Design Document

EECS 2311 – Group 6

Tuan Dau

James Le

Temiloluwa Odunfa

Mohammad Amin Mohammadi

Reviewers: Public

03/30/2021

Table Of Contents

1.Introduction

This Software Design Document (SDD) describes the detailed structure of the components of the Tablature Converter and the precise implementation details required to satisfy the systems requirements as specified in the Requirements Document.

This document shows how different parts of the system work together as demonstrated using the sequence and class diagram. It also talks about the design overview and maintenance scenarios that describes all measures required in order to ensure, maintain and enhance the functional capability of the system.

2.System Overview

Tablature notation is a way of writing down an instrument's parts in an accessible manner. The tablature in the context of this project will refer to guitar,bass and drum tablature. Although tablature is widely used for its ease of use, most tabs exist in a text format, and there does not exist an easy way to turn guitar, bass or drum tab text into a more featured digital file format.

The software product will consist of a desktop application. This application will convert a music tablature for guitar/bass/drum from ASCII text format to musicXML format".

3.Abbreviations and Acronyms

Abbreviations	**Definition**
SDD	Software Design Document
ASCII	American Standard Code for Information Interchange
GUI	Graphic User Interface
UID	User Interface Design

4.Design Overview

The constraints, goals and guidelines that were taken into consideration during the development of the Tablature Converter would be discussed here.

4.1General Constraints

i.) Schedule/Time:

This refers to the planning and schedule of the project for completion including the deadlines for each phase/requirements of the project.

ii.) Resource:

This refers to the project's budget and financial resources available for the successful completion of the project. Recognizing that cost in the context doesn't only mean money but other factors as well such as quality control.

iii.) Quality:

Taking into consideration if the program meets all the expected requirements.

iv.) Bug Fixing:

This refers to how successful the methods of testing were implemented as well as what methods were put in place in order to report and fix bugs.

v.)New feature requests :

What process or procedure would be put in place in order to implement new features to the program.

4.2 Goals and Guidelines

The set goal is to be able to create a completely functional system that converts a music tablature for guitar/bass/drum from ASCII text format to musicXML format". Guidelines taken while creating this program were as follow:

i.) User Interface Design(UID) :

While creating the UID, the goal was to keep the interface simple and create consistency by using common elements and features that would give users ease and comfort while fulfilling the purpose of the program.

ii.) Back End Coding:

The back end code for the Tablature Converter covers the aspect of parsing and getting data from the user and responding appropriately in a concise friendly manner

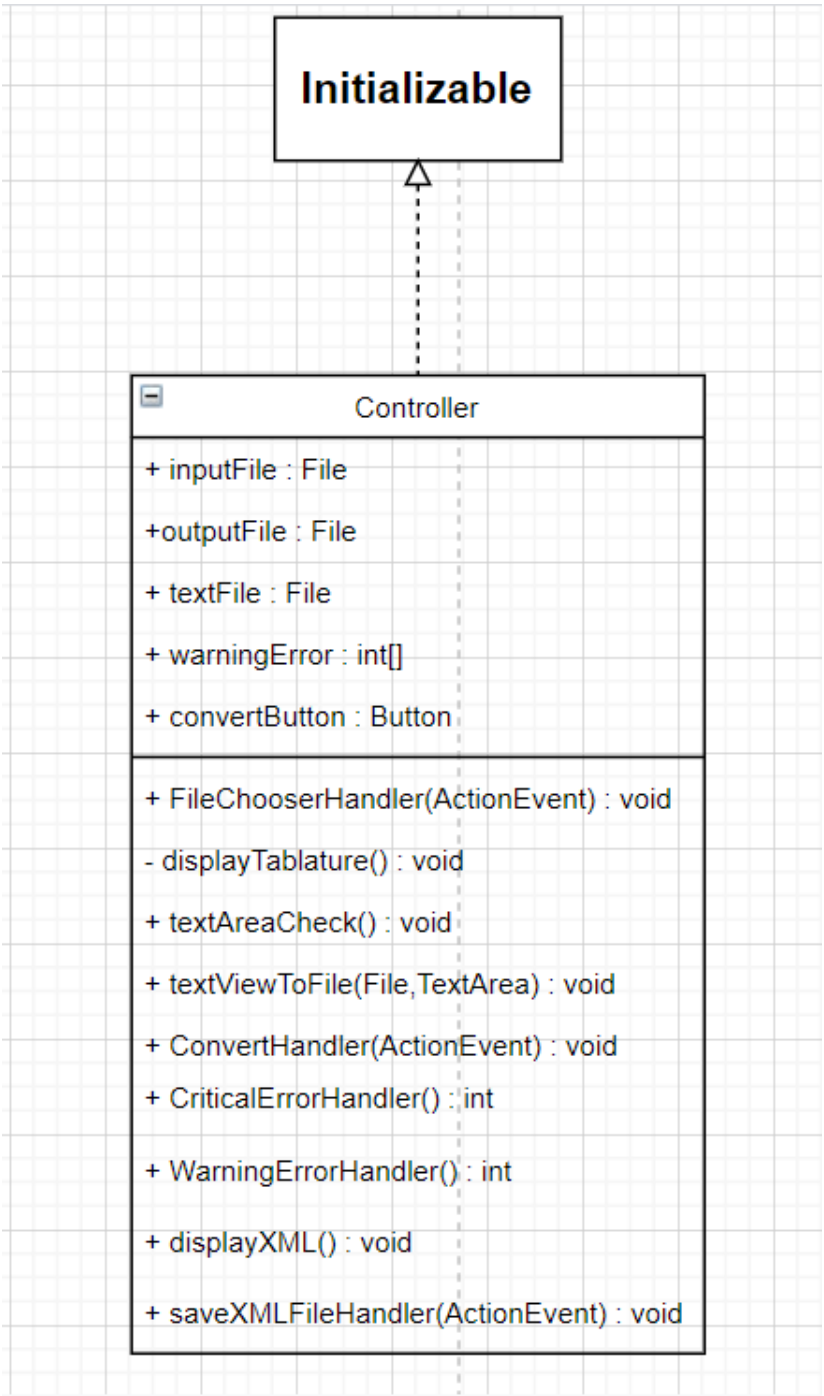
iii.)Testing:

The purpose of testing was to verify and detect errors in the system so as to maintain the expected functionality of the program

5.Class Diagrams

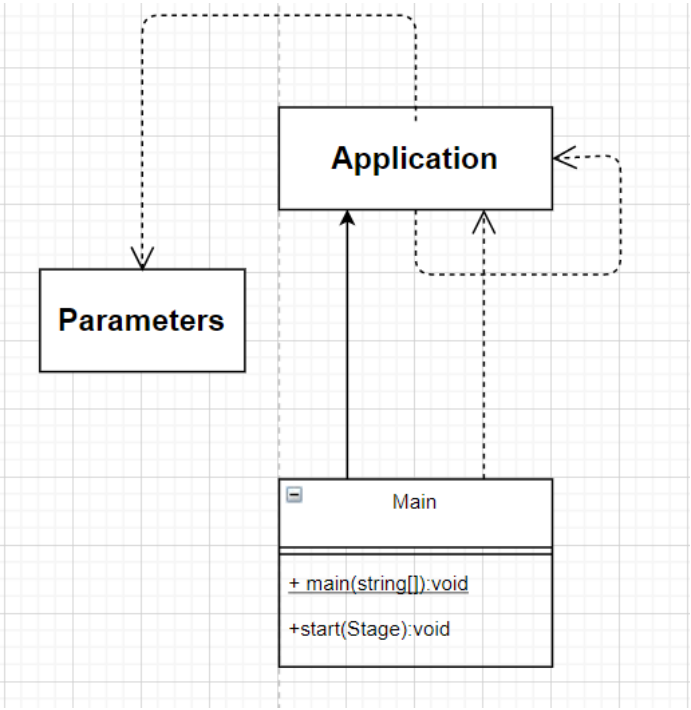
5.1.Controller :

This controller class controls the functionality of the GUI. It contains Action methods that handles the processing of incoming requests from the user and returns appropriate responses to the user.


Initializable
Controller
+ inputFile : File
+outputFile : File
+ textFile : File
+ warningError : int[]
+ convertButton : Button
+ FileChooserHandler(ActionEvent) : void
- displayTablature() : void
+ textAreaCheck() : void
+ textViewToFile(File,TextArea) : void
+ ConvertHandler(ActionEvent) : void
+ CriticalErrorHandler() : int
+ WarningErrorHandler() : int
+ displayXML() : void
+ saveXMLFileHandler(ActionEvent) : void

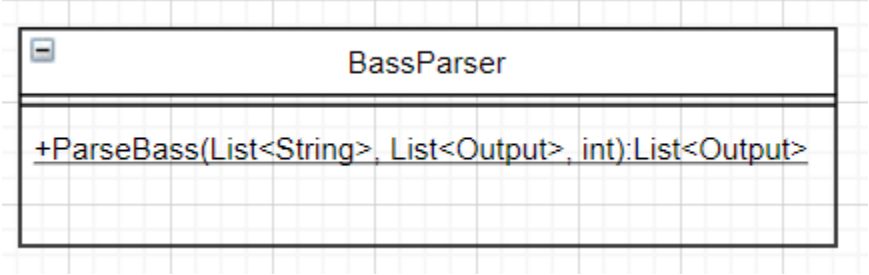
5.2.Main Class:

The main class parses any command line argument and initializes objects for the running of the program.



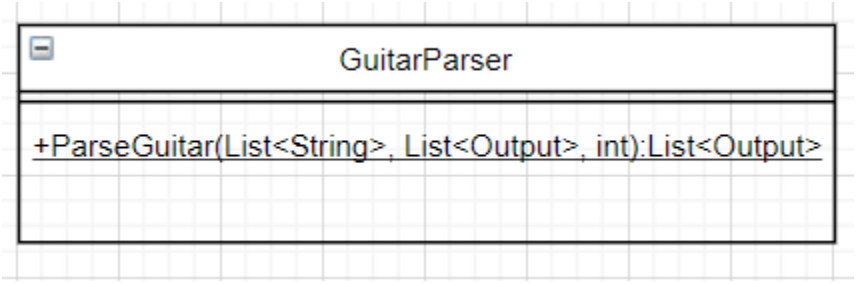
5.3.BassParser:

This class takes the imputed tablature to be parsed and the list of the results so far then makes changes and adds new elements to the list. It returns the result list of output.



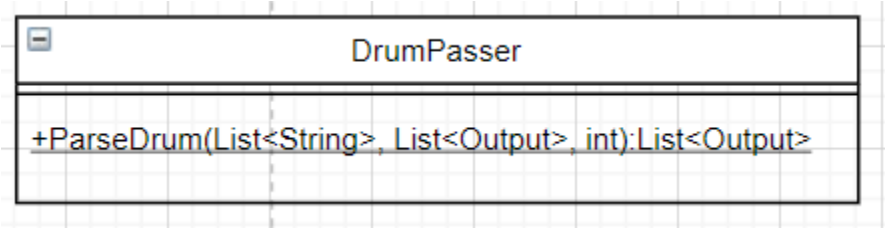
5.4.GuitarParser:

The GuitarParser reads the end of the file to check what type of tablature it is then takes the tab to parse and the list of the results so far. It then makes changes and adds new elements to the list, it returns the list of output.



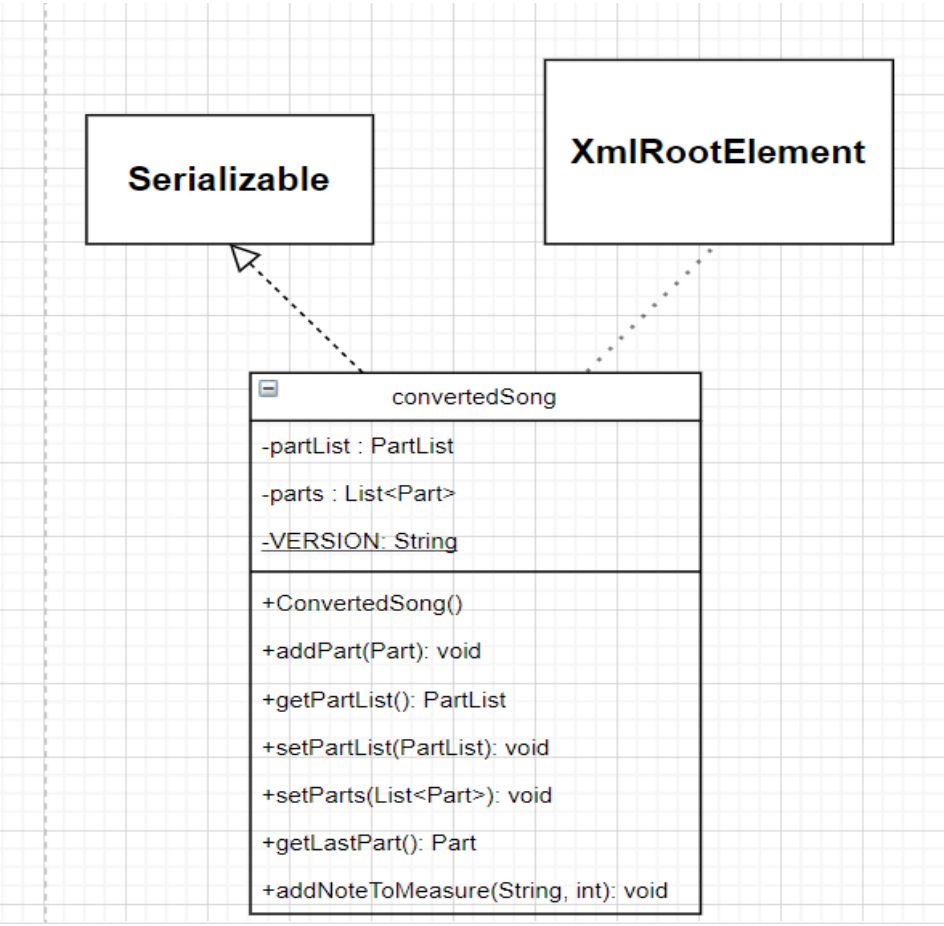
5.5.DrumParser:

The drumParser class takes the tablature to parse and the list of the results so far then makes changes and adds new elements to the list. It returns the result list of output.



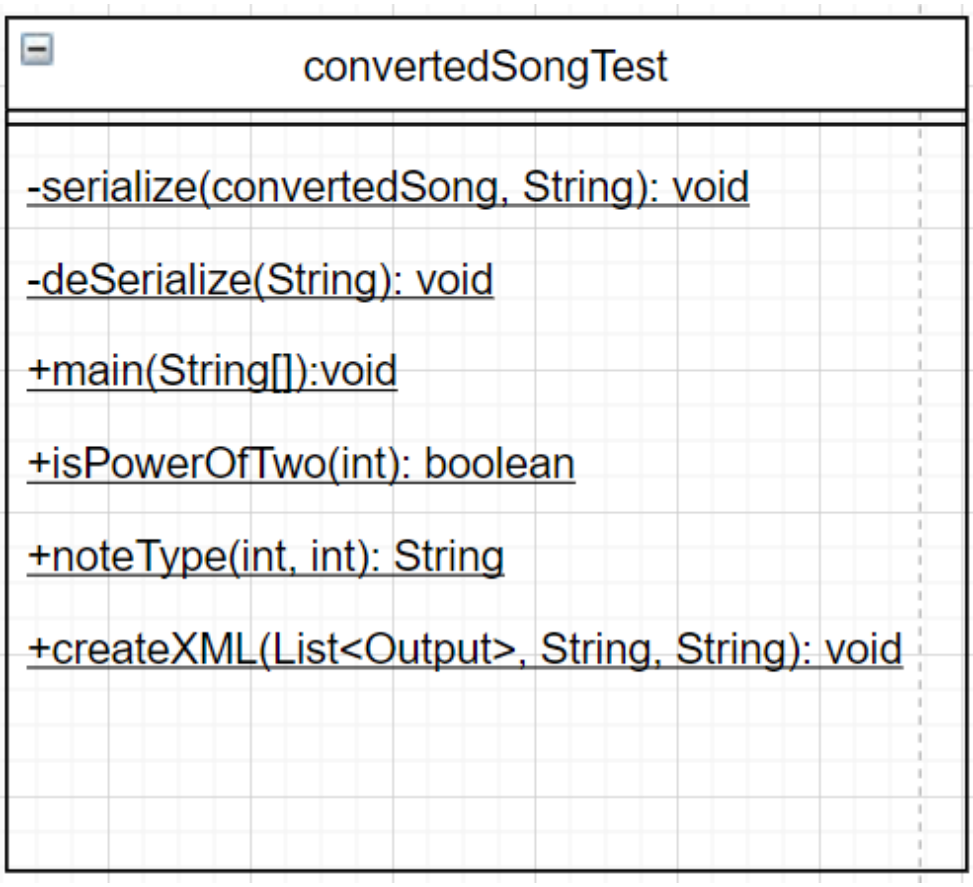
5.6.ConvertedSong:

The ConvertedSong class is the song object class that holds all the elements used for the XML file.



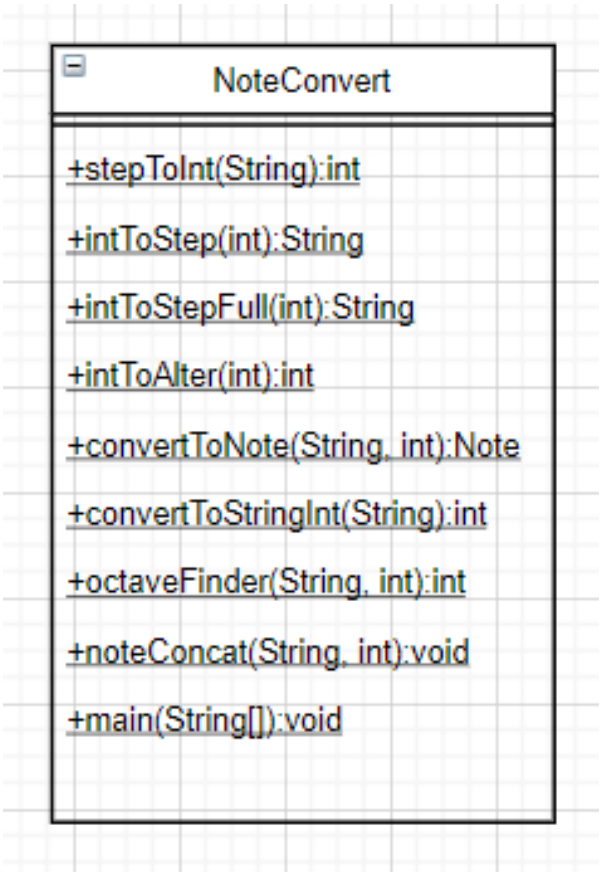
5.7.ConvertedSongTest:

All the objects are turned into tags in the convertedSong.java through marshaling. Jaxb was used to label each xml element and marshal each tag.

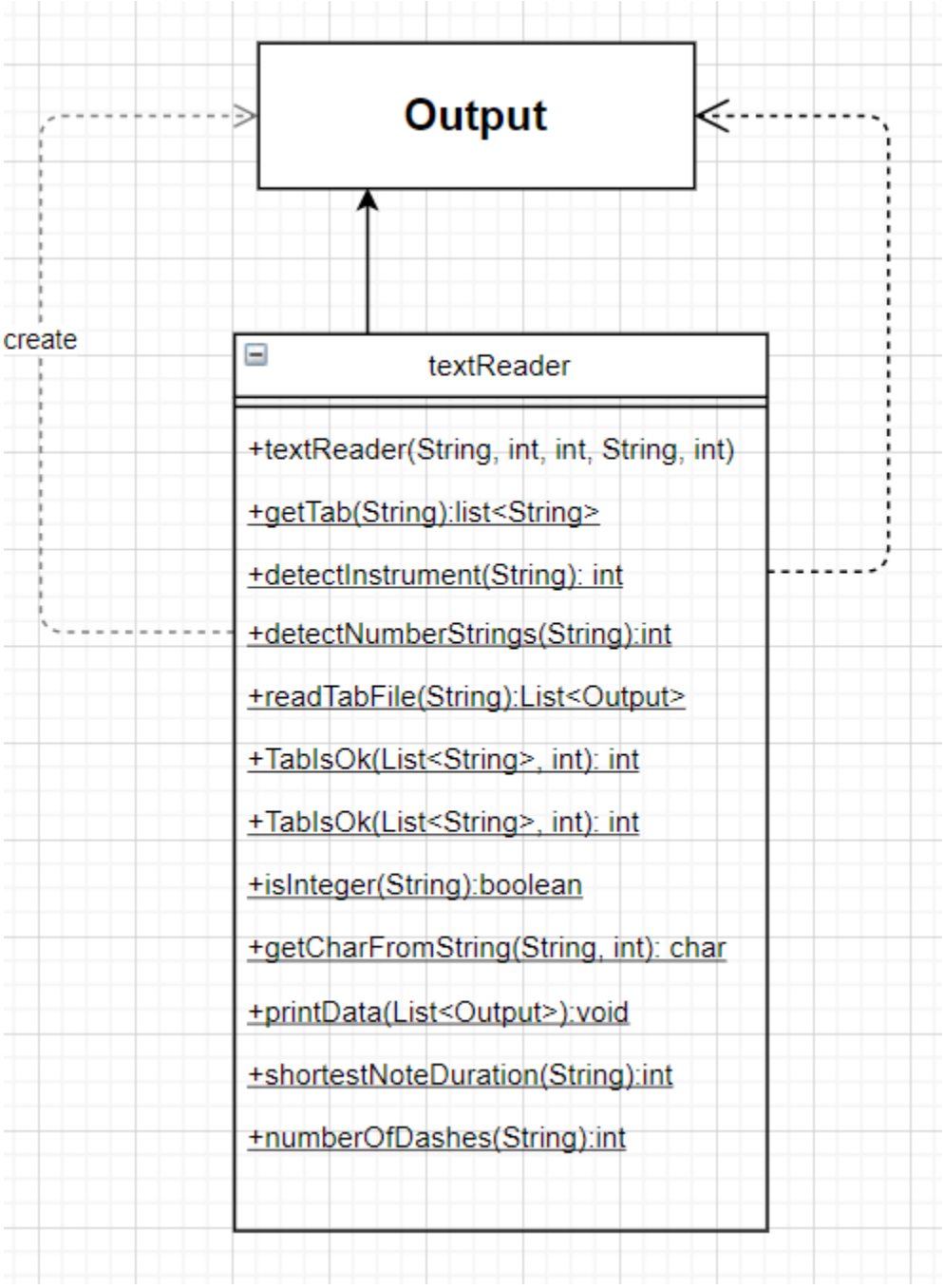


5.8.NoteConvert:

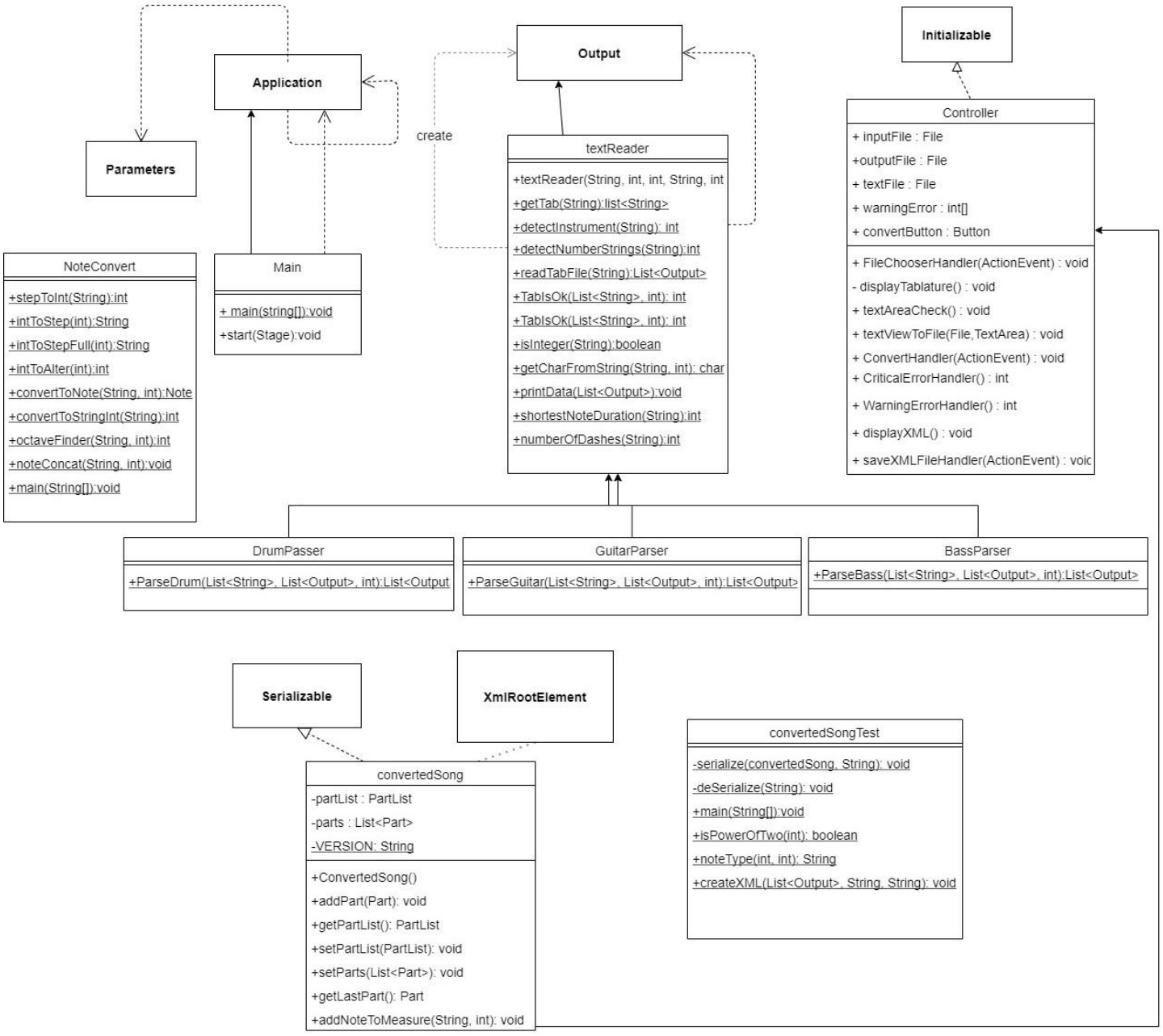
The NoteConverter class contains functions that are relevant to converting tablature into notes and their octaves


NoteConvert
+stepToInt(String):int
+intToStep(int):String
+intToStepFull(int):String
+intToAlter(int):int
+convertToNote(String, int):Note
+convertToStringInt(String):int
+octaveFinder(String, int):int
+noteConcat(String, int):void
+main(String[]):void

5.9.textReader:This is the class that parses tablature files given from the user



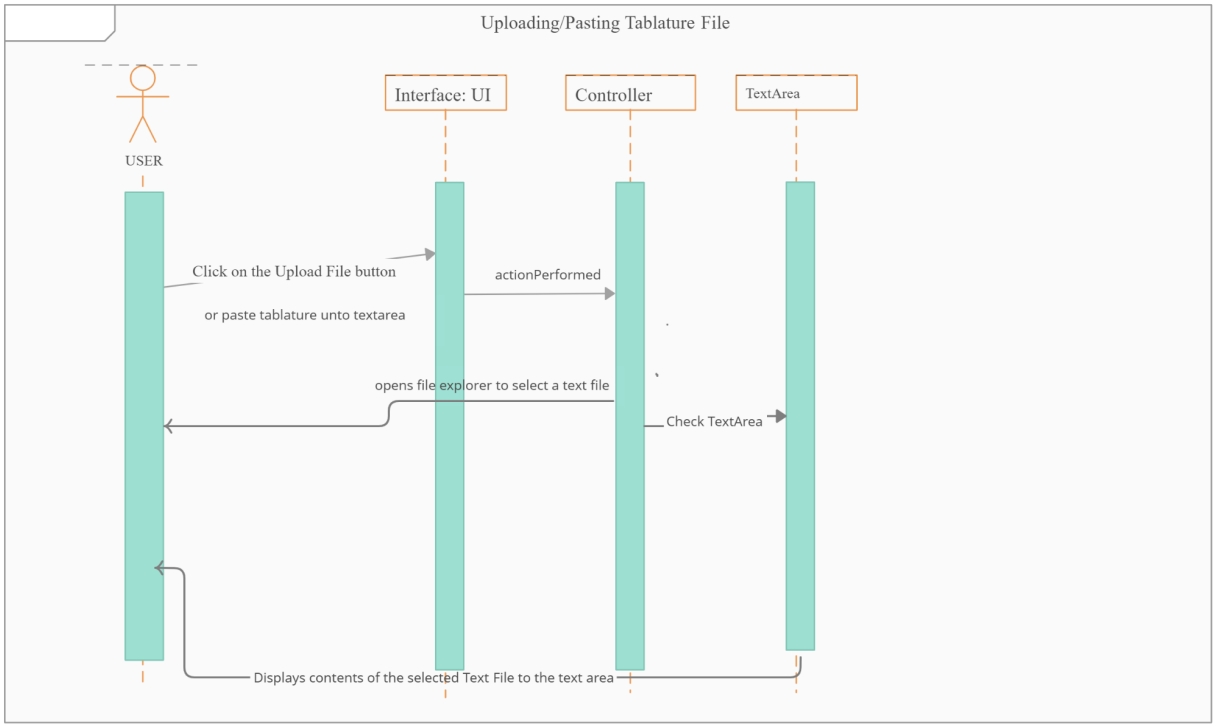
Class Diagram Overview



6.Sequence Diagram:

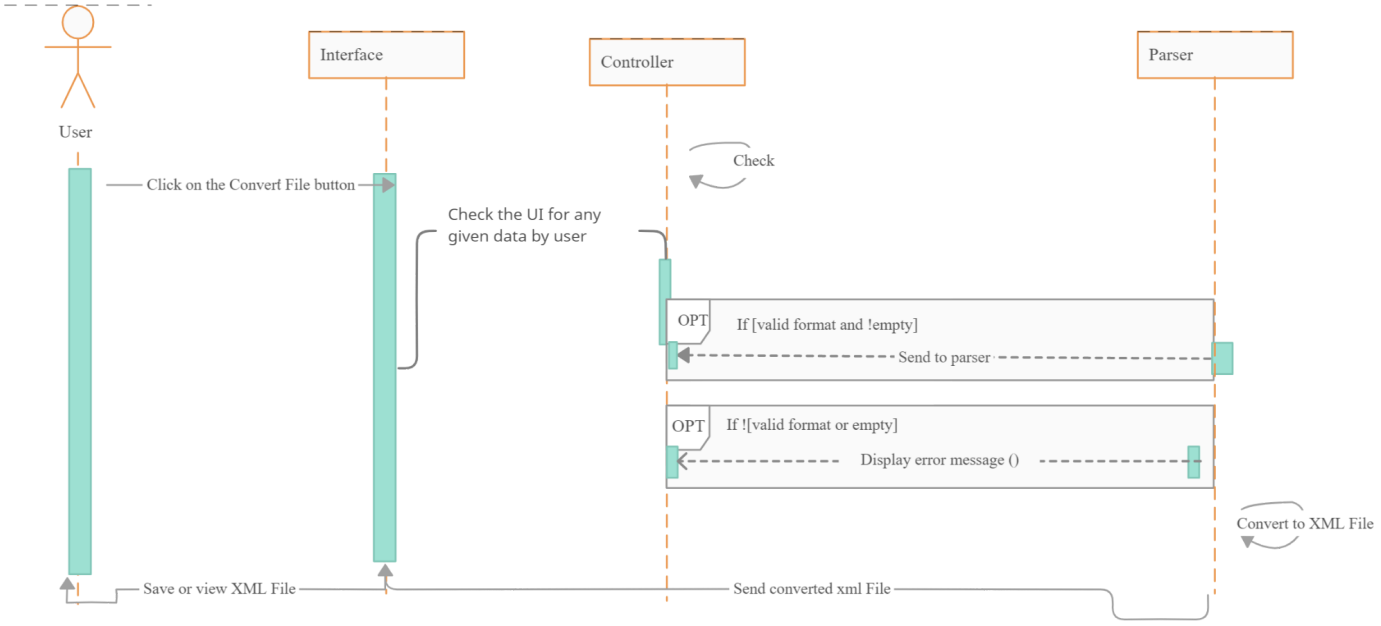
6.1.Upload/paste tablature file:

The user uploads or pastes a tablature file in the program to be converted



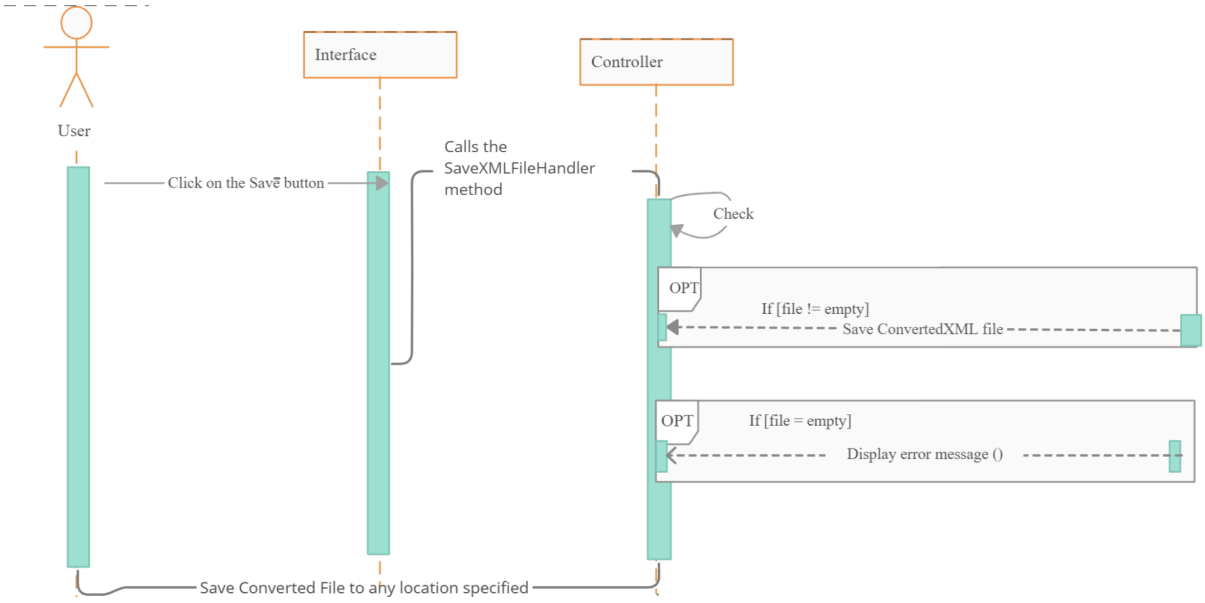
6.2.Convert To MusicXML:

The user clicks on the “Convert File” button to convert the imputed tablature file by the user into a MusicXML file.



6.3. Save ConvertedXML File:

The user clicks on the “save” button after conversion and chooses what location to save the convertedXML file.



7.Development Method

The development method used in this program is the Agile Development Method. This method was used in order to minimize bugs and changing requirements when adding new functionality. The primary benefit of agile software development is that it allows software to be released in iterations. Iterative releases improve efficiency by allowing the programming team to find, fix defects and align expectations early on. It also allows users to realize software benefits earlier, with frequent incremental improvements.

8.Maintenance Scenario

8.1 Adding Features

8.1.1 Adding Drum Features:

If any new drum feature was to be added into the program, changes would have to be made to the drumParser class

8.1.2 Adding Bass Features:

If any new bass feature was to be added into the program, changes would have to be made to the BassParser class

8.1.3 Adding Guitar Features:

If any new Guitar feature was to be added into the program, changes would have to be made to the GuitarParser class

8.1.4 Adding A New Instrument:

If any new instrument was to be added, a new parser class for that instrument would be made in order to parse that instrument's tablature. A new XML tag package for that instrument containing classes defining the functions of that instrument also has to be created. A new method under the textReader class has to be added in order to be able to detect that instrument when a user imputes the new instruments tablature