

# TAB2MUSIC TESTING DOCUMENT

EECS 2311 SOFTWARE DEVELOPMENT PROJECT

---

## **Group 11**

March 2022

John Yacoub  
Muhammad Sawal  
Shaylin Ziaei  
Akarshan Kakkar

## Table of Contents

<b>1. Testing Methodology</b>	<b>2</b>
1.1 Objective	2
1.2 Derivation	2
<b>2. Test Cases</b>	<b>2</b>
2.1 XMLParser	2
2.1.1 testXMLParser1	2
2.1.2 testXNumberOfNotes	2
2.1.3 testNumberOfMeasures	3
2.1.4-2.1.9	3
2.2 XMLParser Sufficiency	3
2.3 Test Guitar Parser	3
2.3.1 Test Strings	3
2.3.2 Test Frets	3
2.3.3 Test Notes	3
2.3.4 Test NotesLength	4
2.3.5 Test Chords	4
2.3.5 - 2.3.10	4
2.4 Test Guitar Parser Sufficiency	4
2.5 Test Drum Parser	4
2.5.1 testID	4
2.5.2 testName	4
2.5.3 testNotes	4
2.5.4 testChords	5
2.6 Test Drum Parser Sufficiency	5
2.7 Test Play Notes	5
2.7.1 TestPlaying	5
2.8 Test Play Notes Sufficiency	5
<b>3. Test Coverage Report</b>	<b>6</b>
3.1 Packages	6
3.1.1 Parser	6
3.1.2 Play Notes	7
3.1.3 Music Notes	7

## **1. Testing Methodology**

### **1.1 Objective**

The goal of testing our application was to ensure that the main features of the application are working. Testing was conducted for all main methods and covers all major classes implemented. All tests have been done using JUnit Testing. Tests have been conducted using several styles and versions of tab input (for guitar and drums.)

### **1.2 Derivation**

We developed our tests based on testing all features of our application with multiple different inputs to ensure each feature works as expected and that it dealt with edge cases/harder inputs in a correct way.

## **2. Test Cases**

### **2.1 XMLParser**

All the test cases in this class take XML input from a helper method and store it in a string parse. A new Xml parser object is created which parses the string to test the correctness of our parser. There are three test cases for XMLParser, three tests to check the correctness of the number of notes in different measures and three test cases to compare the number of measures of tablature with expected value. These tests have been implemented for both drum and guitar.

#### **2.1.1 testXMLParser1**

In this test case, the name of the instrument which is given by the program, is compared with the expected name that is guitar.

#### **2.1.2 testXNumberOfNotes**

This test compares the number of notes for the first and second measure which should be 8 and 14 respectively.

### **2.1.3 testNumberOfMeasures**

The number of measures in the tablature is checked to be equal to expected value which is 2 in this test case.

### **2.1.4-2.1.9**

Test cases 2.1.4 - 2.1.9 perform the same tests on a different XML to further check the correctness of our parser.

## **2.2 XMLParser Sufficiency**

The TestParser class checks the correctness of our XMLParser class. It checks that we get the correct instrument name so that the XML is parsed accordingly. It also tests the values of total number of notes and number of notes per measure which are later on needed for parsing.

## **2.3 Test Guitar Parser**

All the test cases in this class take XML input from a helper method and tests are based on the expected values that should be parsed from the XML. XMLParser and GuitarParser objects are created to test the correctness of our guitar parser class. The following XML elements are tested:

### **2.3.1 Test Strings**

This test case compares the values of the Strings that we get from parsing to their actual values in the XML.

### **2.3.2 Test Frets**

This test case compares the values of the frets that we get from parsing to their actual values in the XML.

### **2.3.3 Test Notes**

This test case checks whether or not our parsing returns correct notes in the correct order.

### **2.3.4 Test NotesLength**

This test case checks the length of each note to its expected value in the XML.

### **2.3.5 Test Chords**

This test case checks if the correct notes are being played as a chord.

### **2.3.5 - 2.3.10**

Test cases 2.3.5 - 2.3.10 perform the same tests on a different XML to further check the correctness of our parser.

## **2.4 Test Guitar Parser Sufficiency**

The test cases in this class test all the components that are used in playing and visualising the guitar tabs. They check if the information that we get from parsing is correct, and is in the correct order, so that our notes are played and visualised correctly.

## **2.5 Test Drum Parser**

All the test cases in this class take XML input from a helper method and tests are based on the expected values that should be parsed from the XML. XMLParser and DrumParser objects are created to test the correctness of our drum parser class. The following XML elements are tested:

### **2.5.1 testID**

This test checks if the ID of the instrument is equal to the expected ID which is different based on the type of instrument.

### **2.5.2 testName**

The test checks if the name of the instrument is the same as the name that is expected.

### **2.5.3 testNotes**

The values stored in the note list after parsing are compared with their expected values.

#### **2.5.4 testChords**

The values that are stored after parsing an array which contains chords are checked with the expected value based on the given tablature.

### **2.6 Test Drum Parser Sufficiency**

The test cases in this class test all the components that are used in playing and visualising the drum tabs. They check if the information that we get from parsing is correct, and is in the correct order, so that our notes are played and visualised correctly.

### **2.7 Test Play Notes**

All the test cases in this class take XML input from a helper method. The class is designed to test the correctness of our JfugueTest and Jfugue for drums classes.

#### **2.7.1 TestPlaying**

This test case tests our final output string (contains the notes with alters) which is used for playing. It checks if the value of the string matches to what's expected from the xml.

TestPlaying2 performs the same test using different guitar XML inputs.

TestPlayingDrums performs the same test for drum XML inputs.

### **2.8 Test Play Notes Sufficiency**

The test cases in this class check if the correct notes are marked as altered notes. It also tests if the notes are correct and are in correct order, so that the tabs are played accurately.

## 3. Test Coverage Report

### 3.1 Packages

#### TAB2XML\_G14

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
MusicNotes		3%		0%	116	120	529	578	27	31	4	5
GUI		0%		0%	226	227	858	860	122	123	16	17
converter.note		39%		32%	202	299	370	687	66	125	4	14
converter		63%		60%	78	153	137	431	15	49	1	7
Parser		76%		57%	78	135	139	682	0	10	0	3
converter.instruction		36%		27%	78	101	139	222	16	35	1	6
converter.measure		67%		57%	67	145	128	390	8	42	0	4
PlayNotes		66%		65%	22	43	64	181	11	21	1	4
models.measure.note		0%		0%	71	71	123	123	69	69	11	11
models.measure.note.notations.technical		0%		0%	57	57	106	106	53	53	9	9
utility		88%		76%	44	115	72	336	27	67	3	12
models.measure.note.notations		0%		0%	36	36	67	67	32	32	4	4
models.measure.attributes		0%		n/a	36	36	66	66	36	36	6	6
models		0%		n/a	33	33	62	62	33	33	5	5
models.part_list		0%		n/a	24	24	51	51	24	24	5	5
converter.measure_line		69%		50%	15	31	23	73	2	13	0	4
models.measure.barline		0%		n/a	21	21	30	30	21	21	3	3
models.measure		0%		n/a	18	18	28	28	18	18	2	2
models.measure.direction		0%		n/a	15	15	21	21	15	15	3	3
custom_exceptions		0%		n/a	4	4	8	8	4	4	4	4
Total	14,888 of 24,409	39%	1,016 of 1,652	38%	1,241	1,684	3,021	5,002	599	821	82	128

The report shows the test coverage for all our major packages. The package that was tested most extensively was “Parser” with 76% test coverage. Packages “Playnotes” and “Music notes” were covered 61% and 3%, respectively.

#### 3.1.1 Parser

##### Parser

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
GuitarParser		70%		52%	50	77	93	352	0	3	0	1
XMLParser		67%		53%	9	18	23	94	0	4	0	1
DrumParser		88%		68%	19	40	23	236	0	3	0	1
Total	763 of 3,248	76%	107 of 250	57%	78	135	139	682	0	10	0	3

All the classes in the “Parser” package were tested to a sufficient extent. The drumParser had a coverage of 88%, GuitarParser of 70%, and XMLParser of 67%. Since getting the parsing correct was a huge part of playing and visualising the notes we made sure to test this package thoroughly.

It may seem like 67% or even 70% coverage might not be enough testing for parsing, but every element of all the ArrayLists that we used for playing or visualising was tested against the expected value from the XML. Overall, we are happy with the amount of testing done for parsing.

### 3.1.2 Play Notes

#### PlayNotes

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
<a href="#">MidiDrum</a>		35%		0%	8	9	44	54	7	8	0	1
<a href="#">JfugueForGuitar.new AnimationTimer() {...}</a>		0%		n/a	2	2	6	6	2	2	1	1
<a href="#">JfugueForDrum</a>		87%		63%	8	20	8	84	0	5	0	1
<a href="#">JfugueForGuitar</a>		87%		83%	4	12	7	38	2	6	0	1
Total	316 of 947	66%	15 of 44	65%	22	43	64	181	11	21	1	4

All the classes in the “Play Notes” package were also tested thoroughly.

The coverage for both the JfugueForDrum, which plays the drums, and JFugueForGuitar, which is used to play guitar, is 87%. The test cases for these classes test the string “total” which is a collection of all the notes and is used to play the instrument.

The MidiDrum class was not tested as extensively because it does not play a huge role in playing the notes, and is only used as a helper class.

### 3.1.3 Music Notes

#### MusicNotes

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
<a href="#">CanvasNotes</a>		4%		0%	90	94	395	444	13	17	0	1
<a href="#">CanvasNotes.new Runnable() {...}</a>		0%		0%	11	11	41	41	2	2	1	1
<a href="#">GuitarNotes</a>		0%		n/a	3	3	40	40	3	3	1	1
<a href="#">DrumNotes</a>		0%		0%	10	10	42	42	7	7	1	1
<a href="#">DrumNotes.new JFrame() {...}</a>		0%		n/a	2	2	13	13	2	2	1	1
Total	4,315 of 4,481	3%	178 of 178	0%	116	120	529	578	27	31	4	5

Since GUI testing was not a requirement for the project we did not spend a lot of time writing test cases related to visualisation. However, through detailed testing of the parsing we made sure that the correct notes are visualised and the output was also compared with musescore on a regular basis.



