Design Document

ℹ️ **EECS 2311 W22 - Group 9** `v0.2`

Members:
- Vivek Wadhwani (*vivek121@my.yorku.ca*)
- Kris Singh (*ksingh7@my.yorku.ca*)
- Kingsley Okon (*King808@my.yorku.ca*)
- Lan Zhang (*zhalala8@my.yorku.ca*)

Project: github.com/devivekw/TAB2XML

Documentation Link

Table of Contents

1 Introduction

`TAB2XML` is Java/Gradle based tool that enables users to convert musical tablature from text to MusicXML (*an open source standard for exchanging digital sheet music*) and enables them to play their converted file as well as visualize their files. This document outlines how a step-by-step guide to install and use the system for common use cases.

1.1 Document's Purpose

The main purpose of this document is to describe the design being implemented in the `TAB2XML` software which relates to the requirements document. This document is intended to be read by developers who wish to understand or work on `TAB2XML`.

1.2 Document's Scope

This document goes over the design details in the `TAB2XML` software. It also includes UML Use Case Diagrams, Sequence Diagrams, Class Diagrams and an Activity Diagram. After reading this document you should be able to understand how the design was implemented in Java.

1.3 Tools & Technologies

- This document is written in **Java** using the `JRE v17`
- It is built and tested with **Gradle** `v7.3.3 or v7.1.1`. Dependencies Include:

    - `JUnit`
    - `JavaFX` & `TestFX`
    - `JFugue`
- The GUI is built upon the **JavaFX** `v15` framework and designed with **SceneBuilder.**

2 Design Choices

Prior to designing a solution, there are various factors need to be considered which may include but are not limited to robustness of the solutions, prior knowledge required, resources and time constraints.

2.1 Assumptions & Constraints

The user of `TAB2XML` must have a basic understanding of music, because of how the music notes are displayed as well as specific terminology used by musicians. The user must also know how to edit and manage the tablature in the first place that get converted into music and notes.
The main constraint behind this project is the time deadline we have been put with as well as the responsibility of other courses taken by all the students in the team.

2.2 System Goals & Guidelines

Our team's (Group 9) goal is to implement the play functionality as well displaying the music sheet for a given tablature with fairly little bugs by the due date. We also try to improve the general user experience with good error handling and informative feedback.

2.3 Development Methodology

The software our team is striving to build should be in essence intuitive and functional. Our team focuses on elegance, so as to provide the least amount of instructions needed for the user to use the system as they need.

The way our software is structured the user does not need a deep understanding of how our technology works instead requires a fairly base level understanding of music and its terminology.

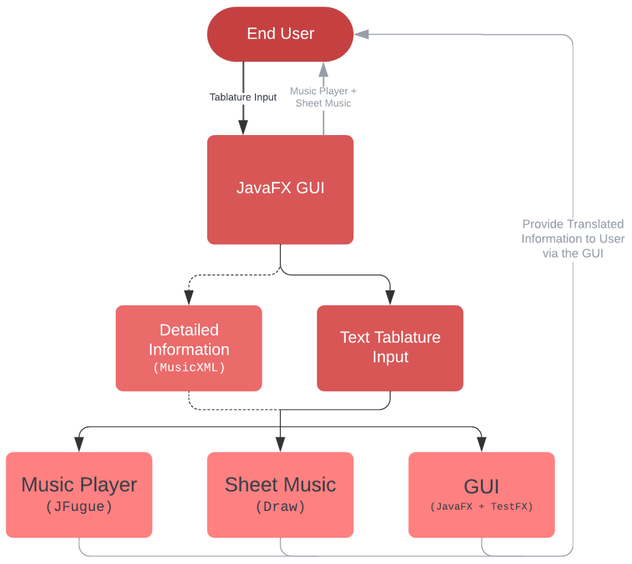

Figure 1: Development Methodology Diagram

3 System Design

3.1 Use Case Scenarios

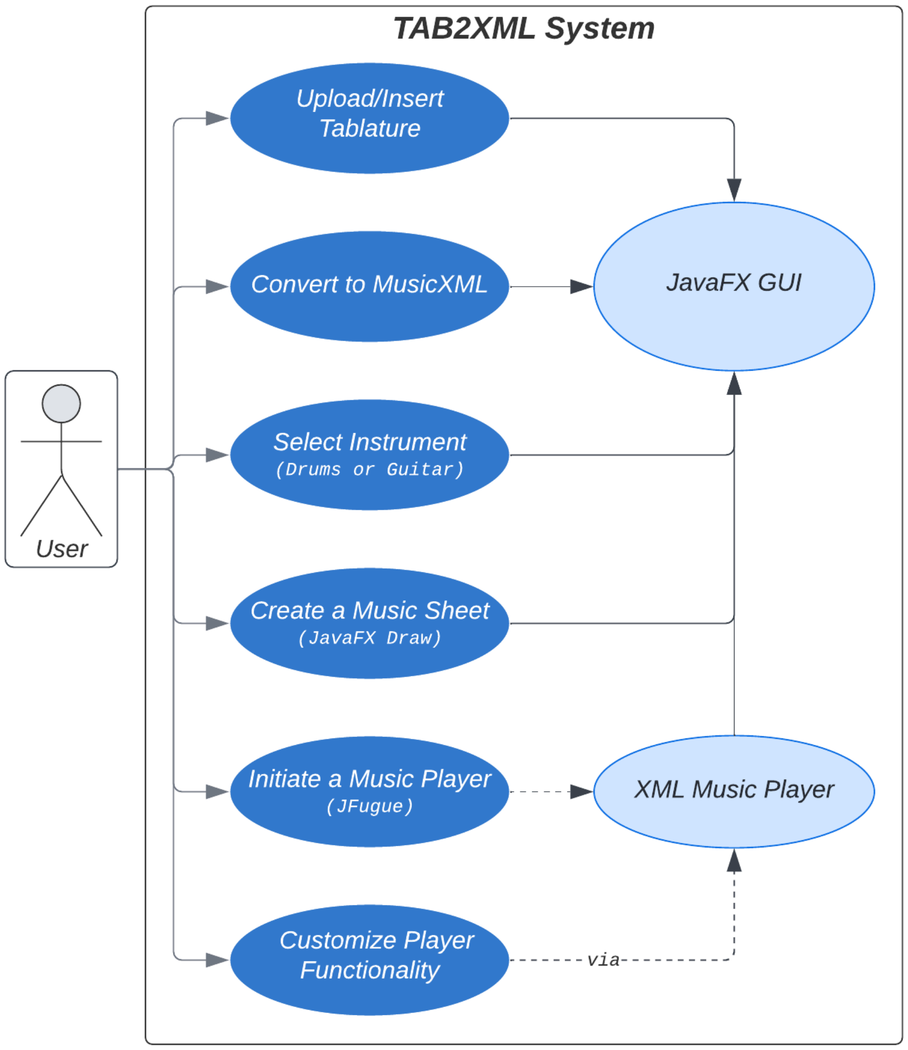
TAB2XML System
Upload/Insert Tablature
Convert to MusicXML
JavaFX GUI
User
Select Instrument (Drums or Guitar)
Create a Music Sheet (JavaFX Draw)
Initiate a Music Player (JFugue)
XML Music Player
Customize Player Functionality
via

Figure 2: Use Case Diagram

The end goal for all functionality its through the `JavaFX GUI`, which mandates all the scenarios for the user from the features all the way to the error handling. The GUI acts as a medium connecting the controller and the numerous utility classes (Sheet Music, Drawing, Music Player + `JFugue`) involved in parsing, playing and displaying the required information.

3.2 UML Class Diagrams

3.2.1 Music Player Class Diagram

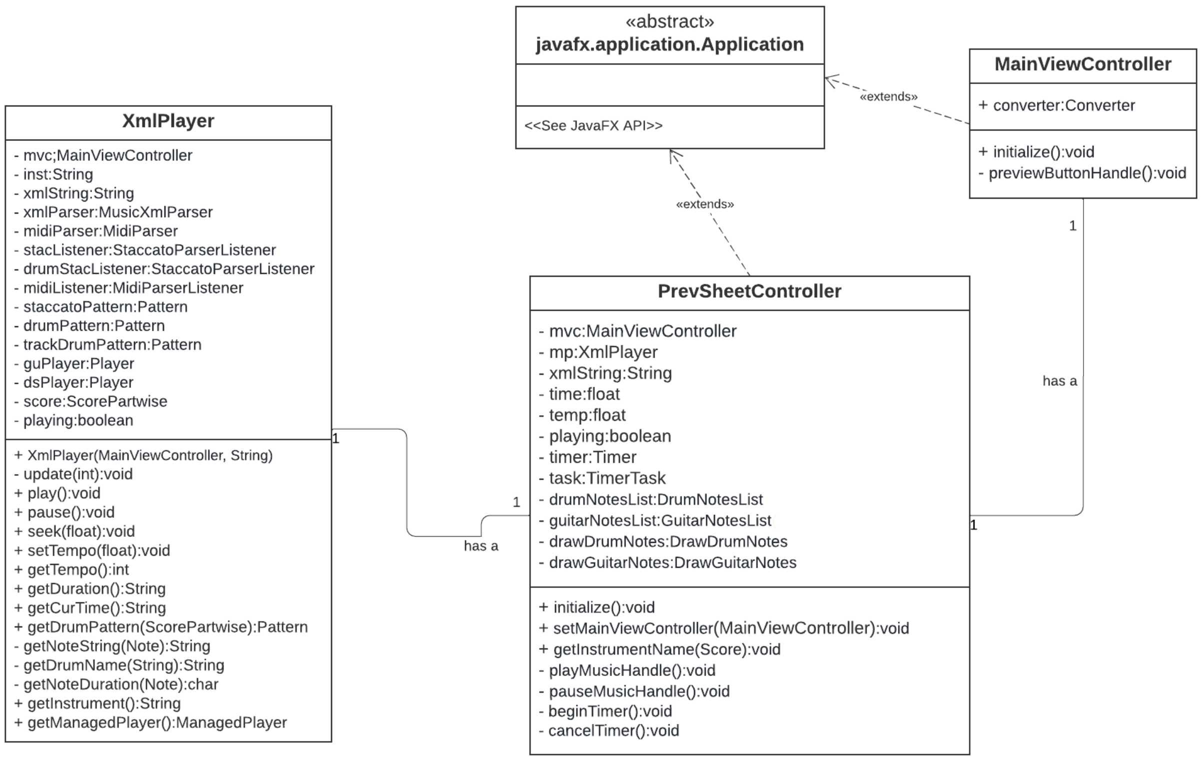

Figure 3: XMLPlayer UML Diagram

Each instance of the `PrevSheetController` class has an instance of the `XMLPlayer` class which it uses to play music from guitar and drum-set tablature which is accessed using the `MainViewController` attributes. The `XMLPlayer` class also provides methods to control and customize the music playback features such as play, pause, seek, and tempo.

3.2.2 Preview Sheet Class Diagrams

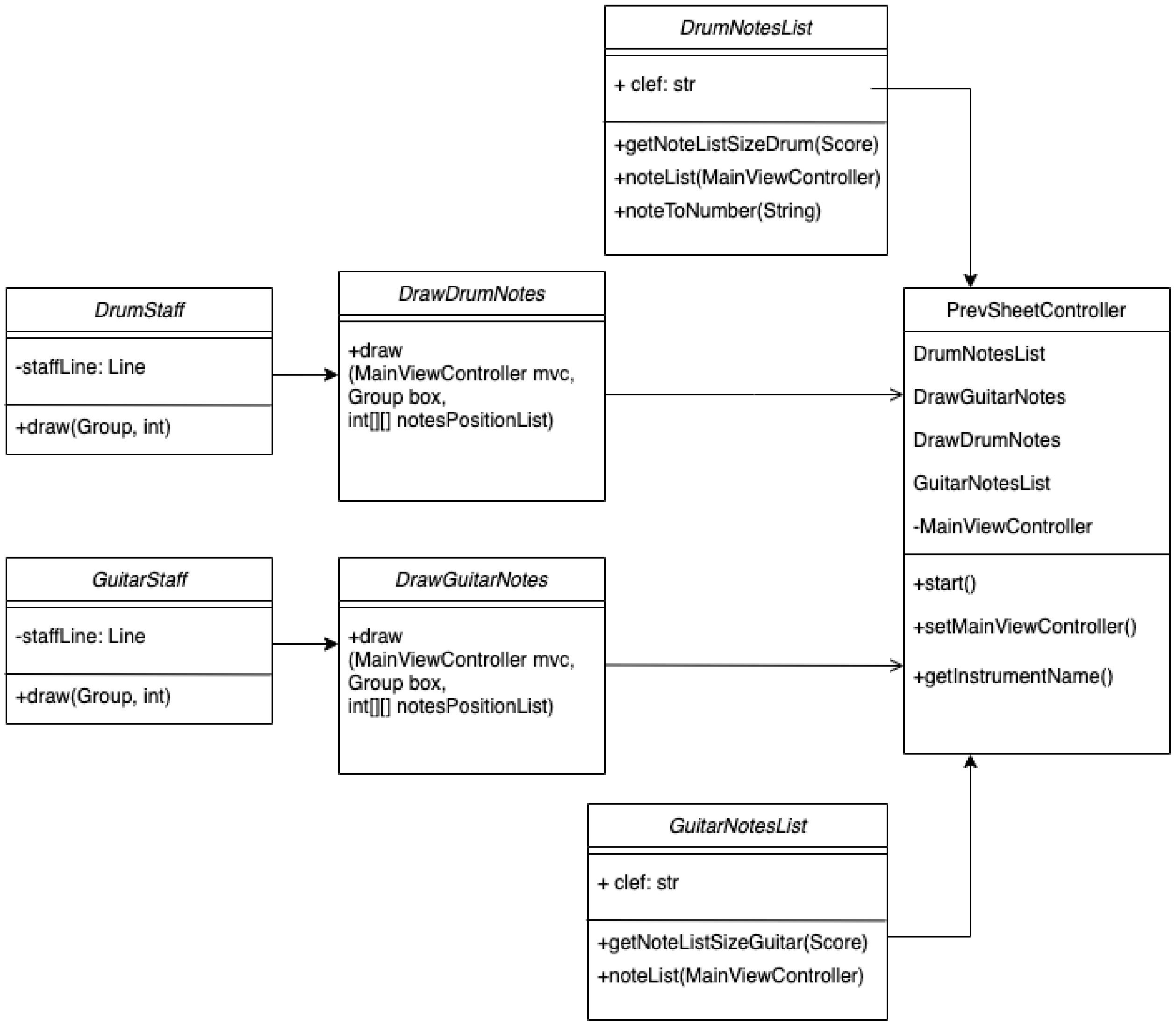

Figure 4: Score/Draw UML Diagram

`DrumNotesList` and `GuitarNotesList` take musicXML from `MainViewController` and extract individual notes information, i.e. Pitch, Unpitched, Chord, Notehead. These two classes return a sorted list of music note which is used by `DrawGuitarNotes` and `DrawDrumNotes` respectively to draw notes onto the Group object embedded in Scroll Pane. `DrawDrumNotes` and `DrawGuitarNotes` both have `DrumStaff` and `GuitarStaff` attributes respectively to add corresponding staff lines based on the size of the music sheet.

3.3 Sequence Diagrams

The two diagrams below show the flow of information and the hierarchy between the classes used in `TAB2XML` there are two diagrams. First one goes over the information flow in the Music Sheet and the second one goes over the information flow in the Music Player.

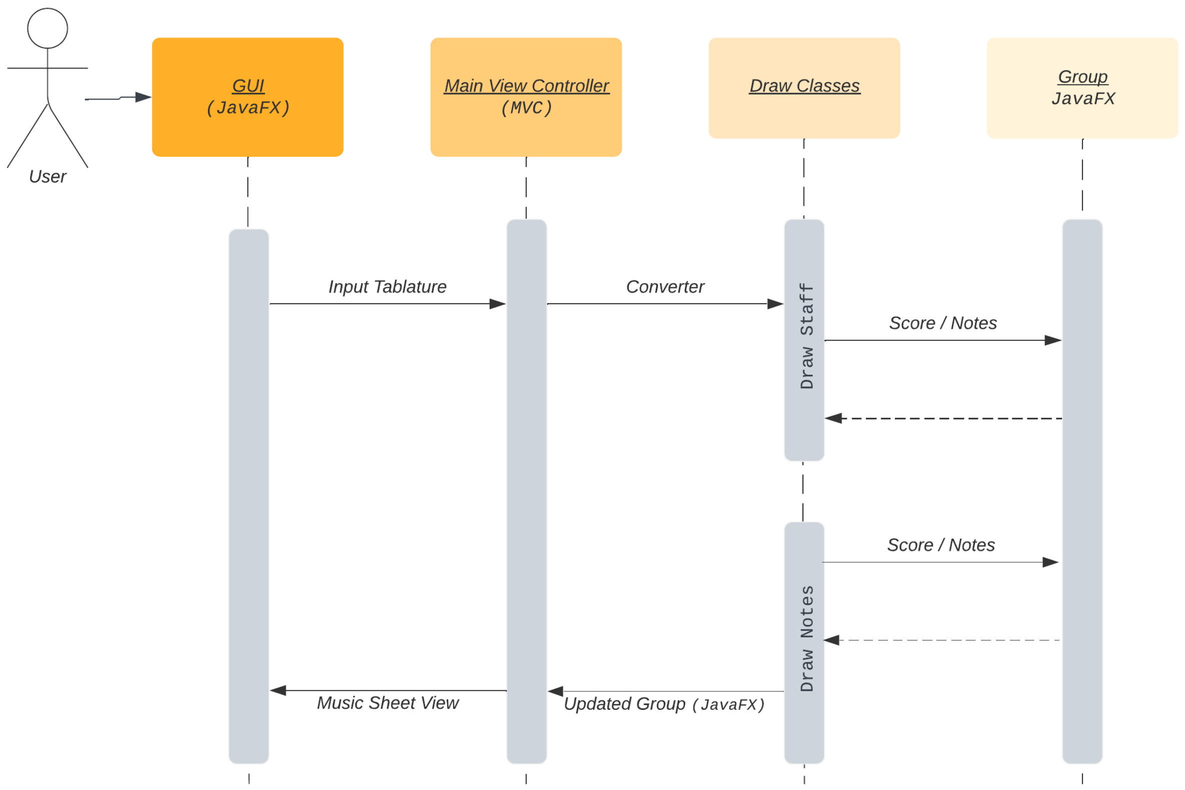

Figure 5: Music Sheet Sequence Diagram

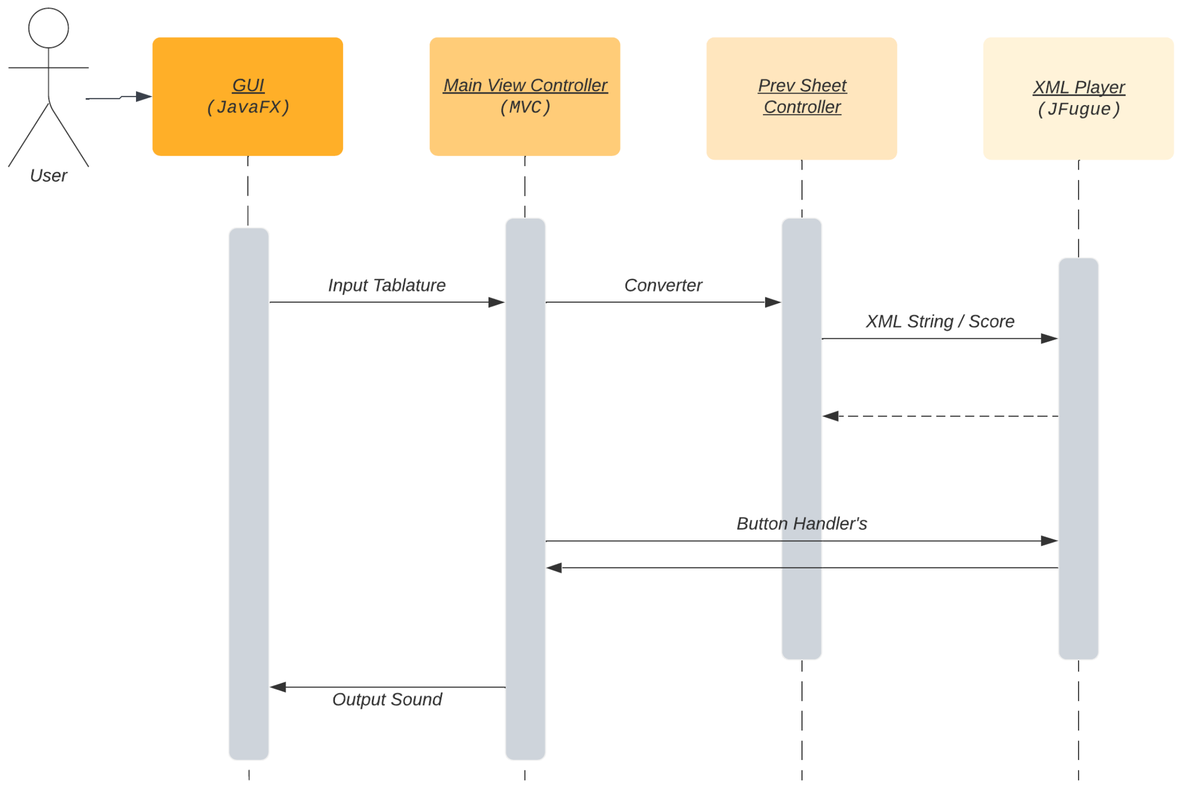

Figure 6: Music Player Sequence Diagram

4 System Architecture

This section includes in depth details about the main classes our team has introduced to the project. It goes over what these classes do and what they are being used for.

4.1 `PrevSheetController`

This class's main role is to interact with the `MainViewController` which in turn passes information over to the GUI written in `JavaFX` and designed in **SceneBuilder.**

Its other role is to work closely and initialize the classes that handle the Music Player and Drawing of the Music Sheet, these classes are discussed below.

4.2 `XMLPlayer`

Currently located under the `utility` folder, this class is handling the playing functionality of Guitar and Drum instruments.

The class is solely responsible for communicating with `JFugue` which is a library that enables writing programs that allow creating music easy. Since the library has a built-in Guitar instrument it just takes the `MusicXML` string as an input. But for the Drumset instrument, this class has a function `getDrumPattern` which generates a pattern that is understood by the library.

This class also handles all the functionality related to the player in the GUI such as play, pause, volume, seek, and tempo. This is handled via various get and set functions that in turn call `JFugue`.

4.3 `Draw/Score/`

This is folder currently contains six classes that are responsible for drawing the Music Sheet the main essence behind these classes is to handle various parts of the sheet that range from the staffs, clefs, and various notes.

The `DrumStaff` and `GuitarStaff` mainly handle drawing the staff's for those specific instruments on to the `JavaFX Group Object`.

The `DrumNotesList` and `GuitarNotesList` handle parsing the information from the `MVC Score` to an internal format of a 2D array that is understood by the following classes. *(We are planning on changing the format of the 2D array to a custom class)*

The `DrawDrumNotes` and `DrawGuitarNotes` are the main classes that handle the drawing of the various notes and the details regarding them onto the `JavaFX Group and Scene`. This is done by the parsed 2D array that includes x, y positions and the type of the note to be drawn.

5 Maintenance Scenarios

5.1 Music Player with Music Sheet

To enable a more fluid workflow for the user, if the Music Sheet were able to flow with the music being played it would greatly improve the user experience. This can be managed and done within the `PrevSheetController` since there is the main focal point of the application.

This can be seen in various web applications where the Music Sheet is highlighted when it is being played as well as allowing the playback functionality of a selected section on the music sheet.

5.2 Add More Note elements to sheet music view

The structure of the `draw.score` package enables developers who may be interested in adding more visual note elements to the sheet music view this can be easily implemented by modifying either the `DrawDrumNotes` or the `DrawGuitarNotes` class and adding methods to support these additional note elements such as the existing `drawSlur` and `drawGrace` methods, then finally calling them in the `drawEverything` method.

5.3 Add new Drum sounds to Drumset

We have added the ability to easily add more drum sounds to the drumset played by the Music Player by simply editing the `getDrumName` method in the `XMLPlayer` class and adding another condition specifying the `Jfugue` instrument constant for the appropriate instrument ID.

6 Error Handling

Since most of the client-side error handling was already presented in the starter code, this section reviews the main errors being handled internally.

6.1 Internal Errors

1. `InvalidMidiDataException` - This Exception is thrown when an error occurs in which the data in the MIDI object of our player is corrupted and inappropriate MIDI data is encountered during music playback

2. `MidiUnavailableException` - This Exception is thrown when an error occurs in which the MIDI component cannot be created because it is unavailable so the music cannot be played.
3. `.getModel()==null` in the scenario when `.getModel()` is null, it’s excluded by an if statement so it does not cause error while running the App.