Design Document for the

Music XML Converter

LE/EECS 2311 - Software Development Project

Prepared for: Vassilios Tzerpos

Prepared by: Group 16

Maisha Rahman 215083876

Nabaa Gaziy 215820095

Wenxuan Li 216374324

Ali Sheikhi 216777120

Lars Jaylen Palalon 217280041

March 31, 2021

LIST OF CONTENTS

1.0 Introduction

This document describes the high-level structure of the Music XML Converter software, including all the important methods and classes in the software. Please do not hesitate to contact us for any questions or concerns at xmlconvertor@gmail.com.

2.0 Functional Description

The MusicXML Converter accepts music text tablatures and converts them into MusicXML files that are supported by many music apps, such as MuseScore. The user is able to convert bass, guitar, and drum text tablatures into MusicXML files. The next sections will describe the classes and methods within the software and how it uses them to convert the text tablature into MusicXML files.

2.1What are the Important Classes?

2.1.1 TablatureScanner

Description

This class scans the tablature to determine if the instrument is a bass, guitar, or drum based on the number of the strings each line contains and other features, and then creates the MusicXML output.

Important Methods

public TablatureScanner(String text)

- This is the default constructor
- It takes as input the tablature, stored in ‘text’

public String detect(String text)

- This method detects if the tablature (stored in ‘text’) is from a bass, guitar, or drum instrument.
    - Depending on this, it calls callBassClass, callGuitarClass, or callDrumClass to get the MusicXML output.
    - It then returns the MusicXML output

public static String callBassClass(String text,int count)

- Assembles the MusicXML output for a bass tablature by:
- Calling xmlHeader, printToXML, and endHeading
    - Appends each string into one StringBuilder object
- And returns it

public static String callGuitarClass(String text,int count)

- Assembles the MusicXML output for a guitar tablature by:
- Calling xmlHeader, printToXML, and endHeading
    - Appends each string into one StringBuilder object
- And returns it

public static String callDrumClass(String text,int count)

- Assembles the MusicXML output for a drum tablature by:
- Calling xmlDrumHeader, printDrumXML, and endDrumHeading
    - Appends each string into one StringBuilder object
- And returns it

2.1.2 StringInstrument

Description

This class creates the MusicXML output for string instruments, bass and guitar.

Important Methods

public StringInstrument()

- This is the default constructor for the StringInstrument object

public StringInstrument(String str1, String str2, String str3, String str4)

- This is the parameterized constructor for the StringInstrument object.

public static Bass getBass(String str1, String str2, String str3, String str4)

- Returns the Bass object

public static Guitar getGuitar(String str1, String str2, String str3, String str4, String str5, String str6)

- Returns the Guitar object

public static String xmlHeader(int c)

- Builds the header part for the MusicXML output and returns it.

public String printToXML(String str1, String str2, String str3, String str4, String str5, String str6,String str7)

- Builds the body part for the MusicXML output and returns it.

public static String endHeading()

- Builds the ending part for the MusicXML output and returns it.

2.1.3 Bass

Description

This class creates the Bass object. It inherits methods from StringInstrument class.

Important Methods

private Bass()

- Default constructor for the Bass object

private Bass(String s1, String s2, String s3, String s4, String s5)

- Parameterized constructor for the Bass object

public static Bass getInstance(String s1, String s2, String s3, String s4)

- Returns the Bass object

2.1.4 Guitar

Description

This class creates the Guitar object. It inherits methods from StringInstrument class.

Important Methods

private Guitar()

- Default constructor for the Guitar object

private Guitar(String s1, String s2, String s3, String s4, String s5, String s6)

- Parameterized constructor for the Guitar object

public static Guitar getInstance(String s1, String s2, String s3, String s4, String s5, String s6)

- Returns the Guitar object.

2.1.5 Drum

Description

This class creates the Drum object as well as it contains methods to return the MusicXML output for drum tablature

Important Methods

private Drum()

- Default constructor for the Drum object

private Drum(String s1, String s2, String s3, String s4, String s5)

- Parameterized constructor for the Drum object

public static Drum getInstance(String s1, String s2, String s3, String s4, String s5)

- Returns the Drum object

public static String xmlDrumHeader(int c)

- Builds the header part for the MusicXML output for the drum tab and returns it.

public String printDrumXML(String str1, String str2, String str3, String str4, String str5, String str6, String str7)

- Builds the body part for the MusicXML output for the drum tab and returns it.

public static String endDrumHeading()

- Builds the ending part for the MusicXML output for the drum tab and returns it.

2.1.6 window

Description

This class creates the GUI for this application. Also, the program can be launched by running this class.

Important Methods

public static void main(String[] args)

- Launches the application.

public window()

- Creates the application

private void initialize()

- Initialize the contents of the frame, such as buttons and menus in the GUI.

2.1.7 Class Diagrams

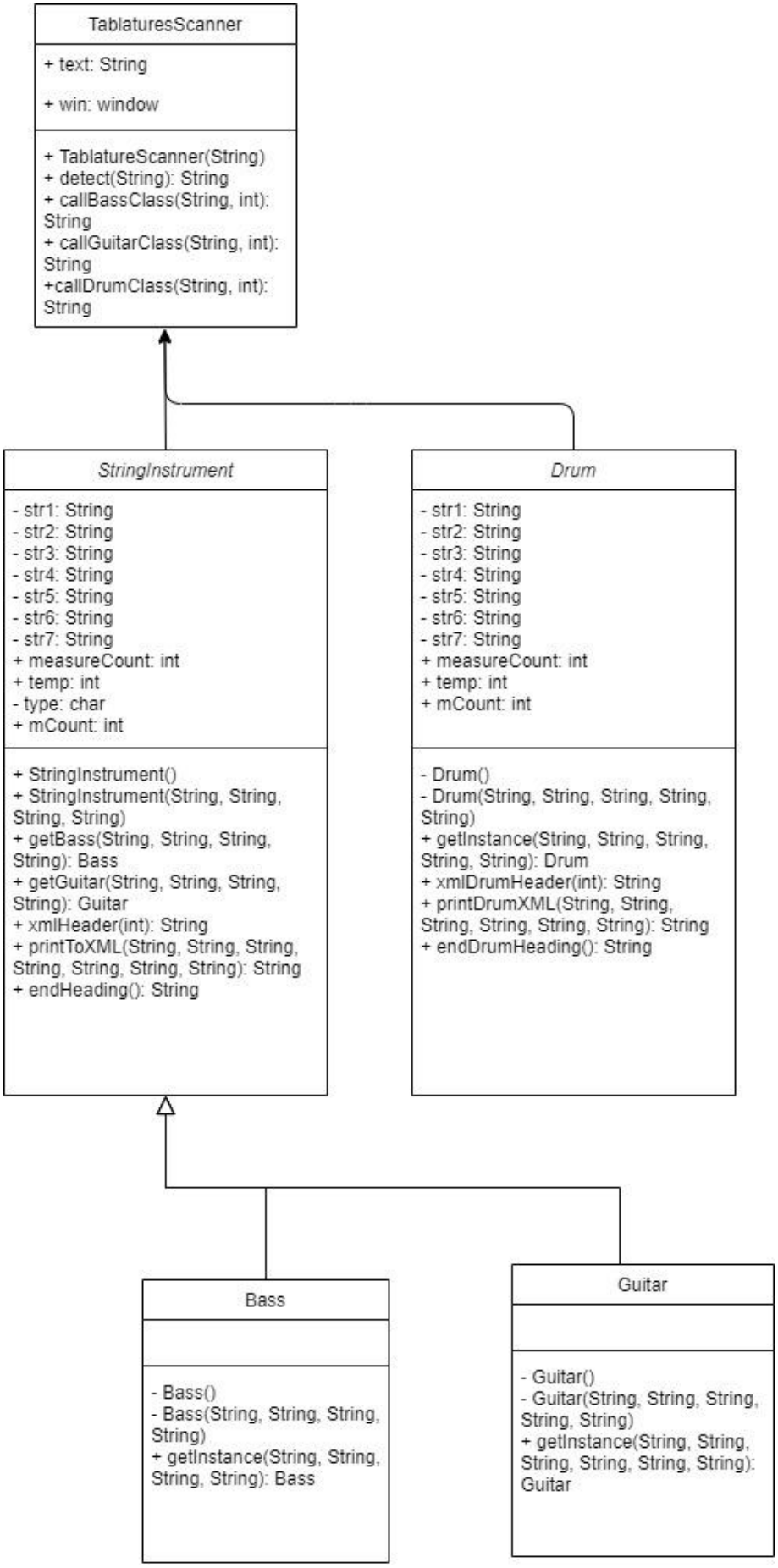

Figure 1. This class diagram shows the inheritance relationship between StringInstrument class and its children: Bass and Guitar classes.

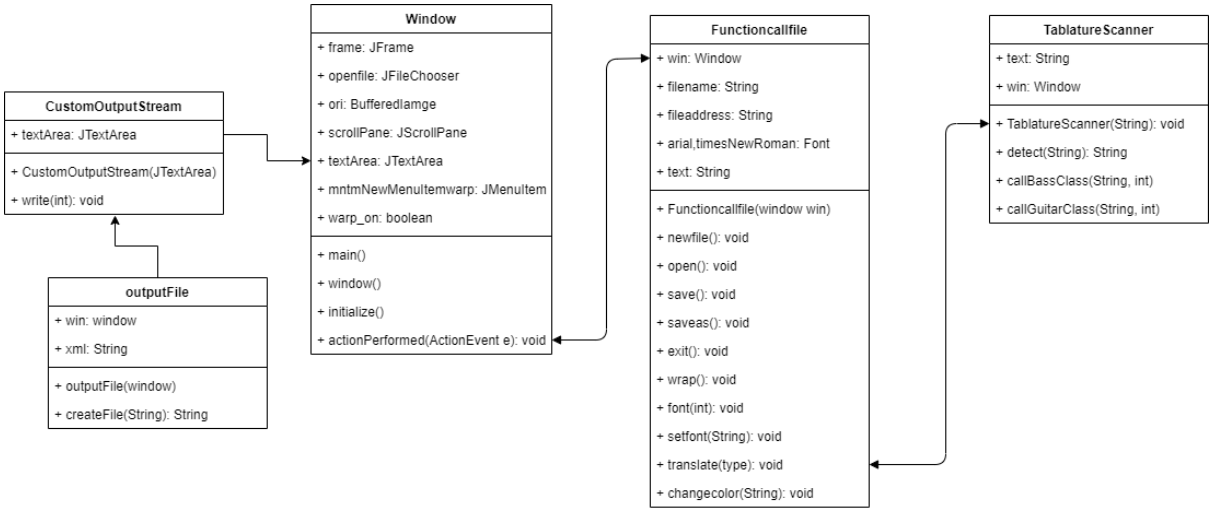

Figure 2. This class diagram shows the gui window how to work and how to print properly.

2.2 What are the Important Methods and Why?

The important methods in TablatureScanner class are detect, callBassClass, callGuitarClass, and callDrumClass. This is because the detect method detects if the input text tablature is a bass, guitar, or drum tablature. It then calls callBassClass if it detects the input as a bass tablature or callGuitarClass if it detects the input as a guitar tablature, or calls callDrumClass if it detects the input as a drum tablature. callBassClass creates a Bass object, callGuitarClass creates a Guitar object, while callDrumClass creates a Drum object.

The important methods in StringInstrument class are xmlHeader, printToXML, and endHeadings. These methods create the strings that contain the xml output. For instance, if a Bass object or Guitar object was created, then these methods from StringInstrument class are called, which appends the strings from xmlHeader, printToXML, and endHeadings into one stringbuilder object and then returns this object. The returned object is then printed in the GUI window.

In a similar fashion, the important methods in Drum class are xmlDrumHeader, printDrumXML, and endDrumHeading. If a Drum object was created, then these methods create the strings that contain the xml output. This output is then printed in the GUI window.

The important method in window class is initialize, it is responsible for calling the various parts of the entire GUI, and combining these parts together and making them visible. This is the most important bridge to communicate with the user and TablatureScanner.

2.3 How Do They Interact with Each Other?

2.3.1 Sequence Diagram

Scenario 1: If User Selects a Tablature and Translates it to MusicXML Format

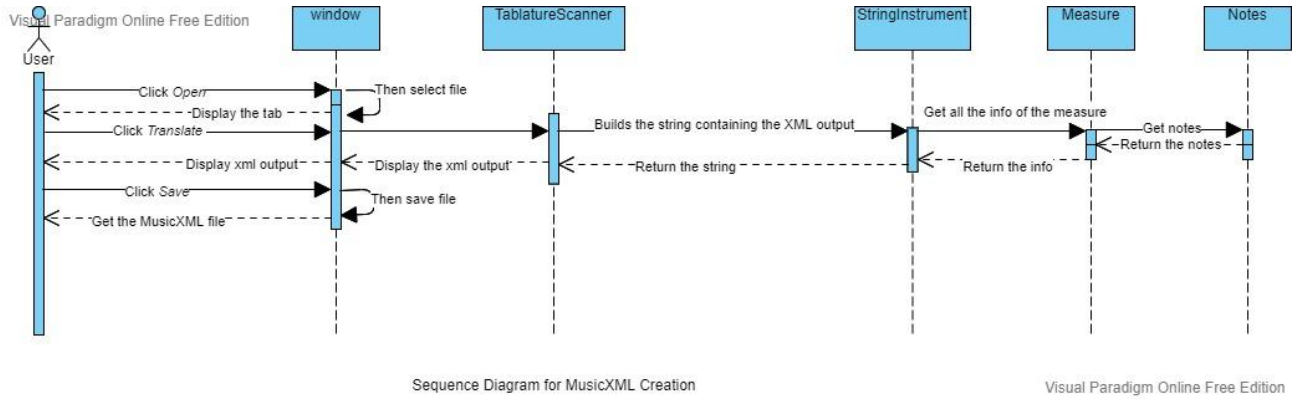

Figure 3. This sequence diagram shows how objects interact during runtime after selecting a tablature and selecting *Translate*. Basically, it shows the sequence of events when you run the application and translate a text tablature.

Scenario 2: If User Pastes the Tablature and Translates it to MusicXML Format

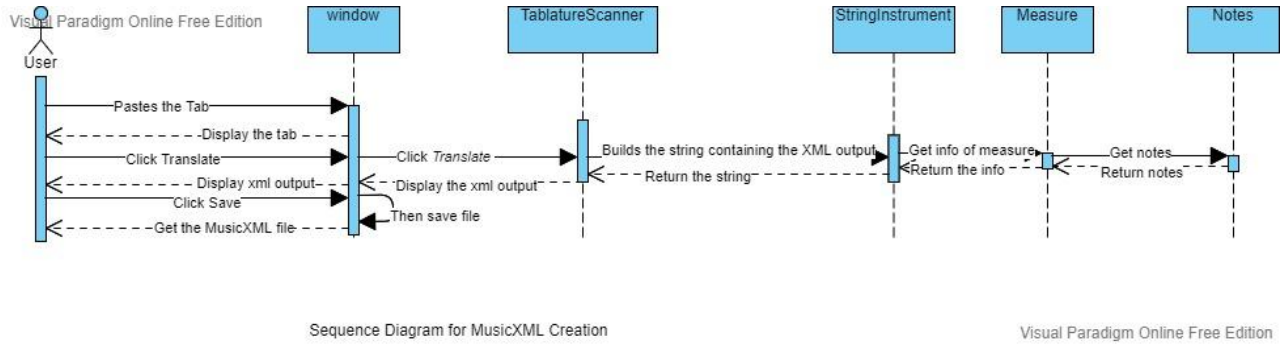

Figure 4. This sequence diagram shows how objects interact during runtime after pasting a tablature and selecting *Translate*.

As shown in **Figure 3** and **Figure 4**, the objects created in TablatureScanner, StringInstrument, Measure, and Notes interact with each other to create the translated xml output and return this string so that it can be displayed in the GUI window.

2.3.2 Activity Diagram

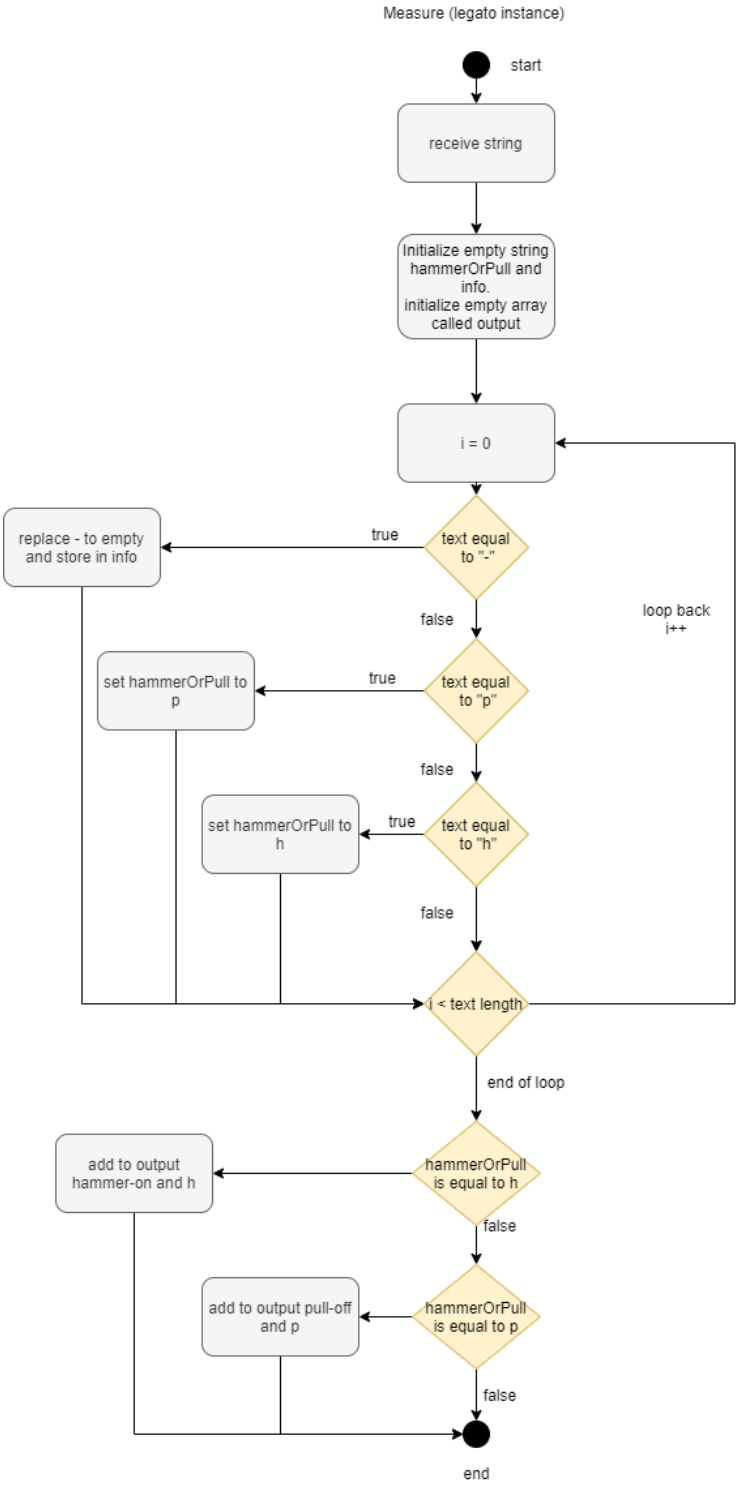

Figure 5. This diagram displays a situation of when a Measure legato instance is called. It shows how the method acts on certain inputs and what is supposed to happen after each value.

2.4 What are the Important Objects Created at Runtime?

The important object(s) created at runtime is the Bass object, Guitar object, or Drum object. This parser is able to separate the measures in a text tablature into xml format, store it in a string, and then return this string in xml format so that it can be displayed in the GUI window and be saved by the user.

3.0 Maintenance Scenarios

3.1 Updating the GUI

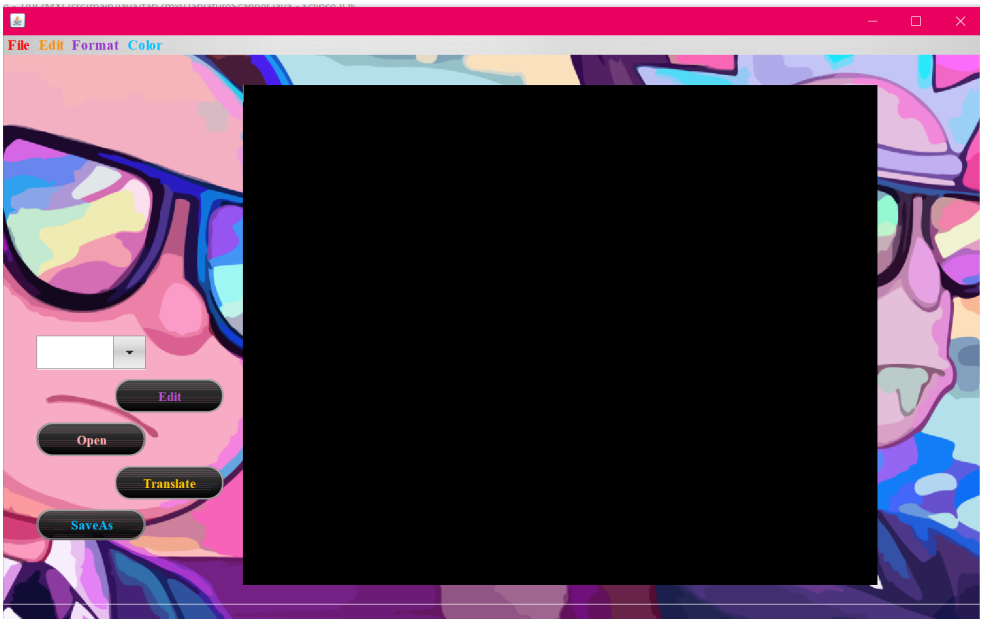

Figure 6. The updated GUI window for the Music XML Converter program.

In order to support the new features, we had to create an Editmeasure class that allows the user to edit the input text tab and change the time signature of the measures. After adding this class, the user is now able to click the *Edit* button, and select a different time signature in the new smaller window (refer to **Figure 7**).

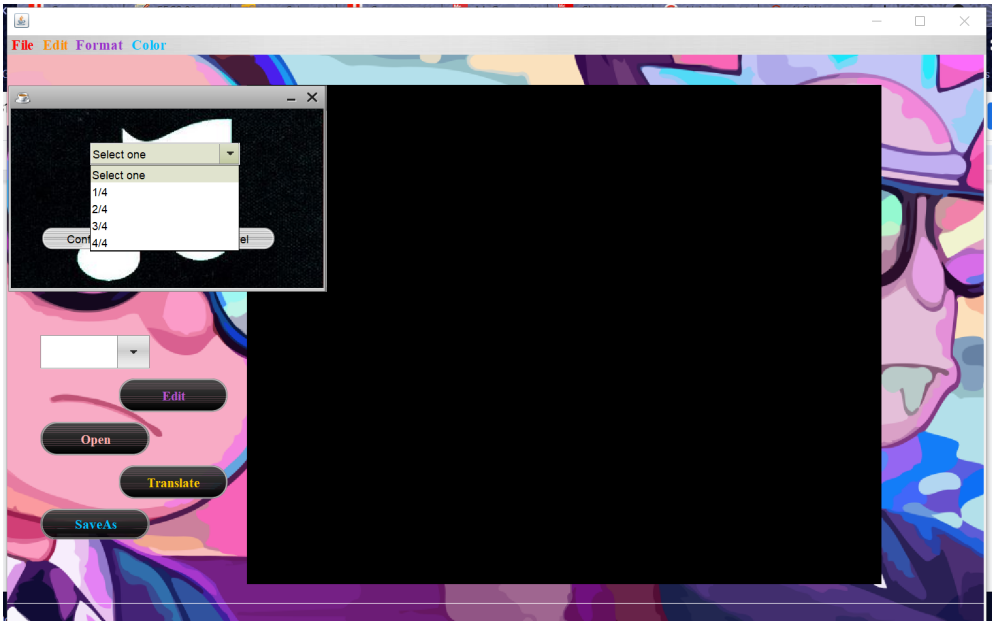

Figure 7. The user is able to change the time signature of the measure from the range of time signatures presented in the smaller window.

3.2 Supporting Additional Drum Instruments

In order to add support for additional drum instruments, the Drum class must be modified. It must contain constructors that will be used to create the Drum object. It also must contain methods that will create the String object (that contains the MusicXML output) depending on the information from the Drum object (like number of strings and the drum tablature itself). Also, the detect method (in TablatureScanner class) must be modified so that it can detect if the tablature is from a drum instrument. Then, in TablatureScanner class, the callDrumClass method must be implemented so that if the detect method detects that the tablature is from a drum, then the callDrumClass method will call the methods from the Drum class, build the xml output, and then return this string so that it can be displayed in the GUI.

3.3 Supporting Resizing of Window

In order to prevent distortion of the background image after resizing the GUI window, the background image must be updated so that it is the right size. It must be big enough to fit the enlarged GUI window. This image must be uploaded to the src/main/resources folder in .png format and the same name.

3.4 Adding New Feature such as Vibrato

In order to add a new feature such as vibrato, you need to modify the StringInstrument class. In the printToXML method each string would be parsed into columns so you need to add a new if statement that checks if the character “v” exists and then do the appropriate modifications to the StringBuilder output to reflect the vibrato. Any other feature can be added following the same approach. The listofColumns arraylist contains the arraylist of each column of the tablature so when the main for loop starts you can use the i iterator with another iterator to go through all the elements and look for the desired letter.

References:

[1] C. J. Fox, “Why Writing Software Design Documents Matters,” *Toptal Engineering Blog*, 10-Oct-2013. [Online]. Available: https://www.toptal.com/freelance/why-design-documents-matter. [Accessed: 26-Mar-2021].