

**TAB2XML**

Testing Document

**Group 6**

**Elmira Onagh**

**Irsa Nasir**

**Long Lin**

**Harjap Randhawa**

**Daniel Di Giovanni**



Winter 2022

**Table of Content**

[**1.** **DrawBar.java class** 2](#_Toc97430275)

[**2.** **Guitar.java Class** 3](#_Toc97430276)

[**3.** **DrawClef.java Class** 4](#_Toc97430277)

[**4.** **DrawMusicLines.java Class** 5](#_Toc97430278)

[**5.** **DrawNoteType.java class** 6](#_Toc97430279)

[**6.** **MLine.java Class** 7](#_Toc97430280)

[**7.** **DrawNoteTest.java Class** 8](#_Toc97430281)

# **DrawBar.java class**

This class allows drawing a vertical bar at specific x and y coordinates on a given Pane. The test methods examine that the double value obtained by getStartX () and getStartY () methods are consistent with the expected values and setStartX () and setStartY () set their corresponding variable to the expected value. Moreover, the test check to make sure that getPane () method returns the expected pane and setPane () adds the correct pane to the class.

* **Test: testGetPane ()**

Tests the getPane () method of the DrawBar.java class to make sure it returns the expected pane.

* **Test: testSetPane ()**

Test the setPane () method to make sure the method returns the correct result of the pane.

* **Test: testGetStartX ()**

Test the getStartX () method to ensure the correct x coordinate is being returned.

* **Test: testSetStartX ()**

Tests setStartX () method to make sure the value for the x- coordinate is set correctly

* **Test: testGetStartY ()**

Test the getStartY () method to ensure the correct y coordinate is being returned.

* **Test: testSetStartY ()**

Tests setStartY () method to make sure the value for the y- coordinate is set correctly

These test cases examine the public classes of DrawBar.java class and ensure they behave as expected. The only public method that is not tested at the moment is the draw() method that adds the created bar object to the Pane (GUI testing will be provided in the future releases).

By testing setter and getter methods we can be sure that a DrawBar object is initialized correctly with the expected values and will behave (get drawn on Pane) as expected.

# **Guitar.java Class**

This class allows creating and drawing a Guitar object based on a given ScorePartwise object. There are two ways to initialize a Guitar object. First by entering ScorePartwise and Pane values directly during initialization or setting the values later by using the setter methods. There are ten public methods: drawGuitar (), extractClef (), noteHasChord (), noteHasTechnical(), highlightMeasureArea(), getMeasureList(), getXCoordinatesForGivenMeasure(), getYCoordinatesForGivenMeasure(), setMeasureList(), and playGuitarNote().

The following methods are tested under TestGuitar.java:

* **Test: testExtractClef ()**

Tests the extractCleft (Measure) method to make sure the method returns the correct Clef object of the given measure.

* **Test: testNoteHasChordTrue ()**

Test the noteHasChord (Note) method to make sure the method returns true if a given note has a chord attribute attached to it.

* **Test: testNoteHasChordFalse ()**

Test the noteHasChord (Note) method to make sure the method returns false if a given note does not have a chord attribute attached to it.

* **Test: testNoteHasTechnicalTrue ()**

Test the noteHasTechnical (Note) method to make sure the method returns true if a given note has a technical attribute (For guitar note).

* **Test: testNoteHasTechnicalFalse ()**

Test the noteHasTechnical (Note) method to make sure the method returns false if a given note does not have a technical attribute (For guitar note).

* **Test: testGetMeasureList ()**

Test getMeasureList () to make sure the method returns the correct list of measures.

* **Test: testSetMeasureList ()**

Test setMeasureList () to make sure given a List<Measure> the method sets the variable measureList correctly.

Technically, all these class can be set to private (in fact, that would be a better practice), however, since testing drawGuitar () -which add elements to GUI- is not possible at the moment, we can test these methods to make sure we are using the correct values in the drawGuitar () method. Therefore, by inference the method is displaying the correct element to the user. In future releases addition of GUI testing will make these tests unnecessary and they can be set to private to ensure abstraction principle.

# **DrawClef.java Class**

This class allows drawing a Clef object on the screen. The class has nine public class which are tested using DrawClefTest.java class. The test methods check if the getX (), setX (), getY (), setY(), getPane (), setPane(), getClef (), and setClef() methods in the DrawClef class are behaving as expected:

* **Test: testGetClef ()**

Ensures that getClef () method returns the expected Clef object.

* **Test: testSetClef ()**

Ensures the setClef () method set the correct value for the Clef object.

* **Test: testGetPane ()**

Ensures that the getPane () method returns the correct Pane.

* **Test: testSetPane ()**

Ensure the setPane () method sets the correct value for the Pane.

* **Test: testGetX ()**

Ensure the getX () method returns the expected x coordinate.

* **Test: testSetX ()**

Ensures setX () method sets the correct value for the x coordinate.

* **Test: testGetY ()**

Ensure the getY () method returns the expected y coordinate.

* **Test: testSetY ()**

Ensures setY () method sets the correct value for the x coordinate.

These tests examine and ensure that the DrawClef object is initialized properly with correct values and therefore will behave as expected when these values are used in the draw () method to add the clef to the GUI.

# **DrawMusicLines.java Class**

This class is used to create a group of 6 horizonal lines of fixed width to be used for music score and displaying them on GUI. This class has four public methods of which following are tested by using DrawMusicLinesTest.java:

* **Test: testGetMusicLineList ()**

Ensures that getMusicLineList() return the correct list of line (each group of six lines are added to a list). Since we are creating a new MLine object to be used as each individual line, the expected and actual list used in the test case cannot be the same. Instead we used other methods to ensure they are creating the same values such testing the size of the list and startX, startY, endX, endY for every individual line to make sure they refer to the same value.

* **Test: testGetPane ()**

Ensures getPane () method returns the correct pane.

* **Test: testSetPane ()**

Ensure the setPane () method set the value of the pane correctly.

These tests Ensure that the created DrawMusicLines object is set with the expected values and therefore when used in draw () method they behaviour should be as expected, and music lines should be displayed at the specified positions.

# **DrawNoteType.java class**

This class creates and displays the type of individual notes for guitar tablature (half note, quarter note, etc.). The class has seven public method, and the drawing of the note types are done through drawDuration () method. Instead of testing this method directly, we test the other methods to make sure the DrawNoteType object is initialized using the correct value and by inference the drawDuration () method is behaving as expected and displays the note type at the correct positions. The test cases are as follows:

* **Test: testGetPane ()**

Ensures getPane () method returns the correct pane.

* **Test: testSetPane ()**

Ensure setPane () method set the value for the pane correctly.

* **Test: testGetStartX ()**

Ensure that getStartX () returns correct value for the x coordinates.

* **Test: testSetStartX ()**

Ensure that the setStartX () method sets the value for x coordinate correctly.

* **Test: testGetStartY ()**

Ensure that getStartY () returns correct value for the y coordinates.

* **Test: testSetStartY ()**
* Ensure that the setStartY () method sets the value for y coordinate correctly.

# **MLine.java Class**

This class create and a single horizontal line with addition of adding a tag value (integer) from 1 – 6, corresponding to 6 lines in music score. The class has six public class and testing them ensures that created MLine object contains correct value and therefore the line is added to the correct position on the screen. The test cases are as follows:

* **Test: testGetPane ()**

Ensures getPane () method returns the correct pane.

* **Test: testSetPane ()**

Ensure setPane () method set the value for the pane correctly.

* **Test: testGetStartX ()**

Given a tag value (integer) ensure that getStartX () returns correct value for the x coordinates associated with tag.

* **Test: testGetStartY ()**

Given a tag value (integer) ensure that getStartY () returns correct value for the y coordinates associated with tag.

* **Test: testGetLineTag ()**

Ensure that getLineTag () method returns the correct value for the tag.

* **Test: testSetLineTag ()**

Ensure that the setLineTag () set the value of the line tag properly.

# **DrawNoteTest.java Class**

* **Test: testDrawFret ()**

Test testDrawFret () parses a guitar tablature and draws the notes to a pane, checking their x-position, y-position, and text value to make sure they are correct.

* **Test: testDrawO ()**

Test testDrawO () parses a drum set tablature and draws the notes that are not cymbal notes to a pane. Each note's x-position and y-position are checked, and the text value is checked to make sure they are all "o".

* **Test: testDrawX ()**

Test testDrawX () parses a drum set tablature and draws the notes that are cymbal notes to a pane. Each note's x-position and y-position are checked, and the text value is checked to make sure they are all "x".

**Group 6**

**Elmira Onagh**

**Irsa Nasir**

**Long Lin**

**Harjap Randhawa**

**Daniel Di Giovanni**