Test Document

Group Members responsible for Testing and writing the Test Document:

Mahdiar Shoraka
Hieu Le

1. ClefTest ("src/test/java/custom_component_data_test/ClefTest"):

Clef Class

+ is specifically declared near the beginning of a measure, otherwise it is the same as the previous measure.
+ Contains 2 attributes: symbol (defined in <sign> </sign>) and line (defined in <line> </line>)
+ Regarding the symbol, it is represented with letters from A to G. If the value mentioned in <sign> </sign> is either percussion or TAB, that means the symbol is G and the line number is 2

Tests:

- **ClefConstructorTest1()**: the test checks if the class functions normally by creating a Clef object and check if it is null
- **ClefTest1()**: This test uses demoDrumsSimple1.xml as the test file. The test checks if the retrieved info of the clef in the 1st measure of the song is correct or not

2. InstrumentTest ("src/test/java/custom_component_data_test/InstrumentTest"):

Instrument Class

+ Is defined inside of <score-instrument> and <midi-instrument>.
+ Contains 7 attributes: id (defined in <score-intrument id = " " > or <midi-instrument id = " "), name (defined in <instrument-name>), volume (defined in <volume>), pan (defined in <pan>), midiChannel (defined in <midiChannel>), midiProgram (defined in <midi-program>) and midipitched (defined in <midi-unpitched>)

Tests:

- **toStringTest()**: Tests the functionality of toString method in Instrument.java.
- **InstrumentTest1()**: This test uses demoBassSimple1.xml as the test file. The test's purpose is to check if there are any instruments played in the melody or not. Since there is no info regarding instrument in this file, it is expected that the method to retrieve the info will return null
- **InstrumentTest2()**: This test uses demoDrumsComplex1.xml as the test file. Since there are various instruments used to play the song, the test checks the total number of instruments then check to see if the retrieved info of an instrument is correct

3. MeasureTest (“src/test/java/custom_component_data_test/MeasureTest”):

Measure Class

+ On a music sheet, each part is divided into smaller bars and these are the measures. It is defined inside <measure> </measure>
+ A measure contains 7 attributes: number of divisions (defined inside <divisions></divisions>), fifths (defined in <fifths> </fifths>), time signature and time display (defined in <time> </time>), clef (defined in <Clef> </Clef>), number of staff lines (defined in <staff-lines> </staff-lines>, a tuning map (defined in <tuning-step> and <tuning-octave>), a list of notes.

Tests:

- **testMeasure1(), testMeasure2(), testMeasure3(), testMeasure4()**: Test Measure.java in custom_component_data package using demoDrumsSimple1.musicxml in the test/resources package.
- Get the first measure in the list of measures (measure ”1”).
- Confirm that it's not null.
- Get the list of Divisions and Fifths of measure “1”.
- Confirms that getMeasures() method returns the correct number of measures in the xml file for a specific measure.
- Confirms that getFifths() method returns the correct number of fifths in the xml file for a specific measure.

4. NotationTest:

Notation Class

+ A special attribute of certain notes in a song. It Is defined inside of <notations> </notations> and within <note>.
+ This class contains 6 attributes: 3 Lists (each for Slur, Tied and Slide), position of the string and fret of the played note and ornament

Tests:

- **NotationTest1(), 2(), 3()**: These tests use demoDrumsSimple1 as the test file. The goal of all these tests is to check whether a notation exists within a note in each measure of the song

5. NoteTest:

Note Contains information about the note sound, position, duration, is it part of a chord, etc.

Tests:

OrnamentTest:

6. PartTest:

Part Class

+ This class contains 3 pieces of info of the melody: name of the part, a part id number and a list of different instruments.
+ Each of the said info pieces also represent their respective attributes. The id is defined in <score-part id = " "> </score-part>, the name is defined in <part-name> </part-name> and the instruments are represented with a map, and each instrument is mapped with an ID of their own

Tests:

- **partTest1(), 2()**: The first test uses demoDrumsSimple1.xml while the second test uses demoBassSimple1.xml as the test file. Both tests have a common purpose, that is to check the correctness of info related to a part such as the name of the part, its id, the total number of parts in the tested file, the number of measures and instruments in a designated part
- **partTest3() and partTest4()**: Test Part.java in custom_component_data package, using demoDrumsSimple2.musicxml and demoDrumsSimple3.musicxml files in test/resources package.
- Confirm that getParts() method returns the correct list of parts.
- Confirm getName() returns the correct string.
- Confirm that getId() returns the corect int.

7. ScoreTest:

Score Contains all the data including title and author, and partList.

Tests:

- ScoreTest2() and ScoreTest3(): Test Score.java in custom_component_data package, using demoDrumsSimple2.musicxml and demoDrumsSimple3.musicxml files in test/resources package.
- Confirm that getAuthor() method returns the correct author.
- Confirm getParts().size() method returns the correct size.

8. SlideTest: ("src/test/java/custom_component_data_test/SlideTest")

Slide class:

+ One of the 3 special notations (slur, slide, tide) that could be assigned to a note and its adjacent one. It is defined in <slide> </slide> and within <notations> </notations>.
+ A Slide object has 2 attributes: type (could be “start” or “stop”) and number (the index of that current notation)

Tests:

- **SlideTest1() - 4()**: using demoGuitarSimple1.musicxml and demoGuitarSimple2.musicxml files in test/resources package.
- Confirm getNotation().getString() returns the expected string from the xml file.
- Confirm getNotation().getFret() returns the expected fret from the xml file.

- **SlideTest5()**: using demoDrumsSimple1.musicxml as the test file, the purpose of this test is to see if a special notation exists in a measure or not, in this case the Slide notation. It also checks if the 3 special notations (each represented by a list of respective type) function accordingly with each other, that is *if 1 of 3 special notations already exists in a note, the 2 other notations MUST NOT exist within the said note*.
    + In order to test this theory, the test first proves the existence of 1 of the notations in a certain note by checking the list of the same notation type (getSlurs().get(0), getTides().get(0) or getSlides().get(0)). If the tested notation doesn’t exist, a NullPointerException is expected to be thrown. Otherwise, an object of said notation type will be created. For example, in the tested file, it has been confirmed that in the 1st measure of the rhythm, there is a total of 13 notes and the last two notes are tied together, meaning 2 Tied objects are expected to be created at note index 11 and 12
    + Once a notation has been confirmed to exist in a note, The test then proceeds to check all the notes in the measure to see if the other 2 notations exist or not. In the 1st measure of the tested file, since only a Tied notation exists in note index 11 and 12 in measure 1, Slur and Slide must not exist in those notes as well as the rest of them. Therefore, the test is expected to throw a NullPointerException at every index retrieved from getNotes().get(index).getNotation().(getSlurs() or getSlides()).get(0)

- **SlideTest6()**: Similar testing method to SlideTest5() but the test file is demoGuitarComplex1.xml
- **SlideTest7()**: This test aims to test the correctness of the retrieved attributes assuming a Slide object is created at a certain index

9. SlurTest: (“src/test/java/custom_component_data_test/SlurTest”)

Slur Class:

+ One of the 3 special notations (slur, slide, tide) that could be assigned to a note and its adjacent one. It is defined in <slur> </slur> and within <notations> </notations>.
+ A Slur object has 3 attributes: type (could be “start” or “stop”), placement (could be “above” or “below”) and number (the index of that current notation)

Tests:

- **SlurTest1()**: Similar to SlideTest5(), the goal of this test is to check if a slur notation exists in a measure of or not. In this test we still use measure 1 of demoDrumsSimple1.musicxml
- **SlurTest2()**: Similar to SlideTest7(), this test checks the correctness of the retrieved attributes assuming the tested note has a slur notation. This test uses demoGuitarComplex1.xml as the test file
-

10. StaffTest:

StaffTuning: is defined in <staff-tuning> within <staff-details>
•tuningStep: char from ‘A’ – ‘G’. From *<tuning-step>D</tuning-step>*
•tuningOctave: integer value from *<tuning-octave>2</tuning-octave>*
Tests:

- staffTest1(): confirms that getStaffLines() gets the correct integer from the xml file.
- staffTest2(): confirms that getTuning() function doesnt return Null and gets Hashmap.

11. TiedTest: (“src/test/java/custom_component_data_test/TiedTest”)

Tied Class:

+ One of the 3 special notations (slur, slide, tide) that could be assigned to a note and its adjacent one. It is defined in <Tied> </Tied> and within <notations> </notations>.
+ A Tied object has 3 attributes: type (could be “start” or “stop”), placement (could be “above” or “below”) and number (the index of that current notation)

Tests:

- **TiedTest1(), 2()**: Similar to SlideTest7(), this test checks the correctness of the retrieved attributes assuming the tested note has a tied notation. Both these tests use demoDrumsSimple1.xml as the test file