TAB2XML

Design Document

Final Version

03/29/2021

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

Table of Contents

List of Figures

1. 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. Software Structure

2.1 Classes

2.1.1 Note.java

Note is a basic data structure that represents a note in a tablature. The Note class assists with MusicXML generation and stores essential attributes such as step, octave, string, and column. Note class is an extendible class and the parent class is only designed for bass and guitar notes.

2.1.2 DrumNote.java

DrumNote is a basic data structure that represents a note in a drum tablature. The DrumNote class extends the Note class. Besides the attributes in its parent class Note, the DrumNote class stores drum-specific attributes such as note head, instrument ID, and instrument name.

2.1.3 GraceNote.java

GraceNote is a basic data structure that represents a grace note. The GraceNote class assists with MusicXML generation and stores essential attributes such as noteList and complexTypeList.

2.1.4 Measure.java

Measure is a basic data structure that represents a measure. The Measure class assists with MusicXML generation. Measure is a class that is mainly used to store all Note objects that belong to one measure. A measure object usually stores attributes such as time signature, divisions, and measure number.

2.1.5 NoteUtility.java

NoteUtility, like its name implies, is a utility class that assists with string parsing. NoteUtility has a 2D array with guitar notes based on note location on the guitar string. It also has a map that maps instrument abbreviations to drum notes. NoteUtility has multiple methods that help with note and note type retrieval.

2.1.6 Beam.java

Beam is a class that is responsible for handling the beam musical notation that is used to connect multiple consecutive notes with a few occasional rests.

2.1.7 Controller.java

Controller is a class used in combination with JavaFX scene builder to define behaviours of GUI elements. Controller connects the front-end elements to the back-end functionality. It contains control methods for the text area, buttons, and checkboxes. It is also responsible for launching additional secondary windows based on user actions.

2.1.8 XMLGenerator.java

XMLGenerator is the main class responsible for translating the object representation of parts, measures, notes, and each of their respective properties and attributes into XML format. This class detects the differences between different instruments and adjusts the format of the output accordingly.

2.1.9 XMLUtility.java

XMLUtility is a helper class to XMLGenerator and is responsible for accounting the instrument type and help in detecting differences between the three instruments and selecting the elements that belong to the final MusicXML file for each instrument.

2.1.10 StringParserUtility.java

StringParserUtility is a class that is responsible for parsing the input by extracting all necessary information so that it may be translated into MusicXML format. It reads the input and creates notes and measures objects as per the tablature. Furthermore, it can detect the type of instrument, repeats, different types of notes, etc. In general, it precisely reads all of the vital information from the input tablature that is required to generate the counterpart MusicXML file.

2.1.11 TuningController.java

TuningController is a class used in combination with the JavaFX scene builder to define behaviours of GUI elements related to instrument tuning and octave selection. TuningController.java along with Controller.java connects front-end elements to back-end functionality. This class is called when the user selects guitar or bass as their instrument of choice or the auto-detect feature detects the user input as guitar or bass tablature.

2.1.12 StringParserUtilityDrum.java

StringParserUtilityDrum is a class that extends StringParserUtility.java and is responsible for specifically reading drum tablature and creates notes and measures objects according to the given drum tablature.

2.1.13 StringParserUtilityBass.java

StringParserUtilityBass is a class that extends StringParserUtility.java and is responsible for specifically reading bass tablature and creates notes and measures objects according to the given bass tablature.

2.2 Important Methods

2.2.1 public void confirmTranslate()

confirmTranslate is a method under Controller.java. confirmTranslate defines the behaviour when the “Confirm” button on the translate option screen. confirmTranslate method first confirms the validity of the input tablature, then it saves the composer, musical title, and time signature. Based on the detected or selected instrument type, confirmTranslate calls one of the subclasses of StringParserUtility and obtains a list of measures. The list is then passed onto XMLGenerator. XMLGenerator returns confirmTranslate the MusicXML content. confirmTranslate then sets the text area with the returned string.

2.2.2 public static ArrayList<Measure> stringParse(String input)

stringParse takes the string input given by the user which it then validates to ensure that it is a translatable tablature. If so, it splits the input into an array of measures, which are each as a string, and then calls the method MeasureParser. MeasureParser is able to translate each measure, which is a string, into the appropriate “Measure” object that we have created in our program. stringParse then returns an ArrayList of these Measures objects.

2.2.3 public static String getNoteType(float typeAsNum)

getNoteType is a method under NoteUtility.java. getNoteType returns a string representing the note type based on a float input of note duration. getNoteType method is frequently used during parsing to set the type attribute in a note object.

2.2.4 public static Measure measureParser(String measureString, int measureNum)

measureParser is a method under StringParserUtility. This method is responsible for parsing every note in each measure, and putting them into a measure object. It checks for all note types that are supported and parses those notes accordingly.

2.2.5 public static void parsingFunction(String currNote, String[] lines, Measure measure, int i, int j, float timeSignature, boolean isHarmonic)

parsingFunction is a method in StringParserUtility. This function is responsible for parsing notes which are of the “basic” type. This function takes the parameters which locate where the note is, and then create the according Note object. It then places that object into the Measure where it is located in. parsingFunctionComplex does the same thing however it only is used for complex notes such as grace, hammer-ons, pull-offs, and slides.

2.2.6 public static int getDivision(String measure)

getDivision is a function under StringParserUtility. This function takes a measure as a String and uses an algorithm to find the division of the measure. It then returns that division.

2.2.7 public static int getDuration(String lines[], int column)

getDuration is a function under StringParserUtility. It takes a String array which is the measure that the note is located in. It also takes the column where the note was located, and then executes an algorithm to calculate the duration of the note. Finally, it returns the duration.

3. Maintenance Scenarios

3.1 Additional Support for new instruments

1. A new class that inherits from StringParserUtility must be created to support the parsing functionality required for the new instrument.
2. Any differences between the parent and child class must be overridden in the child class.
3. Instrument type must be accounted for in XMLUtility.java to allow XMLGenerator.java to generate classes with correct formatting
4. A new note type must be created that inherits from Note.java. This new class will hold any new information pertaining to the instrument that isn't included in the parent class.

3.2 Additional Support for new tablature features

To add support for any additional features in tablature format:

1. If the new features add information to any specific notes or measures, Note.java and Measure.java must be updated with new attributes to hold that information
2. stringParse(String input) in StringParserUtility.java must be updated to recognize the new tablature feature and update it in the Measure/Note classes accordingly
3. XMLGenerator.java must be updated to read the new information in the received measure list and print it into the output MusicXML file accordingly

3.3 Additional Support for different file formats

1. New file formats must be accounted for in a conditional at the beginning of stringParse(String input)
2. Any new string formats, encodings, etc. must be accounted for in previously mentioned conditional

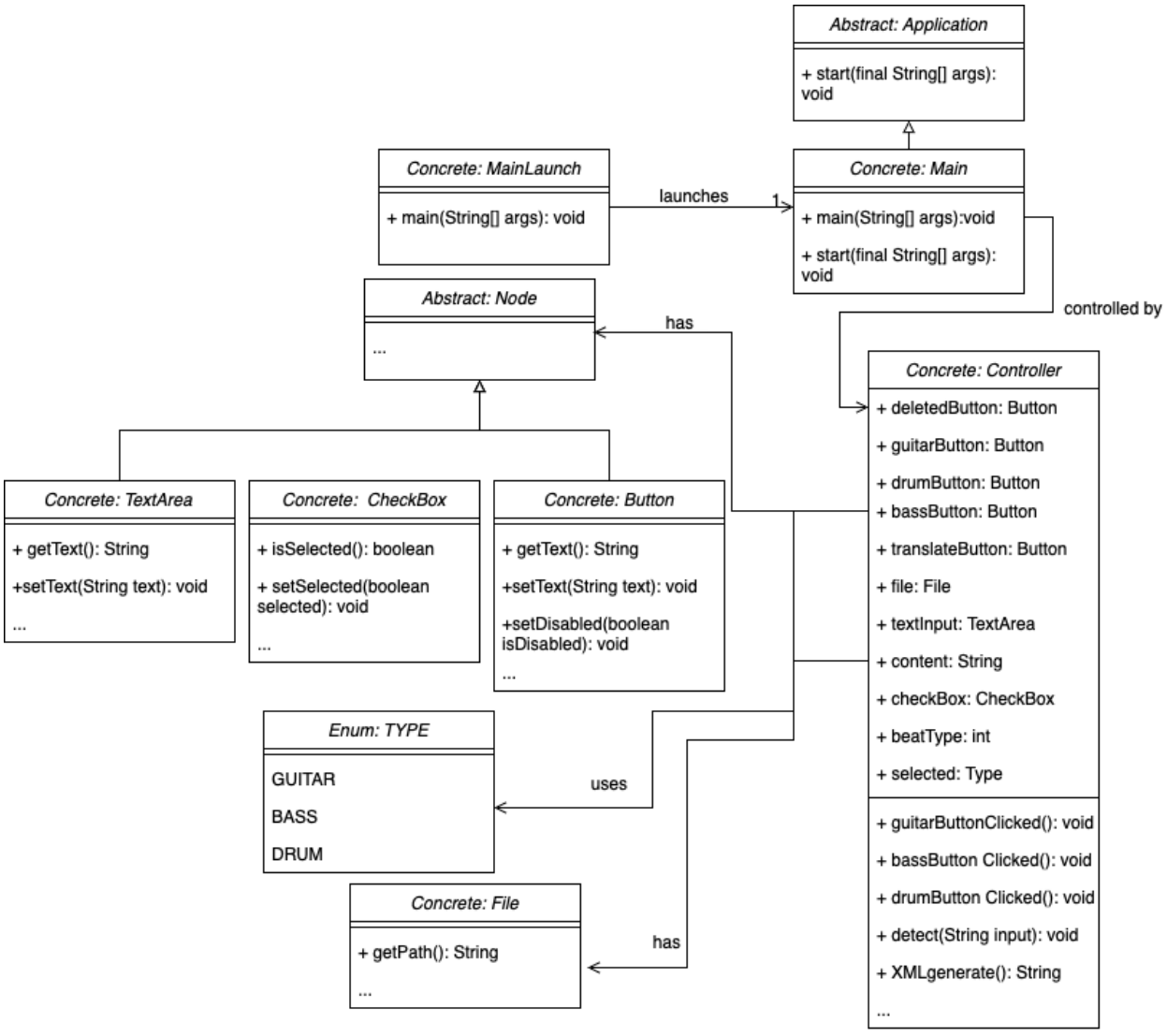
3.4 Additional Support for different tablature format

1. StringParse(String input) must be updated to support the new format, and adjust how it creates Measure objects and fills them with Note objects accordingly

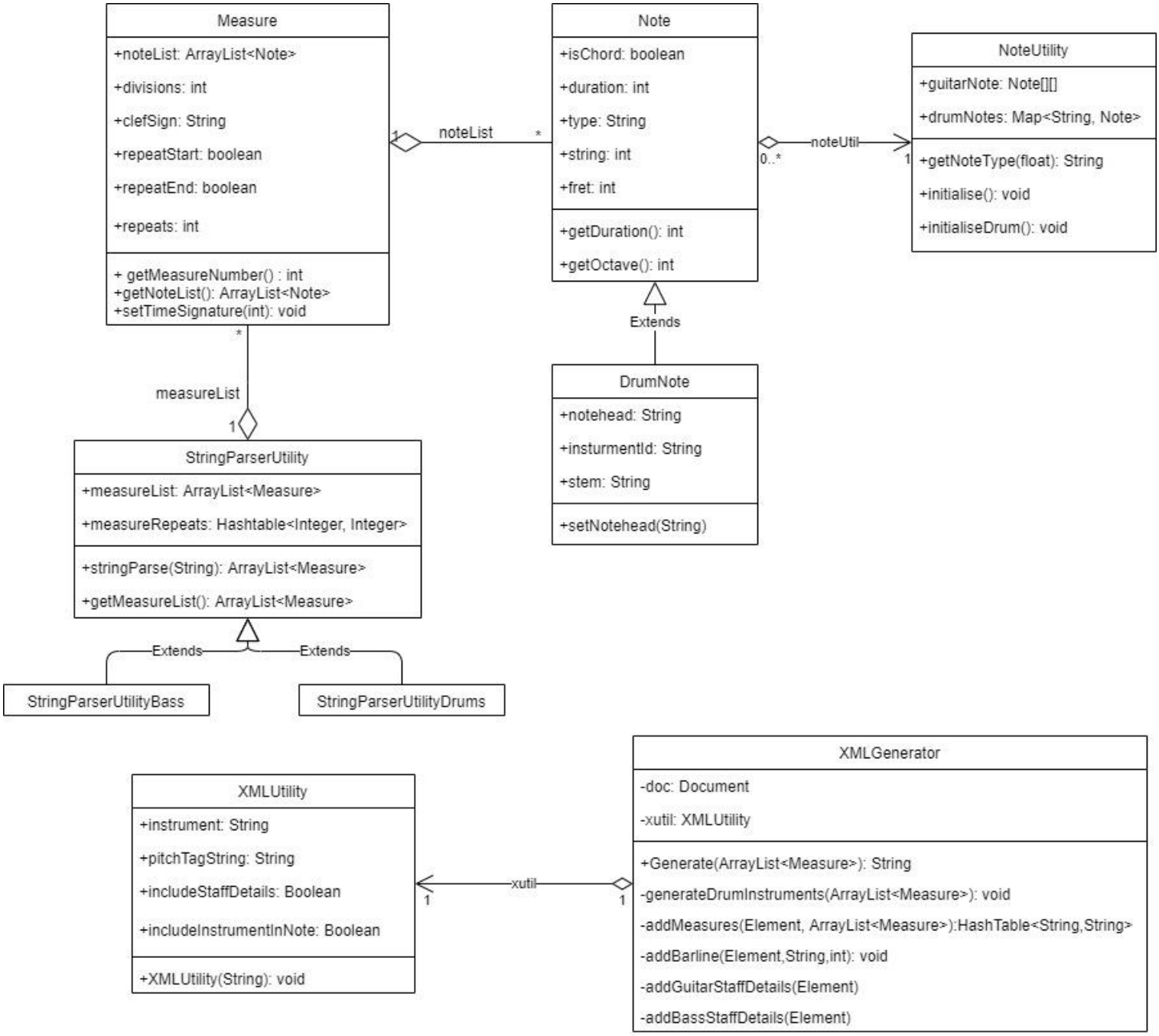
4. Diagrams

4.1 Class Diagrams

4.1.1 Front-End Class Diagram:



4.1.2 Back-End Class Diagram:

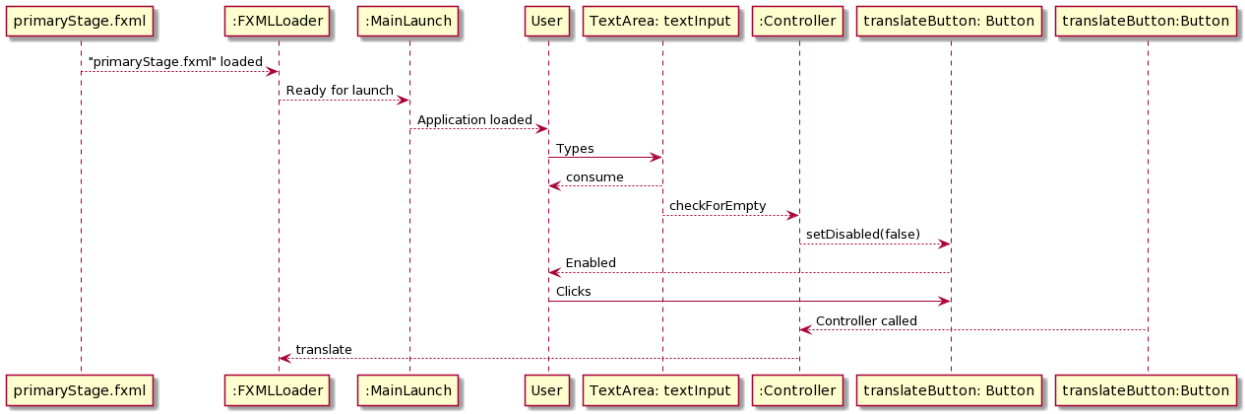


4.2 Sequence Diagrams

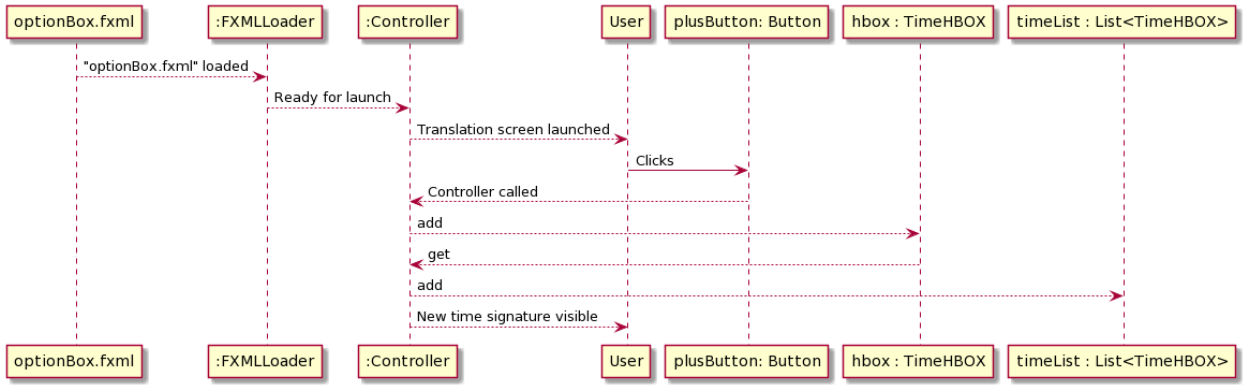
4.2.1 Front-End Sequence Diagram

The original front-end sequence diagram can be viewed with high resolution here.

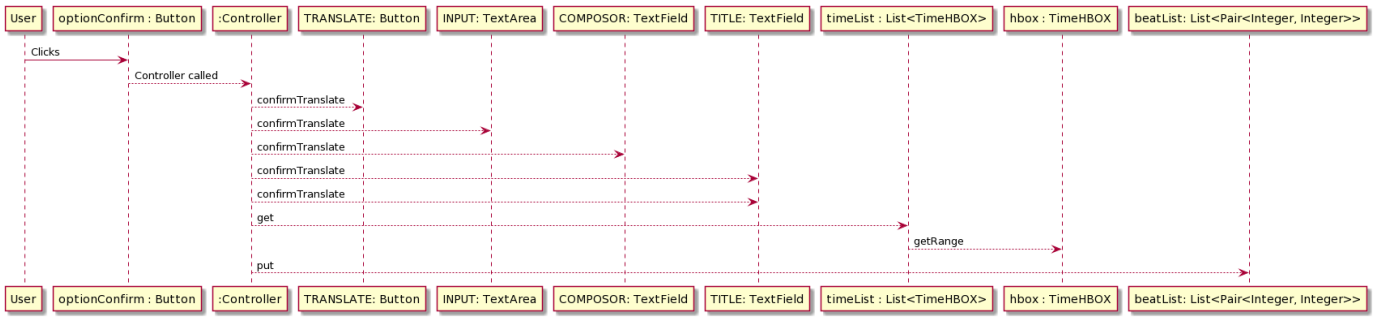
4.2.1.1 Primary Stage Launch



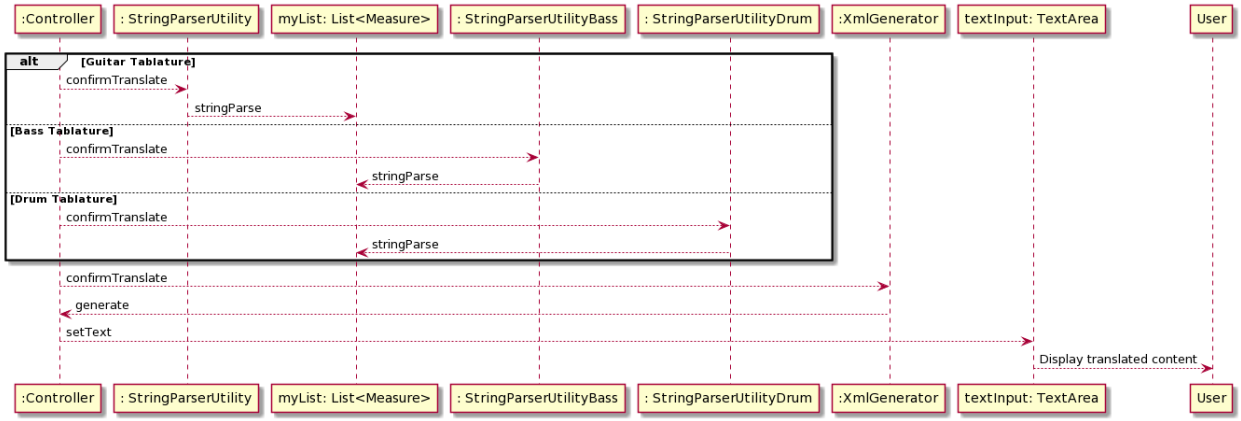
4.2.1.2 Translation Options Selection Process



4.2.1.3 Confirm Translation Process



4.2.1.4 Translation Process


:Controller
: StringParserUtility
myList: List<Measure>
: StringParserUtilityBass
: StringParserUtilityDrum
:XmlGenerator
textInput: TextArea
User
alt
[Guitar Tablature]
confirmTranslate
stringParse
[Bass Tablature]
confirmTranslate
stringParse
[Drum Tablature]
confirmTranslate
stringParse
confirmTranslate
generate
setText
Display translated content

4.2.2 Back-End Sequence Diagram