Testing Document

Tab2XML
Group 7

Joshua Genat
Andy Lin
Uthithmenon Ravitharan
Nicolae Semionov

Table of Contents

1.0 Introduction

1.1 Purpose

The purpose of this document is to describe all JUnit test cases, the classes they are designed to test, how they were derived and why they are useful.

2.0 Guitar Converter

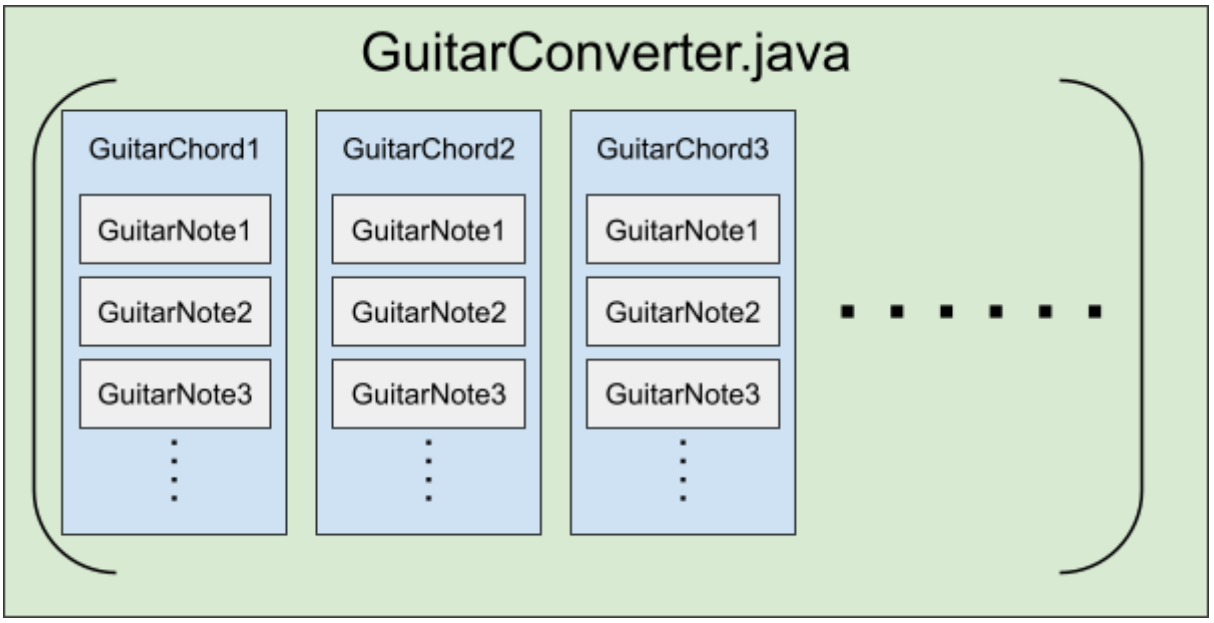
2.1 Class description

The main function of this class is to accept a 2d character array parsed from a tab, and to convert into a form that is more accessible to the xml converters.

Other classes used by GuitarConverter.java

- GuitarChord.java: This class is used to create objects of guitar chords. GuitarChords store an array of GuitarNotes, and include methods to manipulate multiple notes in an organized fashion
- GuitarNotes: Objects used by the program's xml converters. Their main purpose it to have a convenient way of accessing note properties at a particular position in a tab such as;
    - String note
    - String type
    - int alter
    - int octave
    - int string
    - int fret
    - int duration
    - int voice
    - char step
    - boolean isChord

The class returns an array of GuitarChord objects. A GuitarChord is simply a convenient way of organizing arrays of GuitarNotes with extra functionality.



2.2 JUnit Test

The JUnit test GuitarConverterTester.java aims to ensure all note properties are getting assigned proper values. The tester passes a sample input provided in the course wiki as a 2d array of characters.

```
char[][] test = {
        {'|', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '0', '-', '-', '-', '-', '-', '-', '|',
        {'|', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '1', '-', '-', '-', '0', '-', '-', '-', '|',
        {'|', '-', '-', '-', '-', '-', '-', '-', '1', '-', '-', '-', '-', '-', '-', '-', '-', '1', '-', '-', '|',
        {'|', '-', '-', '-', '-', '-', '2', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '|',
        {'|', '-', '-', '-', '2', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '|',
        {'|', '-', '0', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '|',
};
```

The resulting output of GuitarConverter.converter(test) is saved in an array of GuitarChords named ‘result’.

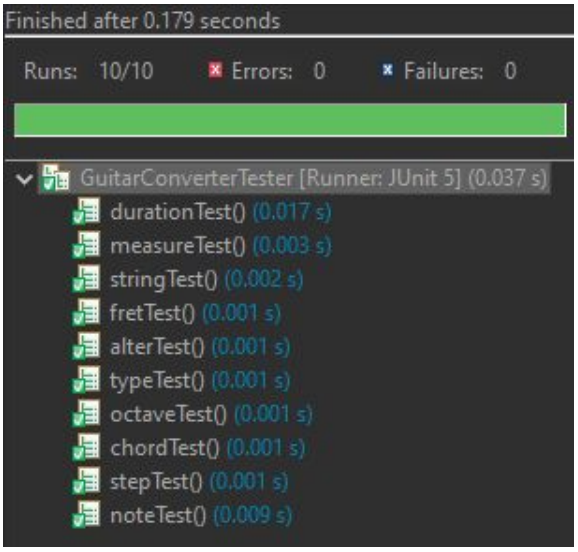
The result is then compared using a randomly generated index with an array of expected outputs. For example, here is a 2d array of expected step chars.

```
char[][] expectedStep = {
    {'-', '-', '-', '-', '-', 'E', '-', '-', 'E', '-'},
    {'-', '-', '-', '-', 'C', '-', 'B', '-', 'B', '-'},
    {'-', '-', '-', 'G', '-', '-', '-', 'G', 'G', '-'},
    {'-', '-', 'E', '-', '-', '-', '-', '-', 'E', '-'},
    {'-', 'B', '-', '-', '-', '-', '-', '-', 'B', '-'},
    {'E', '-', '-', '-', '-', '-', '-', '-', 'E', '-'}
};
```

These are the step values each note is expected to have at that index.

This 2d char step array is used to compare with the steps in GuitarChords array ‘result’ in the following lines of code

```
@Test
void stepTest() {
	assertEquals(result[i[0]].notes[i[1]].step,expectedStep[i[1]][i[0]]);
}
```



This process is repeated for all note properties

These tests are sufficient since the array index is randomly generated. A change that results in the incorrect assignment of a note property will likely be spotted by the tester on the first run through.

3.0 Drum Tester

3.1 Class description

The main function of this class is to accept a 2d character array and return the row and col values of the points at which a percussion instrument is played in the two parallel Arraylist: RowArrayListTest() and ColArrayListTest(). In a percussion MusicXML file, voice 1 notes should be listed first, followed by voice 2 notes, the following test fulfills this requirement (voice number depends on the type of percussion instrument being played. This tester class is very important for the conversion of tabulate to Music XML, because knowing the correct order at which notes are played, is instrumental for creating accurate sheet music. The following two tests also allow us to get other unknown variables which are important for the conversion such as how long a note should be played (duration), Display Octave, and Display Steps.

Test 1 - RowArrayListTest()

This test takes in a 2d character array, and returns an Arraylist of the integers. These integers are the row values of when the percussion notes are played. For example if a note is played at row 4, column 1, the integer 4 would be in the Arraylist.

Input:

```
char[][] test1 =
{
        {'C', ' ', '|', '-', '-', 'x', '-', '-', '-', '|', ' ', '-', '-', '-', 'x', '-', '-', '|'},
        {'R', ' ', '|', '-', 'o', 'x', '-', '-', '-', '|', 'x', '-', '-', '-', '-', '-', 'x', '|'},
        {'F', 'T', '|', '-', '-', '-', 'x', '-', 'x', '|', ' ', '-', '-', '-', '-', '-', '-', '|'},
        {'H', 'f', '|', 'o', '-', '-', '-', '-', 'x', '|', '-', '-', '-', 'o', 'x', '-', '-', '|'},
        {'S', 'D', '|', '-', '-', '-', '-', 'x', '-', '|', 'x', '-', '-', '-', '-', 'x', '-', '|'},
        {'B', ' ', '|', '-', 'x', '-', '-', '-', 'o', '|', 'o', '-', 'o', '-', '-', '-', '-', '|'}
};
```

Expected ArrayList Output:

```
Integer[] expectedValues = {3, 1, 1, 0, 2, 4, 3, 2, 5, 5, 4, 1, 3, 3, 0, 4, 1, 5, 5};
 exp = new ArrayList<>(Arrays.asList(expectedValues));
 assertEquals(exp, act);
```

Sufficiency: This test is sufficient as the output combined with the ColArrayListTest() list the correct order at which the drum notes should be listed in a MusicXML file. Which is a great starting point for converting a drum tabulate to MusicXML.

Test 2 - ColArrayListTest()

This test takes in a 2d character array, and returns an Arraylist of the integers. These integers are the column values of when the percussion notes are played and they are listed in the correct order for an MusicXML file. For example if notes are played at row 4, column 1, the integer 1 would be in the Arraylist.

Input:

```
char[][] test1 =
{
        {'C', ' ', '|', '-', '-', 'x', '-', '-', '-','|', ' ', '-', '-', '-', 'x', '-', '-', '|'},
        {'R', ' ', '|', '-', 'o', 'x', '-', '-', '-','|', 'x', '-', '-', '-', '-', '-', 'x', '|'},
        {'F', 'T', '|', '-', '-', '-', 'x', '-', 'x','|', ' ', '-', '-', '-', '-', '-', '-', '|'},
        {'H', 'f', '|', 'o', '-', '-', '-', '-', 'x','|', '-', '-', '-', 'o', 'x', '-', '-', '|'},
        {'S', 'D', '|', '-', '-', '-', '-', 'x', '-','|', 'x', '-', '-', '-', '-', 'x', '-', '|'},
        {'B', ' ', '|', '-', 'x', '-', '-', '-', 'o','|', 'o', '-', 'o', '-', '-', '-', '-', '|'}
};
```

Expected Arraylist Output:

```
}

Integer[] expectedValues = {3, 4, 5, 5, 6, 7, 8, 8, 4, 8, 10, 10, 13, 14, 14, 15, 16, 10, 12};
 exp = new ArrayList<>(Arrays.asList(expectedValues));
 assertEquals(exp, act);
```

Sufficiency: This test is sufficient as the output combined with the RowArrayListTest() list the correct order at which the drum notes should be listed in a MusicXML file. Which is a great starting point for converting a drum tabulate to MusicXML.

4.0 Parser Tester

4.1 Class description

The main function of these classes is to take in a file or text and extract the necessary information.

Test 1 - ParserRemoveTrashTest()

This test run checks to see if the parser removes excess strings such as lyrics and symbol definers and gets rid of it. This will allow us to extract only the tablature parts of the file and make it easier for conversion.

Implementation: Uploaded a sample file and ran it through the parser, I took the ArrayList<String> that the parser makes and I compared it to a ArrayList<String> that I manually made that just had the tablature and assert equal.

Input:

```
Band:Black Sabbath
Song:War Pigs
Tabber:Jared Myers

This is my fav. Black Sabbath song and i think this is Bill Ward's best performance.  The Fills in this song
are complicated.  The timing is 6/8.  When u can play this song u can play drums well.  This song took me forever to learn.
Enjoy.Please Rate.

 Intro
C |x--------x--------|---------x--x-----|x--------x--------|---------x--x-----|
R |------x-----x--x--|x--x-xx-----------|---x-xx-----x-xx--|x--x-xx-----------|
SD|---------o--------|---------o--o-----|---------o--------|---------o--o-----|
FT|------------------|--------------o-o-|------------------|--------------oo--|
B |o----------------o|o----------o------|o----------------o|o----------oo-----|
  (1t12t13t14t15t16t1|1t12t13t14t15t16t1|1t12t13t14t15t16t1|1t12t13t14t15t16t1)
```

Expected:

```
exp.add("C |x--------x--------|---------x--x-----|x--------x--------|---------x--x-----|");
exp.add("R |------x-----x--x--|x--x-xx-----------|---x-xx-----x-xx--|x--x-xx-----------|");
exp.add("SD|---------o--------|---------o--o-----|---------o--------|---------o--o-----|");
exp.add("FT|------------------|--------------o-o-|------------------|--------------oo--|");
exp.add("B |o----------------o|o----------o------|o----------------o|o----------oo-----|");
```

Sufficiency: This test is Sufficient because the program has shown that it can extract the tablature from numbers, symbols, and strings.

Test 2 - isDrumTypeCorrect()

This test run checks to see if the parser can determine the tablature type and in this case if it can determine that the tablature is a drum type

Implementation: Uploaded a sample file of a drum tablature and ran it through the parser. I took the String type that the parser generates and compare it to “Drum”

Input:

```
Band:Black Sabbath
Song:War Pigs
Tabber:Jared Myers

This is my fav. Black Sabbath song and i think this is Bill Ward's best performance.  The Fills in this song
are complicated.  The timing is 6/8.  When u can play this song u can play drums well.  This song took me forever to learn.
Enjoy.Please Rate.

 Intro
C |x--------x--------|---------x--x-----|x--------x--------|---------x--x-----|
R |------x-----x--x--|x--x-xx-----------|---x-xx-----x-xx--|x--x-xx-----------|
SD|---------o--------|---------o--o-----|---------o--------|---------o--o-----|
FT|------------------|--------------o-o-|------------------|--------------oo--|
B |o-----------------o|o----------o------|o-----------------|o----------oo-----|
  (1t12t13t14t15t16t1|1t12t13t14t15t16t1|1t12t13t14t15t16t1|1t12t13t14t15t16t1)
```

Expected: `assertEquals("Drum", b.Type);`

Sufficiency: This test is sufficient as this shows us that the parser can determine the type of tablature.

Test 3 - isGuitarTypeCorrect()
This test run checks to see if the parser can determine the tablature type and in this case if it can determine that the tablature is a guitar

Implementation: Uploaded a sample file of a guitar tablature and ran it through the parser. I took the String type that the parser generates and compare it to “Guitar”

Input:

```
e|--------------------|----------7-----7--------|-----5-----------3-------|
B|-------7--------5--|----------------8--------|-------7-----------5-----|
G|-------7--------5--|-------------9-----------|---------7-----------5---|
D|----7--------5-----|-------------------------|-------------------------|
A|-5--------3--------|-------7-----------------|-5-----------3-----------|
E|--------------------|-0-----------------------|-------------------------|
```

Expected: `assertEquals("Guitar", b.Type);`

Sufficiency: This test is sufficient as this shows us that the parser can determine the type of tablature.

Test 4 - repeatType1()
This test run checks to see if the parser can interpret the repeat of kind
|----repeatxN---|

Implementation: Uploaded a sample file of a drum tablature that has a repeat of type |----repeatxN---|, and Then took the arraylist<lines> that it generated and compared it to a manually created one where I did the repeats manually.

Input:

```
  1st Verse
   |------------REPEAT-1x------------|
C  |xx--------------|----------------|xx--------------|----------------|
HH|----x-x-x-x-x-x-|x-x-x-x-xox-x-x-|----x-x-x-x-x-x-|----------x-x-x-|
T  |----------------|----------------|----------------|--o-------------|
SD|----------------|----------------|----------------|o----o--f-------|
B  |oo--------------|----------------|oo--------------|-------o-o-oo-o-|
```

Expected:

```
exp.add("C  |xx--------------|----------------|xx--------------|----------------|");
exp.add("HH|----x-x-x-x-x-x-|x-x-x-x-xox-x-x-|----x-x-x-x-x-x-|----------x-x-x-|");
exp.add("T  |----------------|----------------|----------------|--o-------------|");
exp.add("SD|----------------|----------------|----------------|o----o--f-------|");
exp.add("B  |oo--------------|----------------|oo--------------|-------o-o-oo-o-|");
```

Sufficiency: This test is sufficient as this shows us that the parser interpret the repeat and also finds the section that is required to be repeated. And extend it accordingly.

Test 5 - repeatType2()

This test run checks to see if the parser can interpret the repeat of kind xN where N is the number to be repeated

Implementation: Uploaded a sample file of a drum tablature that has a repeat of type xN, and Then took the arraylist<lines> that it generated and compared it to a manually created one where I did the repeats manually. This repeat instead of extending it, actually adds the lines N amount of times.

Input:

```
C  |xx--------------|----------------|xx--------------|----------------|
HH|----x-x-x-x-x-x-|x-x-x-x-xox-x-x-|----x-x-x-x-x-x-|----------x-x-x-|
T  |----------------|----------------|----------------|--o-------------| x2
SD|----------------|----------------|----------------|o----o--f-------|
B  |oo--------------|----------------|oo--------------|-------o-o-oo-o-|
```

Expected:

```
exp.add("C  |xx--------------|----------------|xx--------------|----------------|");
exp.add("HH|----x-x-x-x-x-x-|x-x-x-x-xox-x-x-|----x-x-x-x-x-x-|----------x-x-x-|");
exp.add("T  |----------------|----------------|----------------|--o-------------|");
exp.add("SD|----------------|----------------|----------------|o----o--f-------|");
exp.add("B  |oo--------------|----------------|oo--------------|-------o-o-oo-o-|");
```

```
assertEquals(exp, act1);
assertEquals(exp, act2);
```

Checks if its been added twice

Sufficiency: This test is sufficient as this shows us that the parser interprets the repeat. And adds it multiple times according to the number.