TAB2XML

Testing Document

Group 6

Elmira Onagh
Irsa Nasir
Long Lin
Harjap Randhawa
Daniel Di Giovanni
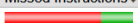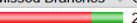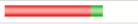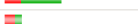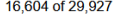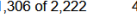
Winter 2022

Table of Content
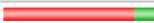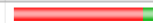
1. Test Coverage

For test coverage, we used Jococo test coverage for the three package: Gui, Gui.draw, and instrument. By adding the

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
instruments		24%		24%	377	473	1,096	1,419	108	145	2	4
GUI		12%		16%	348	385	1,095	1,238	136	148	19	20
GUI.draw		41%		52%	174	309	747	1,191	137	241	6	18
converter.note		70%		62%	121	299	159	687	26	125	1	14
converter.instruction		36%		27%	78	101	139	222	16	35	1	6
converter		78%		71%	63	154	80	432	11	50	1	7
converter.measure		86%		71%	53	145	55	390	5	42	0	4
utility		88%		78%	37	116	67	342	22	68	2	12
models.measure.note.notations.technical		46%		50%	31	57	56	106	29	53	2	9
models.measure.note		59%		50%	28	73	54	126	27	71	1	11
converter.measure_line		69%		50%	15	31	23	73	2	13	0	4
models		43%		n/a	22	33	35	62	22	33	1	5
models.measure.attributes		52%		n/a	21	36	31	66	21	36	0	6
models.part_list		43%		n/a	18	24	28	51	18	24	1	5
models.measure.barline		48%		n/a	11	21	15	30	11	21	1	3
models.measure.note.notations		80%		87%	10	36	14	67	9	32	0	4
models.measure		53%		n/a	8	18	13	28	8	18	1	2
models.measure.direction		49%		n/a	8	15	10	21	8	15	0	3
custom_exceptions		0%		n/a	4	4	8	8	4	4	4	4
Total	16,604 of 29,927	44%	1,306 of 2,222	41%	1,427	2,330	3,725	6,559	620	1,174	43	141

instruments

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Guitar		13%		15%	194	233	612	714	50	71	0	1
Bass		0%		0%	118	118	367	367	43	43	1	1
Drumset		70%		60%	64	121	116	337	14	30	0	1
JfugueTest		0%		n/a	1	1	1	1	1	1	1	1
Total	5,326 of 7,065	24%	495 of 655	24%	377	473	1,096	1,419	108	145	2	4

Gui: MusicPlayer

PreviewMusic.new ChangeListener() {...}		0%		0%
MusicPlayer		75%		63%
SaveMXLController		0%		0%
PreviewMusic.new ChangeListener() {...}		0%		0%

GUI.draw

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
DrawBassNote		0%		0%	23	23	121	121	21	21	1	1
DrawNote		23%		25%	17	25	89	125	15	23	0	1
DrawDrumsetNote		75%		72%	18	49	59	217	6	18	0	1
DrawNoteType		12%		n/a	12	19	77	95	12	19	0	1
DrawBassRepeat		0%		n/a	11	11	75	75	11	11	1	1
DrawBassNoteType		0%		0%	20	20	75	75	15	15	1	1
DrawBend		0%		0%	16	16	57	57	14	14	1	1
DrawBassBend		0%		0%	6	6	38	38	4	4	1	1
DrawBassClef		28%		0%	7	16	28	46	5	14	0	1
DrawBar		26%		n/a	3	10	23	38	3	10	0	1
DrawSlur		35%		0%	3	17	25	47	2	16	0	1
DrawBassLines		0%		0%	8	8	23	23	7	7	1	1
DrawClef		72%		0%	9	23	20	68	7	21	0	1
DrawDrumsetMusicLines		79%		73%	14	30	20	76	8	13	0	1
DrawBassBar		50%		n/a	2	9	9	23	2	9	0	1
DrawMusicLines		87%		100%	3	9	5	21	3	8	0	1
MLine		95%		n/a	1	9	2	25	1	9	0	1
DrawDrumsetBar		95%		n/a	1	9	1	21	1	9	0	1
Total	3,282 of 5,576	41%	60 of 125	52%	174	309	747	1,191	137	241	6	18

By looking at these coverages we can see that we do have a good coverage however for the future version it would be great addition to add extra test to ensure the performance of our system.

2. 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 checks to make sure that the 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 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.

3. 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 (), etc.

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.

❖ **Test: testCountNote()**
test the countNote method which returns number of regular notes (not chord or grace) in a given measure. The test check that the actual count is same as expected one.

❖ **Test: testCountNoteSpacesRequired()**
Tests countNoteSpacesRequired() method which calculates amount of space required for each measure by using countNote() and spacing value.

❖ **Test: testNoteHasSlide()**
Check the noteHasSlide() method which return true if a given note has slide tag and false if it does not. For a note with slide the expected value is true and the test case checks if the actual return value is also true.

❖ **Test: testNoteHasHarmonic()**
Tests noteHasHarmnic() method. For a note with harmonic tag, the expected result is true and for a note without the harmonic tag the expected result is false.

❖ **Test: testNoteHasTie()**
Tests noteHasTie() method. For a note with tie tag, the expected result is true and for a note without the tie tag the expected result is false.

❖ **Test: testNoteHasBend()**
Tests noteHasBend() method. For a note with bend tag, the expected result is true and for a note without the bend tag the expected result is false.

❖ **Test: testNoteHasGrace()**
Tests noteHasGrace() method. For a note with grace tag, the expected result is true and for a note without the grace tag the expected result is false.

❖ **Test: testCountGraceSpace()**
Tests countGraceSpace() method which returns number of space required for grace (how many grace notes are next to each other).

❖ **Test: testNextHasActual()**
Test nextHastActual () method to see if the next note (only regular notes, not chords/grace notes) is also actual note. If next note is also actual the expected value is true.

❖ **Test: testGetActualNum()**
Test getActualNum () method which given a note return number consecutive actual notes including the current note. The expected value is an integer.

Technically, all these classes can be set to private (in fact, that would be a better practice), however, since testing drawGuitar () -which add elements to GUI- and similar methods 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, more testing has to be done to ensure that the performance of the system.

4. DrumTest.java Class

This class allows creating and drawing drumset sheet music based on a given ScorePartwise object. The tests are conducted by parsing a music tablature and using it to initialize the Drumset class.

The following methods are tested under DrumTest.java:

- ❖ **Test: testPositioningValuesDefault ()**
  Tests the values of the spacing, font size, music line spacing, and staff spacing given default parameters.
- ❖ **Test: testPositioningModified ()**
  Tests the values of the spacing, font size, music line spacing, and staff spacing given modified parameters (simulating user changing the appearance of the sheet music).
  Test: testScorePartwise ()
  Test to make sure the score partwise object was initialized and set properly in the Drumset object.
- ❖ **Test: testPane ()**
  Test to make sure the pane object was initialized and set properly in the Drumset object.
- ❖ **Test: testMeasureList ()**
  Test to make sure the list of measures was extracted from the score partwise and set properly in the Drumset object.
- ❖ **Test: testClef ()**
  Test to make sure the clef was extracted from the score partwise and set properly in the Drumset object.

These tests make sure the Drumset class is ready to draw the music. They make sure the integral parts are set up properly for drawing to follow (score partwise, measure list, pane, etc.). The tests involving the actual drawing of drumset music are found in DrawDrumsetNoteTest.java.

5. DrawClef.java Class

This class allows drawing a Clef object on the screen. The class has nine public classes 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 the getClef () method returns the expected Clef object.
- ❖ **Test: testSetClef ()**
  Ensures the setClef () method sets 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.

6. DrawMusicLines.java Class

This class is used to create a group of 6 horizontal lines of fixed width to be used for music scores 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 () returns the correct list of lines (each group of six lines is 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 as 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 sets 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 the draw () method its behaviour should be as expected, and music lines should be displayed at the specified positions.

7. 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 methods, and the drawing of the note types is done through the 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 sets 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.

8. MLine.java Class

This class creates and displays a single horizontal line with the addition of a tag value (integer) from 1 – 6, corresponding to 6 lines in a music score. The class has six public classes and testing them ensures that created MLine object contains correct values 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 sets 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 the correct value for the y coordinates associated with the tag.

❖ **Test: testGetLineTag ()**
Ensure that the getLineTag () method returns the correct value for the tag.

❖ **Test: testSetLineTag ()**
Ensure that the setLineTag () sets the value of the line tag properly.

9. DrawNoteTest.java Class

This class is used to test the drawing of notes of guitar and drums to a pane. The drawFret () method does the drawing for individual notes for guitar. The tests make sure that drawFret () is in fact drawing the notes at the correct positions, with the correct text values.

- ❖ **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.

This is sufficient testing because when the drawing methods are changed, they should still draw the notes at the same positions for the simple tablature used in the test case. This will ensure the fundamental part of the drawing does not contain errors.

10.   DrawDrumsetNoteTest.java Class

This class is used to test the drawing of drumset notes. This is not done manually, instead it is done by parsing a tablature and using the drumset drawer to draw the notes to a pane. The pane is then checked for errors.

- ❖ **Test: testDrawO()**
  Test testDrawO () parses a drum set tablature and draws the notes to a pane. The number of non-cymbal notes is counted to ensure it is the correct value.

- ❖ **Test: testDrawX()**
  Test testDrawX () parses a drum set tablature and draws the to a pane. The number of cymbal notes are counted to ensure it is the correct value.

- ❖ **Test: testDrawRest()**
  Test testDrawRest () parses a drum set tablature and draws the to a pane. The tablature used contains a rest, which is rendered as an SVGPath object. Once the tab is drawn, the pane is checked for this SVGPath object.

- ❖ **Test: testDrawBeam()**
  Test testDrawBeam () parses a drum set tablature and draws the to a pane. The tablature used contains a number of 32nd notes, 16th notes, and eight notes. These notes are beamed together, and the number of beams are counted to ensure there is the current amount.

- ❖ **Test: testDrawTie()**
  Test testDrawTie () parses a drum set tablature and draws the to a pane. The tablature used contains a number of tied notes. The number of ties in the pane is counted to ensure it is the correct amount.

This is sufficient testing because when the drawing methods are changed, they should still draw the same number of notes. Also, as the positioning and sizes of the notes are changed, the number of notes should still remain unchanged. This will ensure the fundamental part of the drawing does not contain errors.

11.   DrawBassBarTest.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 checks to make sure that the getPane () method returns the expected pane and setPane () adds the correct pane to the class.

❖ **Test: testGetPane ()**
Tests the getPane () method of the DrawBassBar.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 DrawBassBar.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.

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

12.   DrawBassClefTest.java Class

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

- ❖ **Test: testGetClef ()**
  Ensures that the getClef () method returns the expected Clef object.
- ❖ **Test: testSetClef ()**
  Ensures the setClef () method sets 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 DrawBassClef 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.

13.   MusicPlayerTest.java Class

This class allows creating music sequences based on input tablature. There are 3 main methods to get music sequence: getGuitarString(), getDrumString(), getBassString(). Our goal in the test cases is to make sure the object gets the correct music sequence. The test cases are as follows:

❖ **Test: testGuitar ()**
   Given a input guitar tablature, ensures getGuitarString() method returns the correct string.
❖ **Test: testDrum ()**
   Given a input drum tablature, ensures getDrumString() method returns the correct string.
❖ **Test: testBass ()**
   Given a input bass tablature, ensures getBassString() method returns the correct string.