Software Design Document

## CSE2311

## Prepared for: Vassilios Tzerpos

Group 3

**By:** Mohamed Nasar, Calvin Tran, Jason Kuffaur, Muhammad Shah,

Siraj Rauf, Waleed Azhar, Umer Zahoor

# Introduction

The purpose of this software design document is to provide a low-level description of the TAB2PDF system, providing insight into the structure and design of each component. Topics covered include the following:

* + Class hierarchies and interactions
  + Data flow and design
  + Processing narratives
  + Maintenance scenarios

In short, this document is meant to equip the reader with a solid understanding of the inner workings of the TAB2PDF system.

Goals

The purpose of TAB2PDF is to facilitate the conversion of Guitar tabs files to a stylized PDF. The final product must be quick, efficient, and extremely easy to use. It must offer useful features without overwhelming the user with options. The user interface must be intuitive and have little or no learning curve. Beyond these general design principles, the application must also provide the following concrete functionalities:

Group 3

* + Read and Parse guitar tabs
  + Create PDF Music Sheet of the tabs
  + Allow User to control style options, such as font, margins, spacing etc
  + Provide a preview of the output

# Design Overview

Introduction

This section is dedicated to giving a brief overview of the program and its basic structure. After reading this section the reader should be familiar with the main classes of the program as well as understand how they interact.

Overview

Broadly speaking, the program’s main use was to translate an ASCII guitar tablature to PDF. Guitar tablatures consist of numbers and symbols representing frets, arranged in six rows which together are called measures. ASCII versions generally use dashes as space delimiting characters, and vertical lines as breaks. Measures are combined into lines called staffs. A staff is a set of measures filling an entire line which depending on the spacing can be anywhere from a single measure to multiple measures.

This in mind, the back-end of the program was focused upon the actual data, which was kept in the Tablature class. This class contained a 2D array with the tablature data as well as fields for the title and subtitle, as well as the updated title and subtitle should the user decide to change it. Another class, Style, was created to take care of all the styling/visual requirements of the tablature. This included spacing between the numbers, lines and measures as well as the font face and individual font sizes for the title, subtitle and data. This is the class that would be modified through the GUI as the user changed the different values to suit their needs to make the tablature more visually appealing.

To input the data, a special class called parser was created that will input a file and then scan the file line by line. This class will compare the scanned lines to preset regular expressions and create a new tablature object containing the data. During this process, it should also replace special characters with easier-to-process characters for output later as well as give default values for any missing fields. This class will also be responsible for identifying problems with the input and logging them appropriately.

A separate class is used to output the actual user data (including the array, styling changes, etc.) into an actual readable PDF. This class, called PDFOutputCreator, takes a tablature object as well as a style object and creates the PDF using the data from tablature, and formatting it according to the fields in style. This class contains the methods for replacing the special symbols as well as drawing any symbols and data on the PDF.

The back end of the program would be hidden behind a graphical user interface created by the UserInterface class. The user would interact only with this class consisting of buttons, sliders and text fields. This class would call upon various functions/classes of the back end depending on the user input. This is also the class that handles the final output, as PDFOutputCreater only creates a temporary PDF (named “temp.pdf”) that UserInterface will them move and rename accordingly. Any errors or warnings are also handled by this class, generally displayed in dialog boxes, e.g. File cannot be saved, etc. prompting the user to take action.

Overview

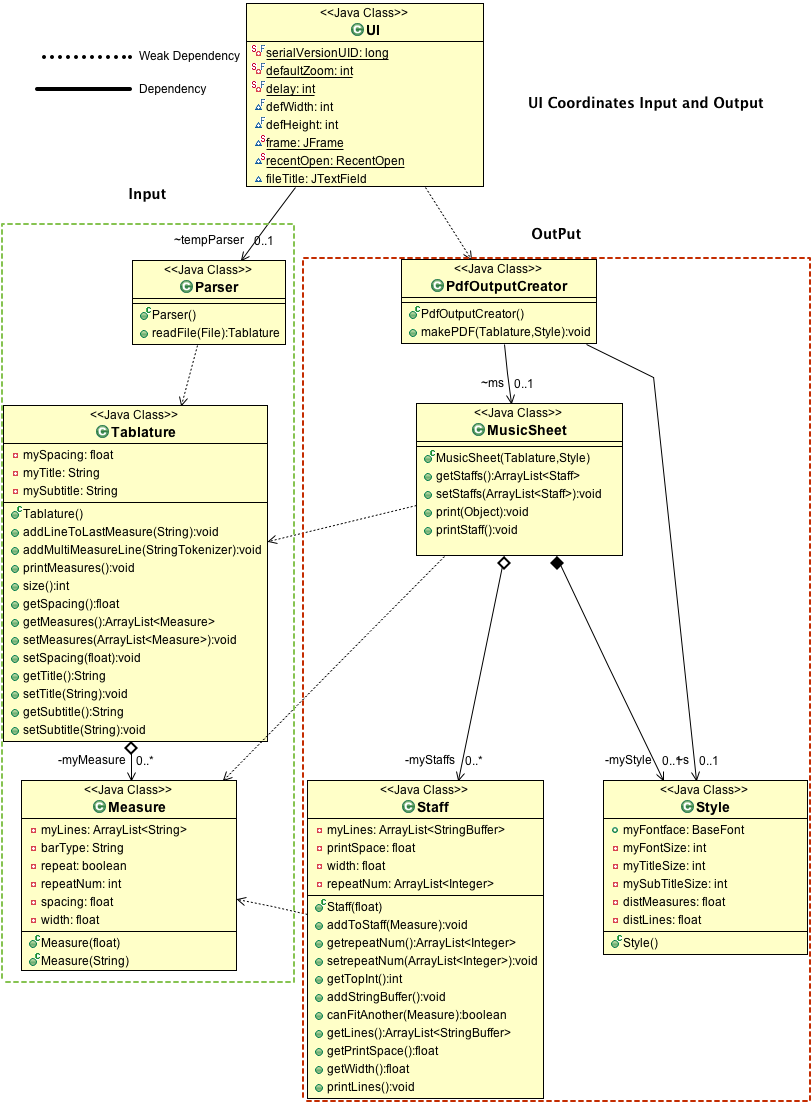
Diagram 1.1 UML Overview

Diagram 1.2 Sequence Diagram

Group 3

# individual components

Group 3

| Measure Class | | | |
| --- | --- | --- | --- |
| Important Methods | addLine(String); | corrections(); |  |
| Important Variables | ArrayList<String> myLines | float width | int repeatNum |
| Role(Data) | Hold and manage lines in a Measure. It is used by Tablauture Class. | | |
| Summary | Measure are sent strings,which are added as a node In the array list.It limits the size of the array to 6 nodes. This class checks the bar types on each end,calculates the width of this measure will be on a pdf page and if it needs a repeat flag.When it is filled, it calls the corrections method that will check the 6 line for equal length. | | |

| Staff Class | | | |
| --- | --- | --- | --- |
| Prameters | float printSpace - this max width of Staff | | |
| Important Methods | addToStaff(Measure); | barsAdded(String); | canFitAnother(Measure); |
| Important Variables | float printSpace | float width |  |
| Role(Data) | This class holds sets of measures that will be printed to PDF in a single line. It is used by MusicSheet Class | | |
| Summary | This Class holds measures that fit in a single Staff. The class is given a printSpace, which is the max width of the staff, in its constructer. The client can tell whether it has room by using the canFitAnother(..) method, then call addToStaff(..). It also checks that the bars on every measure are aligned correctly, and removes mistakes with barsAdded(..). | | |

| Style Class | | | |
| --- | --- | --- | --- |
| Important Methods | getPrintSpace() | getWidth(char char1) | getHeight() |
| Important Variables | BaseFont myFontface | int myFontSize | float distMeasures |
|  | float distLines | float leftMargin | float rightMargin |
| Role(Data) | Used by UI to pass information about user Layout choices to PdfOutputCreator class. It also does layout related calculations. | | |
| Summary | This class is mostly information but preforms 3 important calculations.  First printSpace, this the amour of width available to a staff based on users margins. Second getwidth(..) calculates the width of a specific char for the font and font size selected by the user. lastly getHeight calculates the height of the font based on font and font size. | | |

| Tablature Class | | | |
| --- | --- | --- | --- |
| Important Methods | addLineToLastMeasure(String) | addMultiMeasureLine(StringTokenizer) | |
| Important Variables | ArrayList<Measure> myMeasure | float mySpacing | String myTitle |
|  | String mySubtitle |  |  |
| Role(Data Handler) | It used to construct a ‘Tablature’. Holds all info in text file for program to manipulate. It is used by the Parser class. | | |
| Summary | This class is a data structure for a tablature. It holds data in an ArrayList of measures. When addLineToLastMeasure(..) is called it will add the passed string to the last measure in the array. It will do so till the Measure is filled then add a new measure to the array. It had be passed String that contain multiple measures with addMultiMeasureLine(..). In this case it will make the add the necessary amount of Measure to the ArrayList, and add the tokens, from the passed StringTokenizer, to the correct Measure. | | |

| MusicSheet Class | | | |
| --- | --- | --- | --- |
| Parameters | Tablature tab - the data | Style s - the formatting |  |
| Important Methods | private makeStaffs (Tablature) |  |  |
| Important Variables | ArrayList<Staff> myStaffs | Style myStyle |  |
| Role(Data Handler) | To arrange data in Tablature to match the users layout based on Style. | | |
| Summary | This class takes a Tablature object and arranges the data meet the layout. It does it by constructing Staff objects from the measures in Tablature. It then holds the Staffs in myStaffs ArrayList. It appends to the last staff on the array list till filled , then it makes a new Staff and begins appending to it. | | |

| PdfOutputCreator Class | | |
| --- | --- | --- |
| Important Methods | makePDF(Tablature,Style) | Various Drawing Methods |
| Role | To make PDF file from Tablature and Style. | |
| Summary | The classed is used for it makePDF(..) method. The method uses the tab and styles to make a MusicSheet. It then takes the data from the MusicSheet to write to the pdf.  The makePDf method loops thorough every staff in MusicSheet, then loops through every line in the staff, then loops through every char in the string. It checks which type of symbol each char is and calls the appropriate method. For some symbols, such as fret numbers, it also checks the next char in the sting to check for special case, like double digit frets.  The Writing to the pdf is done in block units. These blocks width is the spacing variable from the text file or the user choses in the UI, the height is the line spacing from the style. It is important to understand that the horizontal lines are just dashes that touch on both ends, the bars are short vertical that touch. Both give the appearance of drawing full lines but really aren't. This approach cuts down the complexity since all the formatting can be done before this class is used.  This class also has many private methods that handle all the drawing of each type of symbol. | |

| FontSelector Class | | | |
| --- | --- | --- | --- |
| Important Methods | getFont(int); |  |  |
| Important Variables | String Fonts[] |  |  |
| Role | To manage fonts and map font to iText compatible Class. Used by UI | | |
| Summary | This class is used to manage the fonts for the back end. It takes font files and converts them in to a BaseFont that can be used by PdfOutCreator.It also keeps an array of available Fonts used by the UI to display choices to user. | | |

| Parser Class | | | |
| --- | --- | --- | --- |
| Important Methods | subsituteSymbols  (String sections) | readFile(File file) | readHeader() |
| Important Varibles | String acceptedSymbolsRegex | String correctLineRegex | String titleRegex |
|  | String subtitleRegex | String spacingRegex | String measureSeparatorsRegex |
| Role | This class reads the file containing the Tablature and parses it. Used by UI | | |
| Summary | This class reads the file file containing the Tablature and parses it.  It returns a Tablature object to that contains the parsed info in the file.  It reads each line in the file and compares to the REGEX contained in the variables.  Checking to see if   * It’s an Empty Line * it is at least 4 chars long * it contains at least one of ‘|’ or ‘-‘ * there're any leading or hanging text that is not tab info and remove it * if it ,after passing all above the, only contains accepted symbols   The lines that make it are tokenized with measureSeparatorsRegex as the delimiters. Depending on the amount of tokens the appropriate methods is call on Tablature to insert the line. | | |

# Maintenance

TAB2PDF will likely require very little Maintenance, This is because it uses very few custom objects, and relies mostly on Objects provided by Java. The UI is made mostly of default swing objects which are not likely to change much over time. The same is true for the back-end, which uses ArrayList for data structures etc. These classes are not likely to chafe much over time.

PdfOutCreator is most likely to need maintenance. Since it is still being developed, the api may change and some important drawing functions may change with it. To keep using the latest version may require rewriting some of the drawing methods in this class.