DESIGN DOCUMENT

Project: TAB2XML
Course: Software Development Project (EECS 2311)

THE TEAM (GROUP 12):

Matteo Pulcini - [217536756]
Krishna Raju - [218199497]
Patrick Qi - [218095091]
Riffi Manoj - [218061986]

Table Of Contents

1.0. Introduction

1.1 Overview

The purpose of this document is to describe how the software operates and performs in a concise manner from one viewpoint to ensure individuals editing or maintaining the software in the future can have a thorough understanding of how the software was developed and how it functions.

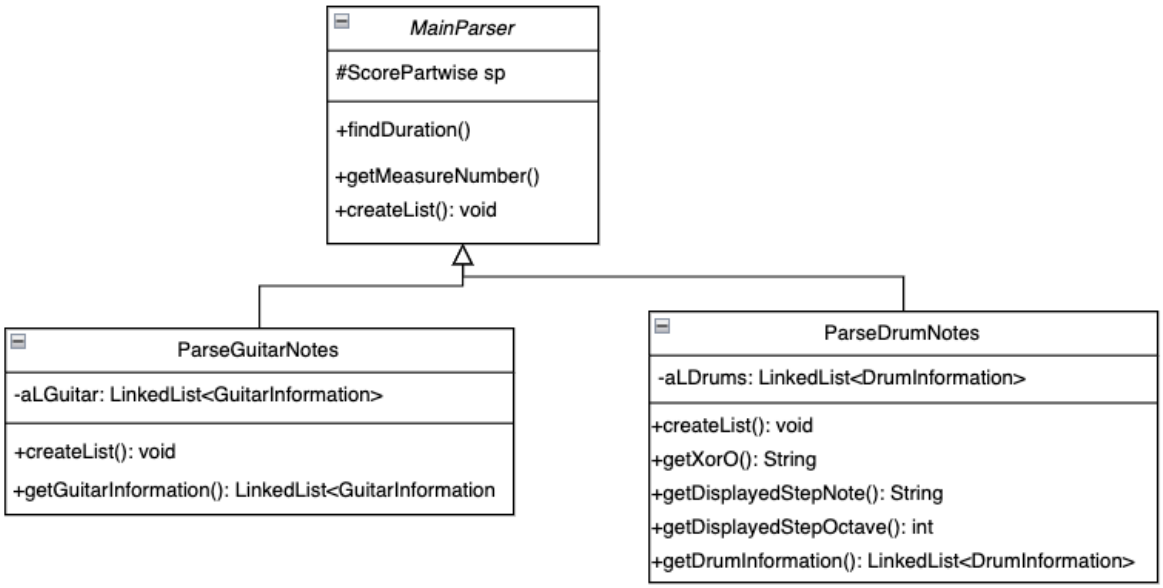
1.2 Scope

This document includes …

1. Viewing Tablature
2. Listening to input music audio
3. Moving to certain measures
4. Printing the Tablature
5. Downloading the tablature & audio file

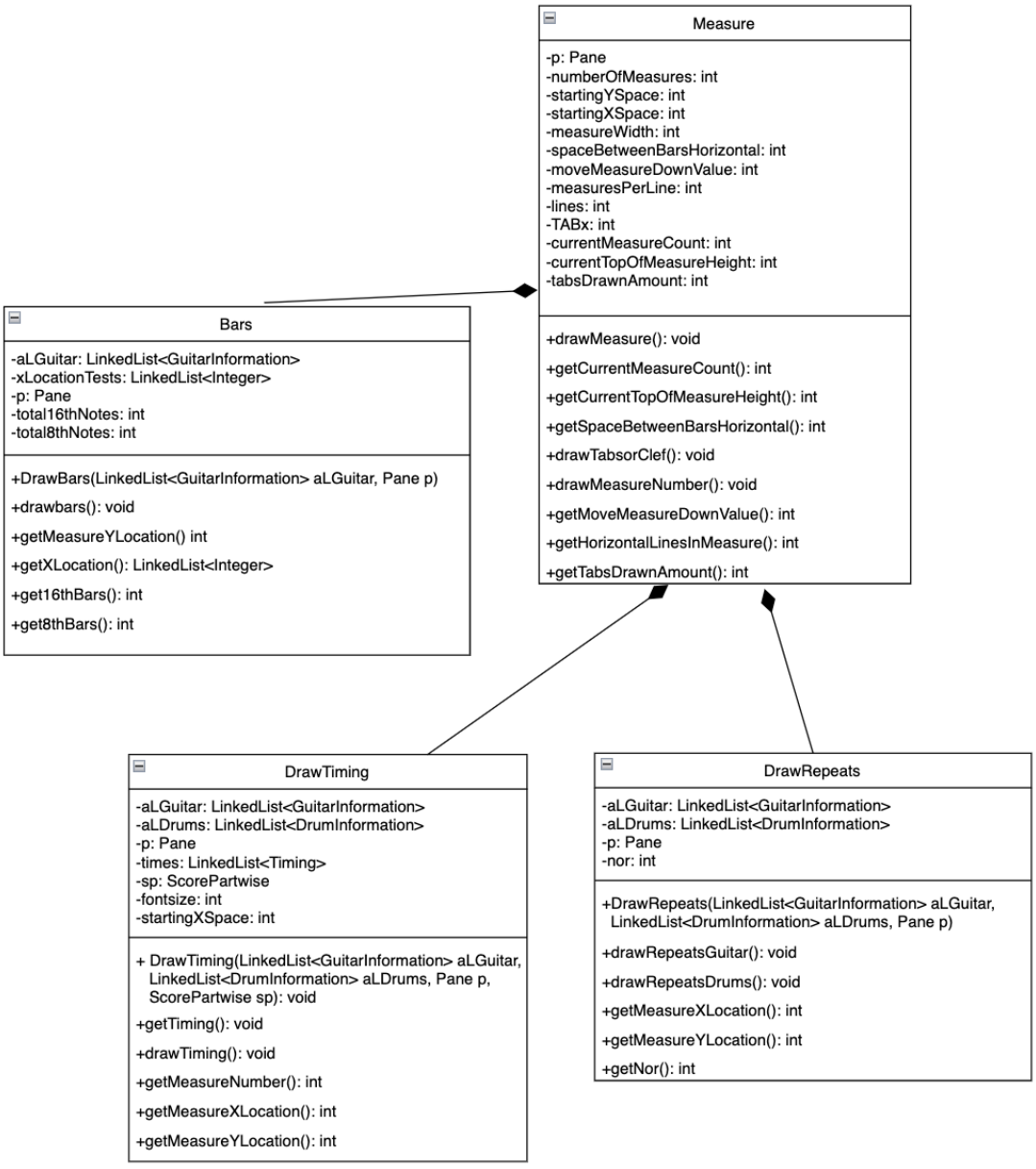
3.0 Class Diagram

3.1. Parser



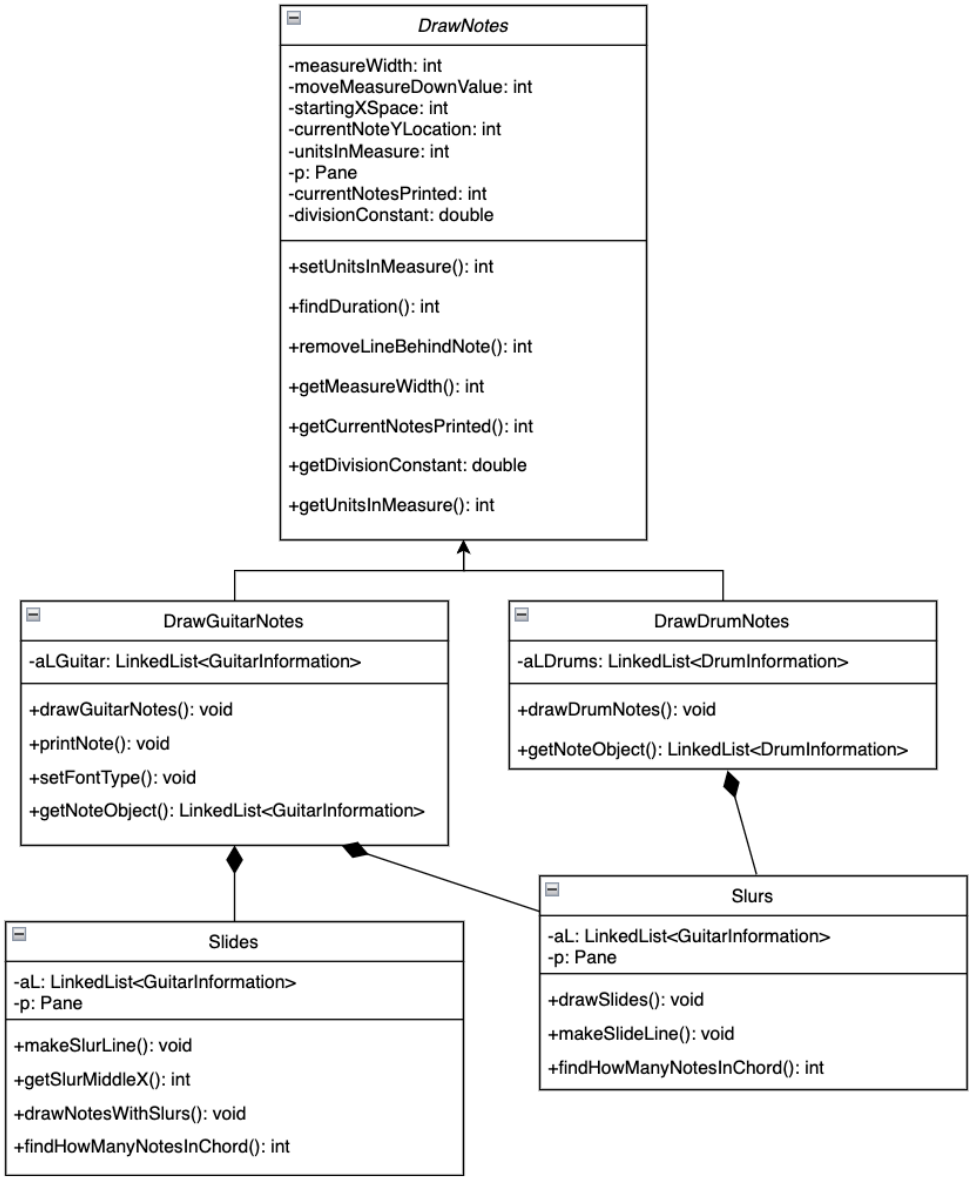
Description:
The Main Parser is a parent class to the Guitar and Drum Parser, they both share the scorePartwise object which contains all the note information. The ‘sp’ object is parsed creating a list of objects of the type **GuitarInformation** or **DrumInformation** (view in section 3.3), depending on how the software flags the input. Both objects are similar but contain minor differences, they both contain information about the type of note, its X and Y position if the note is a chord, a slur, a grace note, the current measure of that note, slurs, and the duration of a note. Meanwhile, guitars have a string and fret numbers along with sides.

3.2. Draw Measures



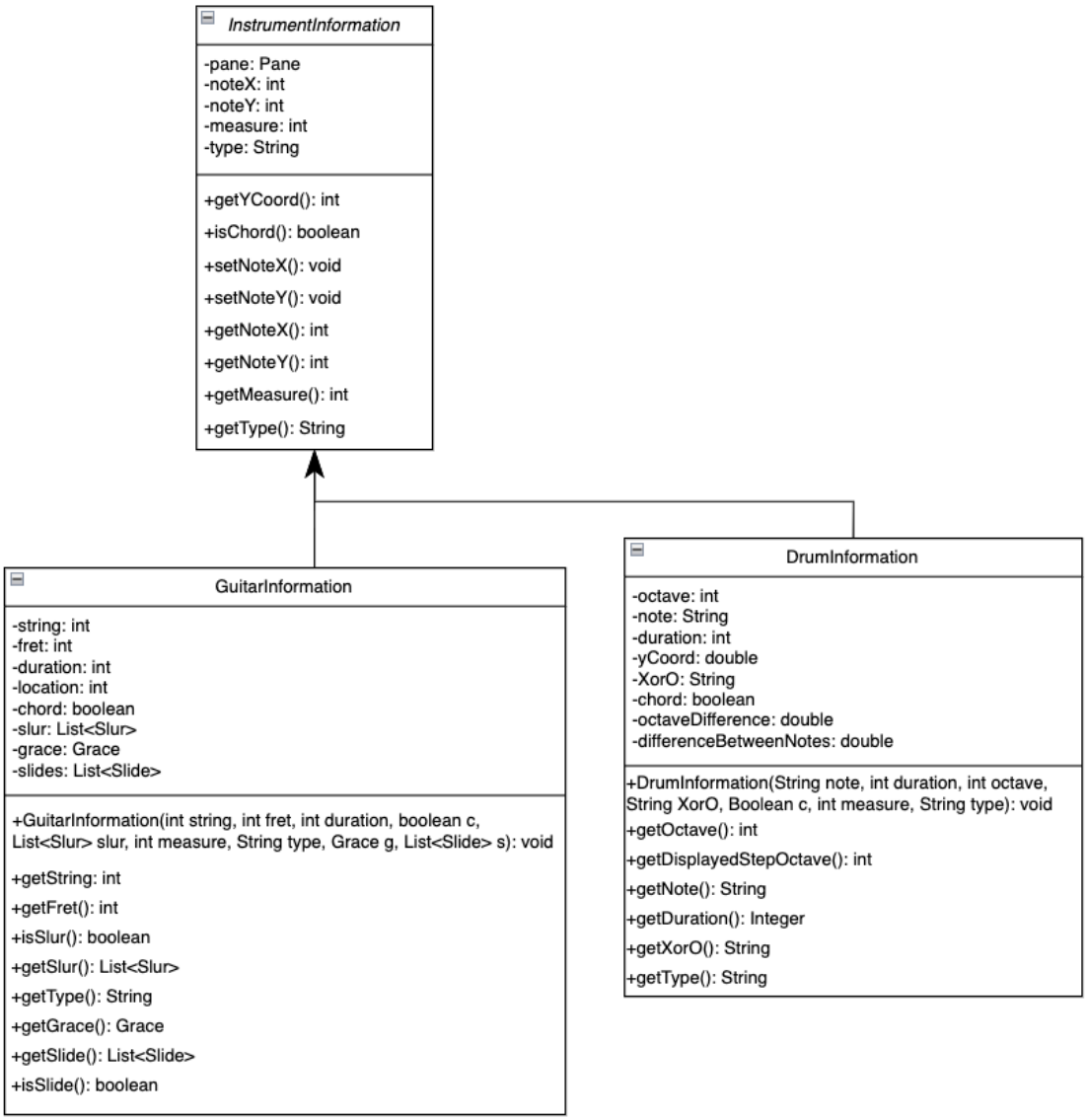
Description:
After the ScorePartwise is parsed and placed into a LinkedList of type DrumInformation or GuitarInformation we can draw the correct amount of measures on the pane along with bars underneath each measure to represent duration for Guitar notes as well as timing and repeats which are derived after the measured class.

3.3. Draw Notes



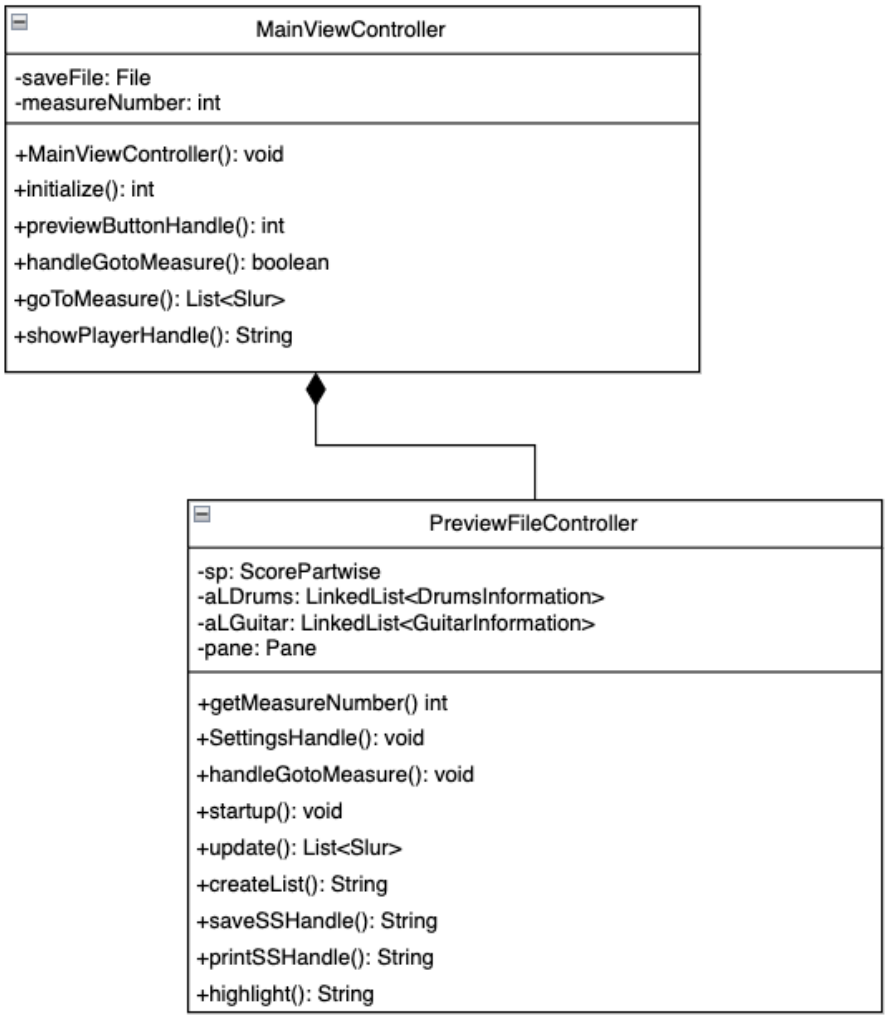
Description:
After the ScorePartwise is parsed and placed into a LinkedList of type DrumInformation or GuitarInformation we can draw the notes onto the pane. First, the Measure then, the notes are drawn onto the screen by DrawGuitarNotes or DrawDrumNotes with help of the parent class DrawNotes. Where extra features are drawn afterward such as slides and slurs.

3.4. Instrument Information



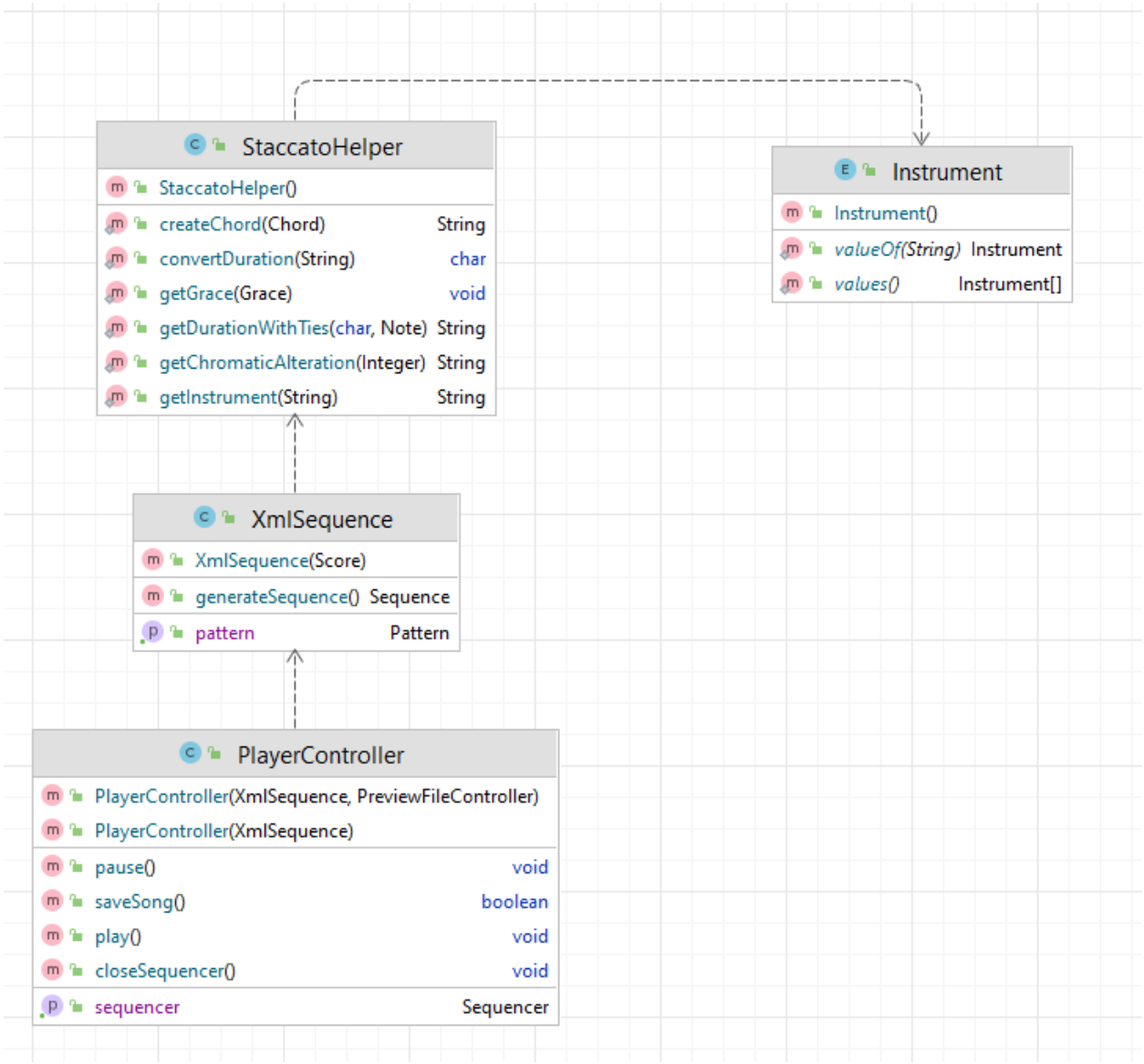
Description:
The InstrumentInformation class is the parent class to GuitarInformation and DrumInformation, which is the foundation for the List which contains a set of objects containing information regarding drums or guitar tablature.

3.5. GUI Controllers



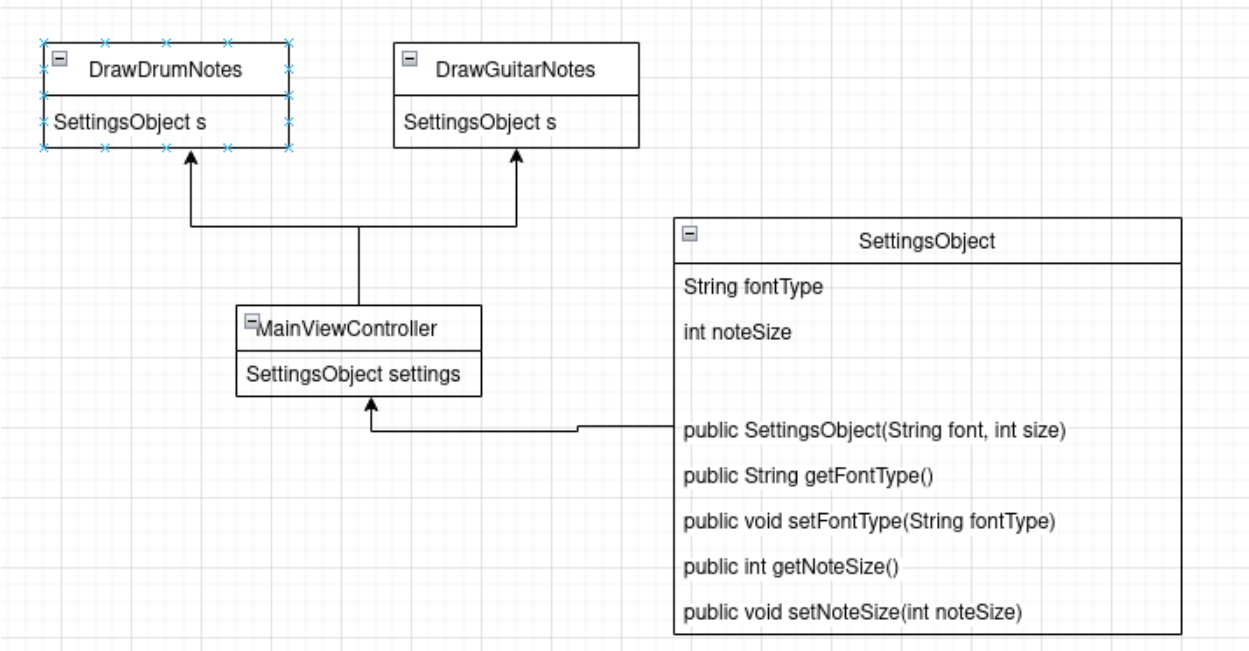
Description:
After the MainView is created the option to open the view of the tablature appears and hence PreviewFileController is run.

3.6. Music player



Description:
XmlSequence generates a JFugue Staccato string, to be played by a sequencer. The XmlSequence class parses from the already implemented Score and generates a String. generateSequence() appends onto a StringBuilder and returns the Sequence. The PlayerController constructor initializes the *Sequencer* which acts as the primary player of music. JavaFX.sound.midi.Sequencer was chosen instead of JFugue's Player because it is much more robust and modifiable, e.g. moving to an arbitrary position in the sequence and changing the tempo of the playback. initialize() sets up all the video/audio/tempo slider's with their relevant event listeners. Note this is done as opposed to the regular class constructor as FXML elements are not injected yet.

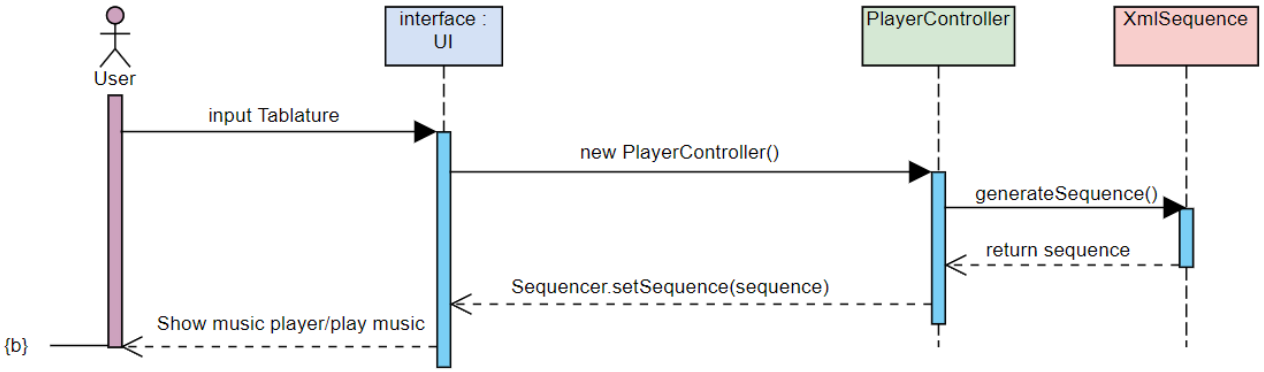
3.7. Settings Controllers



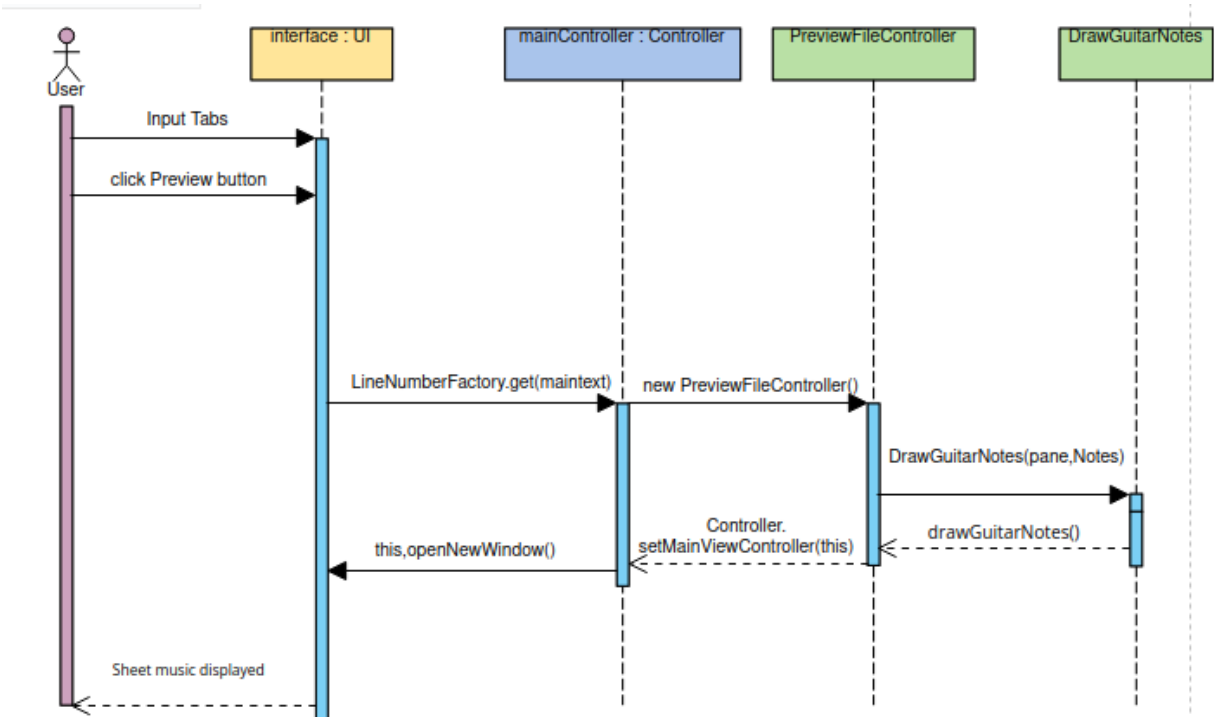
Description:
After the MainView is created the option to change the settings is granted by the SettingsObject class which closes the opened GUI and allows the user to relaunch it with new settings.

4.0 Sequence Diagram

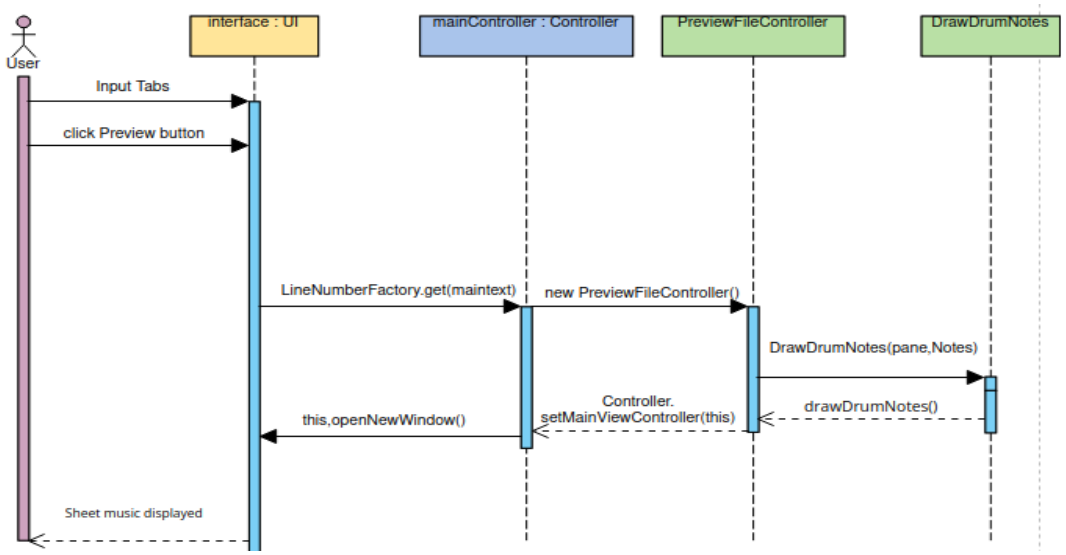
4.1. Music player



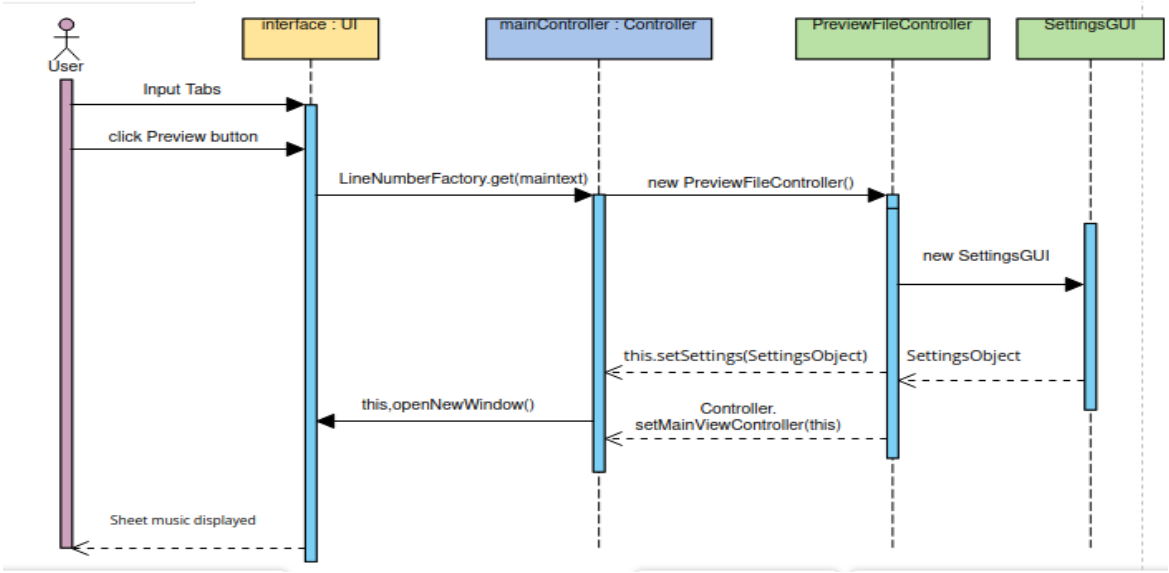
4.2. Sheet Music Display Guitar



4.2. Sheet Music Display Drums

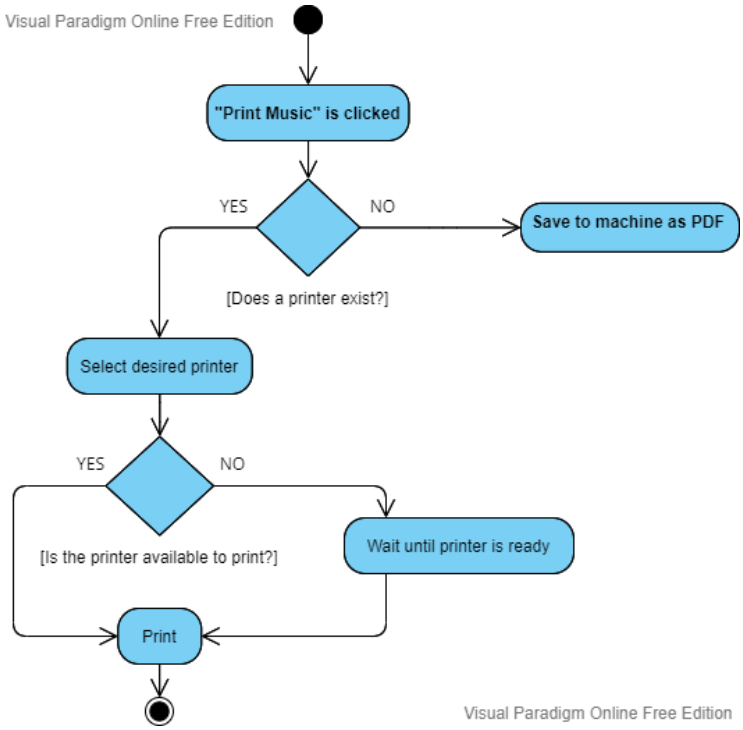


4.4. Customization

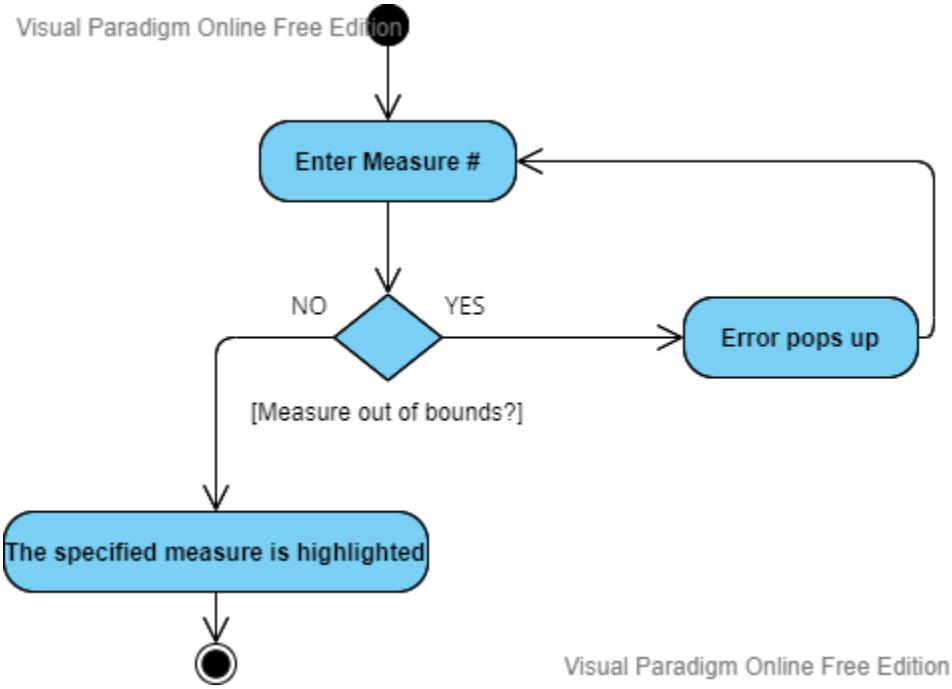


5.0 Activity Diagrams

5.1 Printing Feature:



5.2 Go-to-Measure:



6.0 Maintenance Scenarios

In order to perform proper maintenance on this system, a thorough understanding of the software is required. That is why we recommend checking out the user manual and the requirements document to get a thorough understanding of how this software is intended to function.

6.1 Main GUI Feature

Under the circumstances, the main GUI is not working, view the MainViewController class, as this is the class responsible for creating the MainView. Start debugging at methods called from the FXML documents such as initialize(), showPlayerHandle(), saveMXLButtonHandle() or showMXLButtonHandle(). These methods are run by pressing the buttons defined in the FXML classes. For extra features view the MainViewController methods and create similar ones in a similar way.

6.2 Preview GUI Feature

If the Preview Button does not work accordingly or has any noticeable bugs, check the PreviewFileController class. This class prints the input to the screen, first it prints the measures to the screen, then it creates a list of objects of the type DrumsInformation or GuitarInformation which store meaningful information about all the notes. It then parses the list placing each note in a specific Y & X coordinate in each measure, then adding any extra features. PreviewFileController should be modified if it is intended to add any aesthetics to the screen.

6.3 Play Music Feature

To support new playing features, refer to XmlSequence.java, that is where the sequence to be played is generated. Parse whatever notes are needed in the input, and then refer to JFugue documentation in order to see how to append the new string to implement your feature (must be supported by JFugue). Adding more functionality for the music player can be done in music_player.fxml and PreviewController.java, the controller for the GUI. Simply inject the new FXML element into your Controller, and then work from there.

6.4 Save Music Feature

If the save music functionality has issues. Troubleshoot the SaveSheetMusic class using a debugger. One common problem is that the device does not have a default printer that is used to print to pdf. To resolve this simply set the default printer on the device. If this does not resolve the issue, start by debugging the saveButtonClicked() method the issue most likely occurs at the ImageIO.write() method call this might be because it is unable to get the pane object which leads to a null pointer exception. Debug the method to find when the pane object is null.

6.5 Print Music Feature

The print functionality is handled by the printSSHandle() method in the PreviewFileController Class. Debug the method and check if the Printer is null. This is an issue because the program will not be able to get a printer to print the sheet music. To resolve this try to change the printer settings on the device and try again.

6.6 Customize sheet music appearance

To customize the appearance of the sheet music displayed the note type and font size fields need to be modified this will give the ability to use different fonts and make the notes displayed larger or smaller.

6.7 Custom saving location

To change the default save location the modification needs to be made inside the saveSheetMusic class. This class is responsible for saving functionality. A custom path will need to be passed in and the sheet music will automatically be saved to that path on request

7.0 Conclusion

All in all, our design document thoroughly outlines the structure of our program. Various different diagrams such as class diagrams and sequence diagrams are used to clearly outline the most important classes and methods in the software. The relationship between the important classes and methods in our system was shown in the class diagrams and the software used to create the class diagram was diagrams.net.