TAB2XML

Testing Document

Final Version

04/11/2021

Written By:	Ziqi Zhou	214726283
	Hargovind Singh	217303025
	Ali Yamany	216513269
	Amer Alshoghri	214992291
	Uwais Kazi	215263940

Table of Contents

List of Figures

1. Introduction

This Testing Document (TD) provides the information necessary for quality assurance to effectively use TAB2XML final release. This TD also details existing unit tests and current testing coverage.

2. Overview

TAB2MXL is a standalone desktop application designed to convert different formats of ASCII music tablatures to a popular music description file format called MusicXML. This software can convert different types of tablature (guitar, drums, and bass) into platform and instrument independent MusicXML.

TAB2XML provides an easy-to-use user interface compatible with most operating systems. Both file processing and direct text input methods are available to users. Translated MusicXML files are accessible through both the interface and the file output.

2.1 Conventions

The term ‘user’ is used throughout this document to refer to a person who requires and/or has acquired access to the TAB2XML.

The term ‘action’ is used throughout this document to refer to a mouse click on a menu or button, and typing in the text area while interacting with the graphic user interface of TAB2XML.

2.2 Cautions & Warnings

The current release of TAB2XML is only intended for users with access granted by the product owner. This application may not be distributed or referenced without the team’s consent.

3. Design Test Cases

3.1 Note Class Test Cases

Note class is a basic data structure in TAB2XML that is used to represent the fundamental element Note in a tablature. Some important elements in a Note object include attributes such as octave, step, and methods such as getter and setters.

noteTestVoice

test Note constructor and getVoice()

noteTestOctave

testNote constructor and getOctave()

noteTestStep

Test Note constructor and getStep()

noteTestDuration

Test setDuration(int duration)

noteTestVoiceSetter

Test setVoice(int voice)

testSetOctave

Test setOctave(int octave)

testSetColumn()

Test setColumn(int column)

testSetStep()

Test setStep(String step)

testSetString()

Test setString(int string)

testSetFret()

Test setFret(int fret)

testSetType()

Test setType(String type)

testAlter()

Test setAlter(int alter)

testConstructorAlter()

Test constructor Note(String step, int alter)

testConstructorAlterOctave()

Test constructor Note(String step, int alter, int octave)

3.2 DrumNote Class Test Cases

DrumNote class is a basic data structure in TAB2XML that is used to represent the fundamental element Note in a drum tablature. DrumNote extends Note class. Some important elements in a DrumNote object include attributes such as octave, step, and methods such as getter and setters.

noteTestVoice()

Test getVoice() working as expected after extending Note class

noteTestOctave()

Test getOctave() working as expected after extending Note class

noteTestStep()

Test getStep() working as expected after extending Note class

noteTestDuration()

Test setDuration(int duration) and getDuration() working as expected after extending Note class

noteTestVoiceSetter()

Test setVoice(int voice) and getVoice() working as expected after extending Note class

testSetOctave()

Test setOctave(int octave) working as expected after extending Note class

testSetColumn()

Test setColumn(int column) and getColumn() working as expected after extending Note class

testSetStep()

Test setStep(String step) working as expected after extending Note class

testSetString()

Test setString(int string) working as expected after extending Note class

testSetFret()

Test setFret(int fret) working as expected after extending Note class

testSetType()

Test setType(String type) working as expected after extending Note class

testSecondConstructor()

Test DrumNote(String step, int octave, String instrumentId, String instrumentName)

testSetInstrument()

Test setInstrument(String instrumentId)

testSetNotehead()

Test setNoteHead(String notehead)

testSetInstrumentName()

Test setInstrumentName(String instrumentName)

3.3 Measure Class Test Cases

Measure class is a basic data structure in TAB2XML that is used to represent the fundamental element Measure in a tablature. Measure class contains Note objects. Some important elements in a Measure object include attributes such as measureList, division, and methods such as getter and setters.

testGetMeasureNumber()

Test getMeasureNumber(), gettimeBeatType(), and getTimeBeats()

testNoteList()

Test getNoteList()

testGetClefLine()

Test getClefLine()

testGetDivisions()

Test constructor Measure(int division) and getDivision()

3.4 NoteUtility Test Cases

NoteUtility class is a utility class in TAB2XML that is used to assist with DrumNote and Note parsing. NoteUtility class contains important attributes such as guitarNote and drumNote, and it contains methods such as makeArray and initialise.

testDrumInitialize()

Test initializeDrum() method properly sets up the DrumNote HashMap

testGuitarInitialize()

Test initialise() method properly sets up the guitarNote array based on either default or user tuning.

testNoteType()

Test if getNoteType(float type) method properly retrieve the noteType based on a float input

testCounterGetterSetter()

Test if counter and octave getters and setters work properly

3.5 BassNoteUtility Test Cases

BassNoteUtility class is a utility class in TAB2XML that is used to assist with Bass tablature parsing. NoteUtility class contains important attributes such as bassNote, and it contains methods such as makeArray and initialise.

testInitialize()

Test if initialise() method properly sets up the bassNote 2D array based on either default or user tuning.

testCounterGetterSetter()

Test if counter and octave getters and setters work properly

3.6 StringParserUtility Test Cases

StringParserUtility is a utility class in TAB2XML that is used to parse input guitar tablature. StringParserUtility includes important static attributes such as stringParse, and it contains methods such as generate, getDivision, and getNote.

testSimple()

Test if the StringParserUtility class properly parses a simple guitar tablature

resetOctaveTest()

Test if the StringParserUtility class properly applies user tuning

testComplex()

Test if the StringParserUtility class properly translates grace notes

3.7 StringParserUtilityDrum Test Cases

StringParserUtilityDrum is a utility class in TAB2XML that is used to parse input drum tablature. StringParserUtilityDrum extends StringParserUtility. It includes important static attributes such as measureList, and it contains methods such as stringParse, getDivision, and getNote.

testDrumSimple()

Test if StringParserUtilityDrum class properly parses a simple drum tablature

3.8 StringParserUtilityBass Test Cases

StringParserUtilityBass is a utility class in TAB2XML that is used to parse input drum tablature. StringParserUtilityBass extends StringParserUtility. It includes important static attributes such as measureList, and it contains methods such as stringParse, getDivision, and getNote.

3.9 XmlGenerator Test Cases

XmlGenerator is a utility class in TAB2XML that is used to generate MusicXML output based on a list of measures. It includes important static methods such as generate.

testException()

Test if an exception is thrown when the input measure list is empty

testSimpleTranslation()

Test if XmlGenerator class properly translate a simple measure list for guitar

testSimpleDrum()

Test if XmlGenerator class properly translate a simple measure list for drum

testSimpleBass()

Test if XmlGenerator class properly translate a simple measure list for bass

testComposerTitle()

Test if XmlGenerator class properly includes user-defined title and composer

4. User Interface Test Cases

4.1 Application Test Cases

ApplicationTest class uses testFX to test the behaviours of the Primary Stage.

should_contain_translate_button(FxRobot robot)

Check if translate button is on the screen

should_contain_save_button(FxRobot robot)

Check if the save button is on the screen

bothButtonDisabled(FxRobot robot)

Check if both translate and save buttons are disabled when the input text area is empty

bothButtonEnabled(FxRobot robot)

Check if both translate and save buttons are enabled when the input text area is not empty

restartButtonDisabled(FxRobot robot)

Check if the restart button is disabled upon launching the application

fileButtonEnabled(FxRobot robot)

Check if the file button is enabled upon launching the application

4.2 Controller Test Cases

ControllerTest class tests static methods of the controller.

testCleanUp()

Check if the cleanup method ignores invalid lines of the tablature

testRepeatCleanup()

Check if the repeatCleanUp method ignores REPEAT in a tablature

4.3 Translation Options Test Cases

TranslationOptionsTest class uses testFX to test the behaviours of the Translation Options window.

should_contain_confirm(FxRobot robot)

Check if the translation options screen has the confirm button

should_contain_confirm(FxRobot robot)

Check if the translation options screen has the cancel button

4.4 Tuning Options Test Cases

TuningOptionTest class uses testFX to test the behaviours of the Tuning Options window.

has_confirm_button(FxRobot robot)

Check if the tuning options screen has the confirm button

has_cancel_button(FxRobot robot)

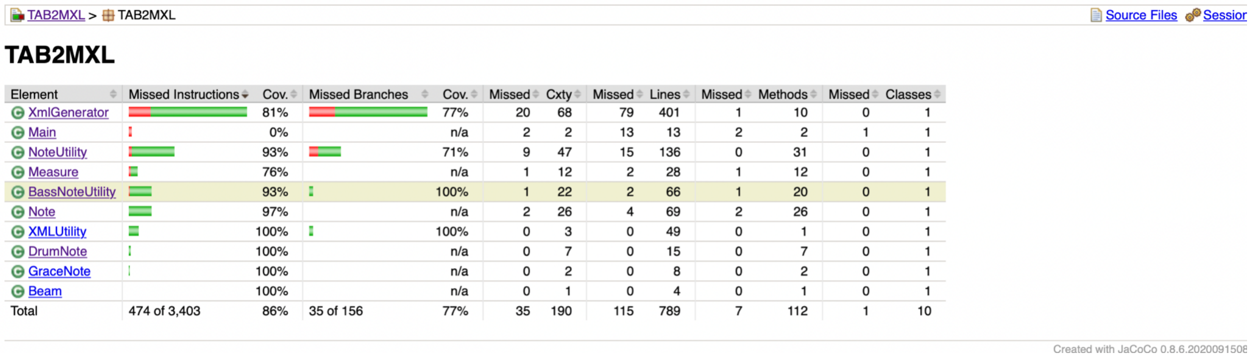
Check if the tuning options screen has the cancel button

5. Testing Coverage

5.1 Coverage

Coverage for Design

Figure 1 - coverage of TAB2XML design



TAB2MXL > TAB2MXL   Source Files Session

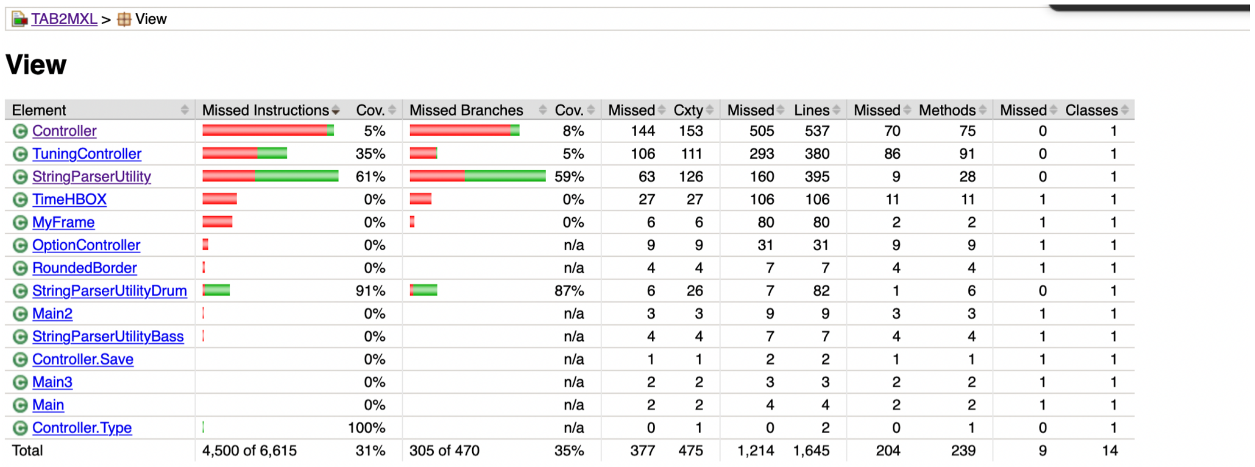
TAB2MXL

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
XmlGenerator		81%		77%	20	68	79	401	1	10	0	1
Main		0%		n/a	2	2	13	13	2	2	1	1
NoteUtility		93%		71%	9	47	15	136	0	31	0	1
Measure		76%		n/a	1	12	2	28	1	12	0	1
BassNoteUtility		93%		100%	1	22	2	66	1	20	0	1
Note		97%		n/a	2	26	4	69	2	26	0	1
XMLUtility		100%		100%	0	3	0	49	0	1	0	1
DrumNote		100%		n/a	0	7	0	15	0	7	0	1
GraceNote		100%		n/a	0	2	0	8	0	2	0	1
Beam		100%		n/a	0	1	0	4	0	1	0	1
Total	474 of 3,403	86%	35 of 156	77%	35	190	115	789	7	112	1	10

Created with JaCoCo 0.8.6.2020091508

Coverage for User Interface

Figure 2 - coverage of TAB2XML interface



TAB2MXL > View

View

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Controller		5%		8%	144	153	505	537	70	75	0	1
TuningController		35%		5%	106	111	293	380	86	91	0	1
StringParserUtility		61%		59%	63	126	160	395	9	28	0	1
TimeHBOX		0%		0%	27	27	106	106	11	11	1	1
MyFrame		0%		0%	6	6	80	80	2	2	1	1
OptionController		0%		n/a	9	9	31	31	9	9	1	1
RoundedBorder		0%		n/a	4	4	7	7	4	4	1	1
StringParserUtilityDrum		91%		87%	6	26	7	82	1	6	0	1
Main2		0%		n/a	3	3	9	9	3	3	1	1
StringParserUtilityBass		0%		n/a	4	4	7	7	4	4	1	1
Controller.Save		0%		n/a	1	1	2	2	1	1	1	1
Main3		0%		n/a	2	2	3	3	2	2	1	1
Main		0%		n/a	2	2	4	4	2	2	1	1
Controller.Type		100%		n/a	0	1	0	2	0	1	0	1
Total	4,500 of 6,615	31%	305 of 470	35%	377	475	1,214	1,645	204	239	9	14

5.2 Future Improvement

As shown in Figure 1, the statement coverage for design has achieved more than 80 percent. The lack of coverage is mainly due to unused classes and methods. Statement coverage for the user interface is only 31%. The team did not receive clear direction for unit testing using TestFX and many functionalities failed to yield expected results. Many classes such as MyFrame and RoundedBorder are also legacy code for our previous JavaSwing GUI and not covered by our test cases. For future improvement, we hope to improve code coverage for the interface by familiarizing the QA team with TestFX.