of strings is parsing and storing the information correctly in an arraylist of measures that hold an arraylist of several notes. In this case, the length of the arraylist containing measures was used as an indicator of this property. This will be improved and broken down into sections in the future.

4.1.3 test3()

Checks to see if the drumparser object that is created stores the information passed as an arraylist of strings is correctly transposed before beginning the parsing process. This test ensures that the pre processing is done correctly which makes the other tests and processes more reliable. This will be updated in the future with several variations of of tablature passed to it.