

**TAB2XML**

Design Document

**Group 6**

**Elmira Onagh**

**Irsa Nasir**

**Long Lin**

**Harjap Randhawa**

**Daniel Di Giovanni**



Winter 2022

**Table of Content**

[**1.** **Visualization of MusicXML** 2](#_Toc99491845)

[**1.1.** **Instrument: *Guitar*** 3](#_Toc99491846)

[**1.1.1** **Sequence Diagram** 3](#_Toc99491847)

[**1.1.2** **UML Class diagram** 7](#_Toc99491848)

[**1.2.** **Instrument: *Drum*** 8](#_Toc99491849)

[**1.3.** **Instrument: *Bass*** 10](#_Toc99491850)

[**2.** **Playing the tablature** 11](#_Toc99491851)

[**2.1.** **Activity Diagram** 11](#_Toc99491852)

[**2.2.** **UML class diagram: *Guitar*** 12](#_Toc99491853)

[**2.3.** **UML class diagram: *Drum*** 12](#_Toc99491854)

[**2.4.** **UML class diagram: *Bass*** 13](#_Toc99491855)

[**3.** **Printing the Music sheet** 14](#_Toc99491856)

[**3.1.** **Sequence Diagram** 14](#_Toc99491857)

[**3.2.** **Activity Diagram** 15](#_Toc99491858)

[**4.** **Go to measure** 16](#_Toc99491859)

[**4.1.** **[Name] Diagram** 16](#_Toc99491860)

# **Visualization of MusicXML**

Based on the instrument specified in the input tablature, three different objects are created: *Guitar*, *Drum*, and *Bass*. In the following section, we look at different diagrams related to the creation and relationships of these classes.

Diagram

Description automatically generated

**Figure 1.** Overall activity diagram of the visualizing a tablature as a music sheet.

### **Instrument: *Guitar***

If the input tablature is a guitar tablature, then a *Guitar* class is instantiated. The creation and displaying of the elements of the tablature in form of a music sheet are done through the *drawGuitar* method.

In section 1.1.1 the sequence diagrams depicting the sequence of events taken to visualize guitar notes can be found. The Diagrams are broken down into parts to ease the understanding and visualization.

In section 1.1.2 UML Class diagram is included to show the interactions between the *Guitar* class and other classes that result in displaying the musical elements on the screen.

#### **Sequence Diagram**

Diagram, text

Description automatically generated

**Figure 2.** Overall Sequence diagram of displaying guitar tablature.

Graphical user interface

Description automatically generated

**Figure 3.** Sequence diagram describing the events taken in *drawGuitar* method of *Guitar* class.

Diagram

Description automatically generated

**Figure 4.** part A dictates the events that have taken place inside a method that draws the notes inside a given *Measure* object. For each given *Note* object in the *Measure*, if a note has a technical tag, then *drawNoteWithTechnical* method is used to draw/ display the note. Part B shows events taking place inside the previously mentioned method (*drawNoteWithTechnical*) and indicates which methods are used based on whether the given note has a chord and/or grace tag respectively.

Diagram

Description automatically generated

**Figure 5**. The sequence diagram of the four self-calls is depicted in figure 4. B in more details and their interaction with *DrawNote* class to display their respective notes on the screen. Diagram A indicates the events that result in grace notes, Diagram B, the ones that result in regular notes, diagram C, ones that result in grace chords and finally diagram D the ones that result in regular chords being displayed on the screen.

Graphical user interface

Description automatically generated with medium confidence

**Figure 6.** Detailed sequence diagram of *drawType* self-call on *Guitar* class.

Diagram

Description automatically generated

**Figure 7.** Detailed sequence diagram of *drawBend* (A) and *drawDot*(B) self-calls mentioned in the previous diagrams.

#### **UML Class diagram**

A screenshot of a computer

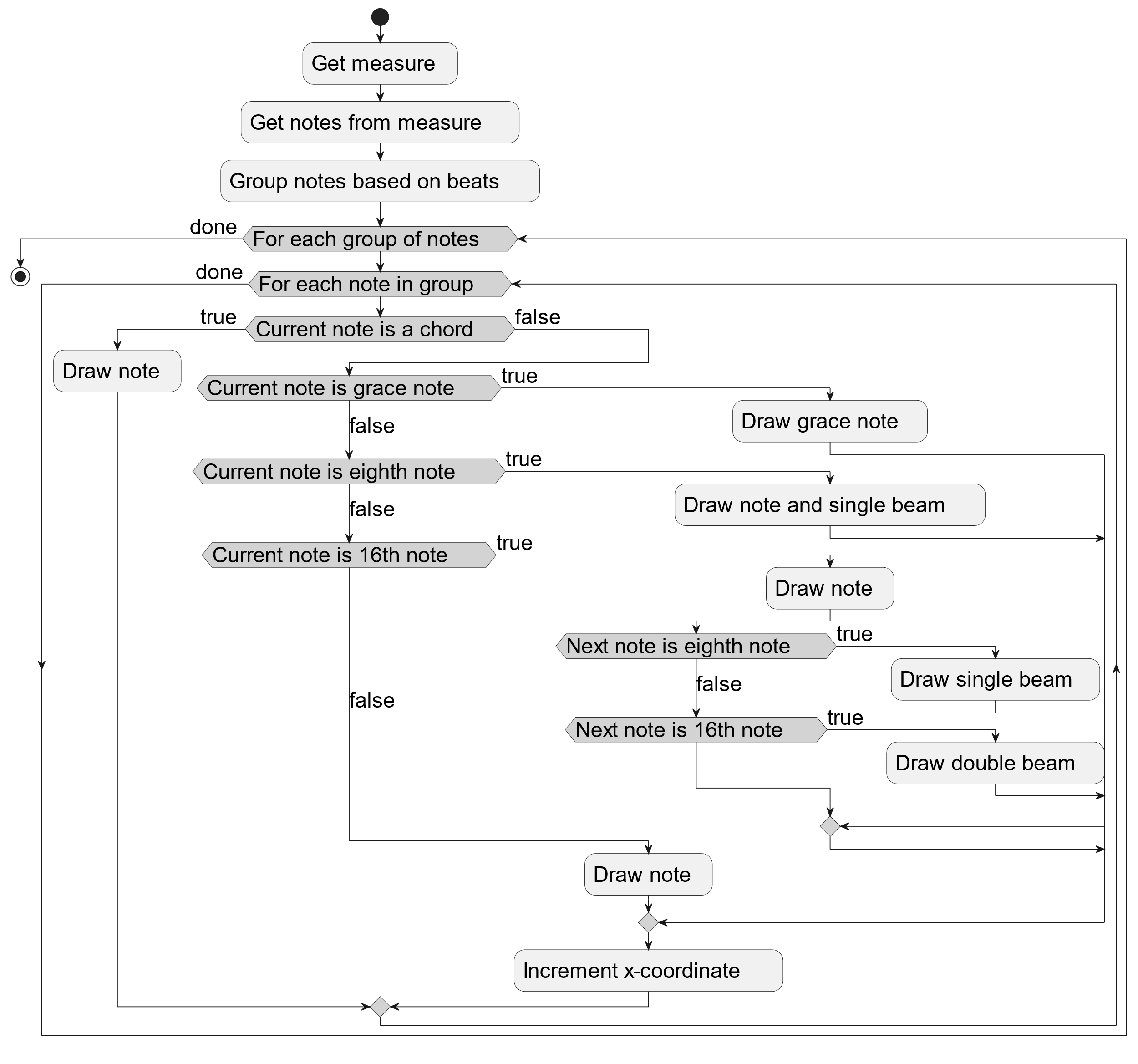
Description automatically generated with medium confidence

**Figure 8.** Class diagram of the *Guitar* class and its interactions. The green coloured *Guitar* class belongs to the *instrument* package while the blue classes belong to the *GUI.draw* package. The public, private, protected attributes and operations are denoted by “+”, “-”, and “#” respectively.

### **Instrument: *Drum***



**Figure #**. description



**Figure #**. description



**Figure #.** description

### **Instrument: *Bass***

# **Playing the tablature**

In this section, we will cover the design diagrams related to the Play functionality of the system. In section 2.1 an Activity diagram describing the overall events taken to play the notes is included. In the following sections, UML class diagrams depicting the specific methods of *MusicPlayer* class used for each instrument can be found.

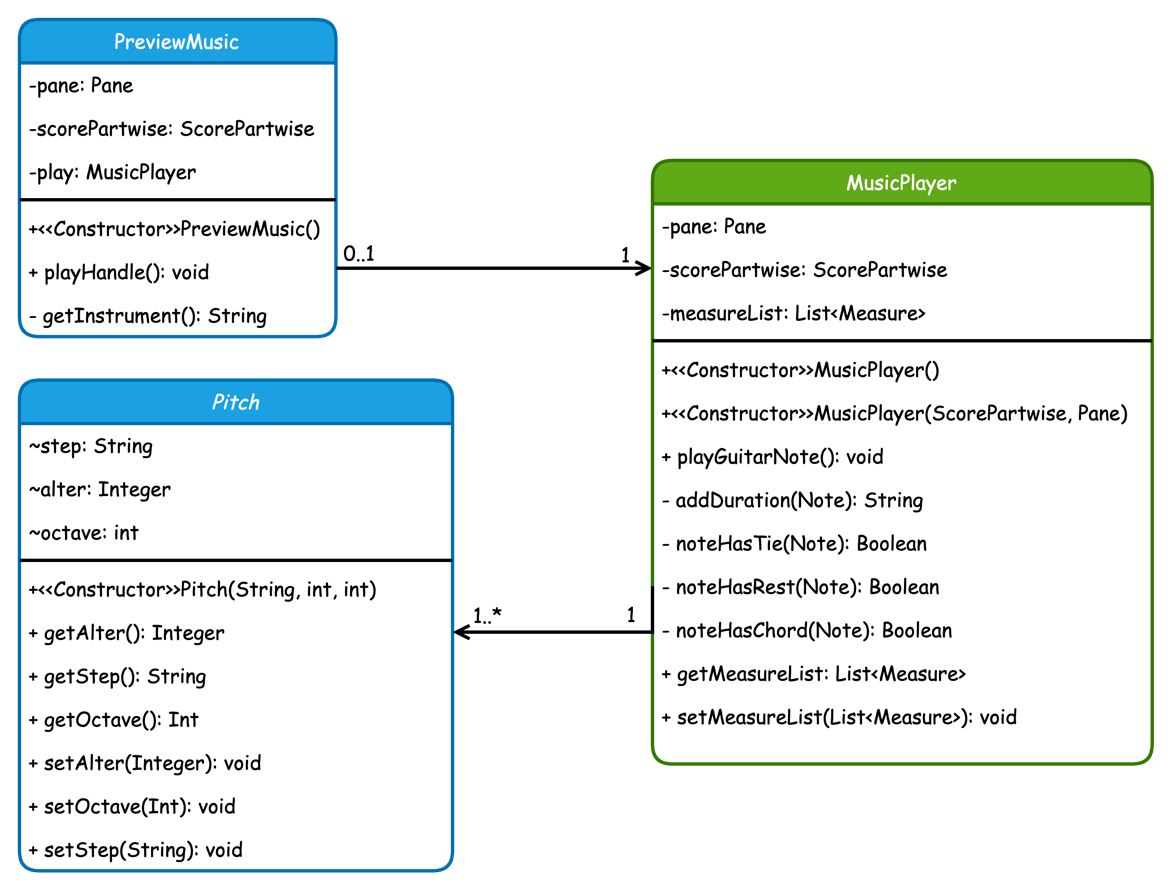
### **Activity Diagram**

1. A screenshot of a computer screen

   Description automatically generated with medium confidence

**Figure #**. Activity diagram of play music note function for guitar and drum

### **UML class diagram: *Guitar***

1. 

**Figure #**. The public, private, protected attributes and operations are denoted by “+”, “-”, and “~” respectively.

### **UML class diagram: *Drum***

1. A screenshot of a computer

   Description automatically generated with low confidence

**Figure #**. The public, private, protected attributes and operations are denoted by “+”, “-”, and “~” respectively.

# **Printing the Music sheet**

Some description…

### **Sequence Diagram**

A picture containing diagram

Description automatically generated

**Figure #**. description

### **Activity Diagram**

Diagram

Description automatically generated

**Figure #.** description

# **Go to measure**

### **[Name] Diagram**

**Group 6**

**Elmira Onagh**

**Irsa Nasir**

**Long Lin**

**Harjap Randhawa**

**Daniel Di Giovanni**