Design Document

EECS 2311 Group 3

Group 3:

Aman Patel

Phuong Tran

Mike Shen

Maksim Kolotev

Table Of Contents

Page No.

1. Introduction 2

    1.1 Purpose 2

    1.2 Scope 2

    1.3 Definition, Acronyms, Abbreviations 2

2. System Operations 3

3. Class Diagram 5

4. Maintenance Scenarios 10

INTRODUCTION

1.1 Purpose

MusicXML is a digital sheet music interchange and distribution format. The goal is to create a universal format for common Western music notation, similar to the role that the MP3 format serves for recorded music. The musical information is designed to be usable by notation programs, sequencers and other performance programs, music education programs, and music databases. This software converts musical tabs into a MusicXML file. So, users can create musical notations out of the MusicXML file.

1.2 Scope

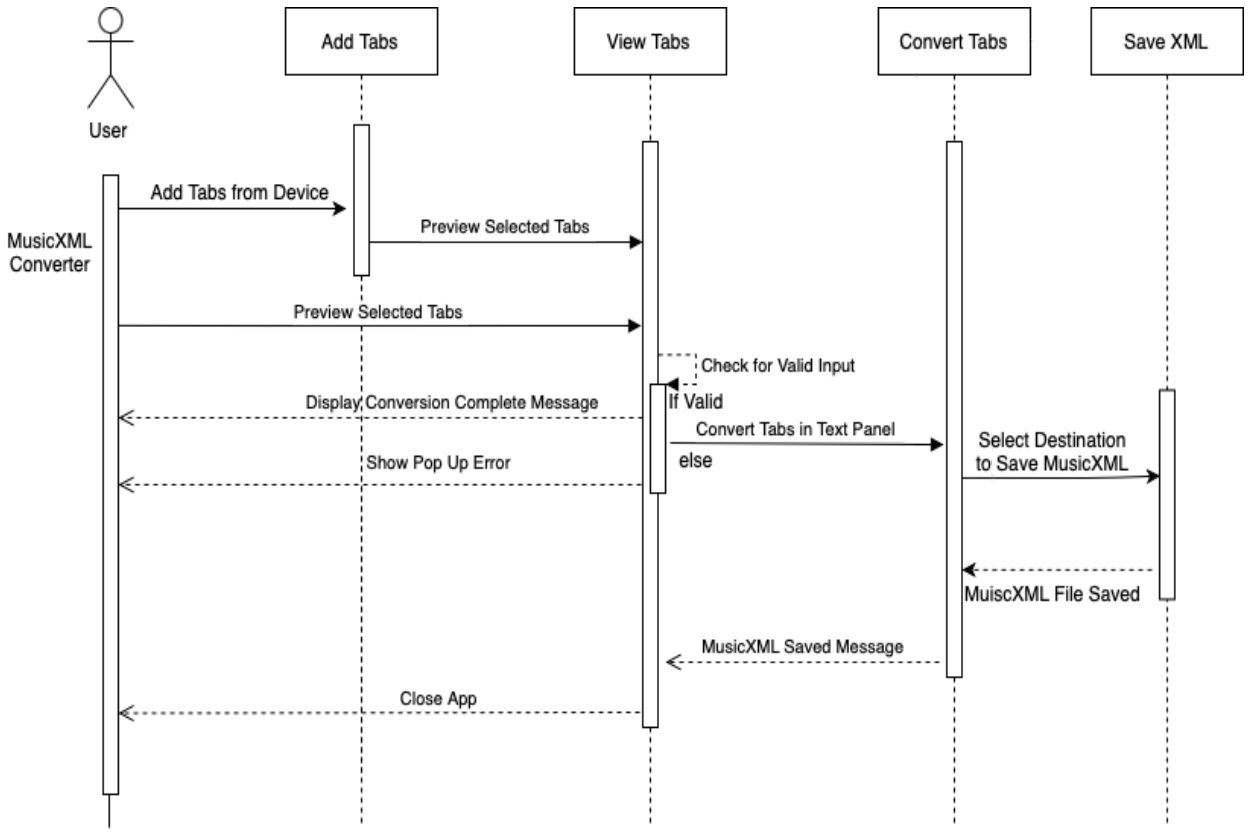
Musicians use sheet music to represent musical compositions, known as scores, in a textual form. Traditionally, the paper has been the medium for creating and sharing sheet music; however, this is changing. For over 10 years now developers have been working on and improving a product that stores sheet music digitally in plain text so that it can be easily shared and manipulated by musicians. This product is called MusicXML, it uses XML to store musical notation digitally, and it has become the standard open format for representing sheet music. The purpose of this project is to create a desktop application built in Java that can keep track of a library of MusicXML files in a database and parse these files on the fly. Covered in this report is the reasoning for creating the application, what tools were used, and how these tools were used to implement the project. As well, the design of the project, issues encountered when creating the application, and what was accomplished is detailed.

1.3 Definition, Acronyms, Abbreviations

1. MusicXML Convertor: Name
2. TextArea: Area where tabs are viewed or edited
3. XML Tree Area: XML Tree View

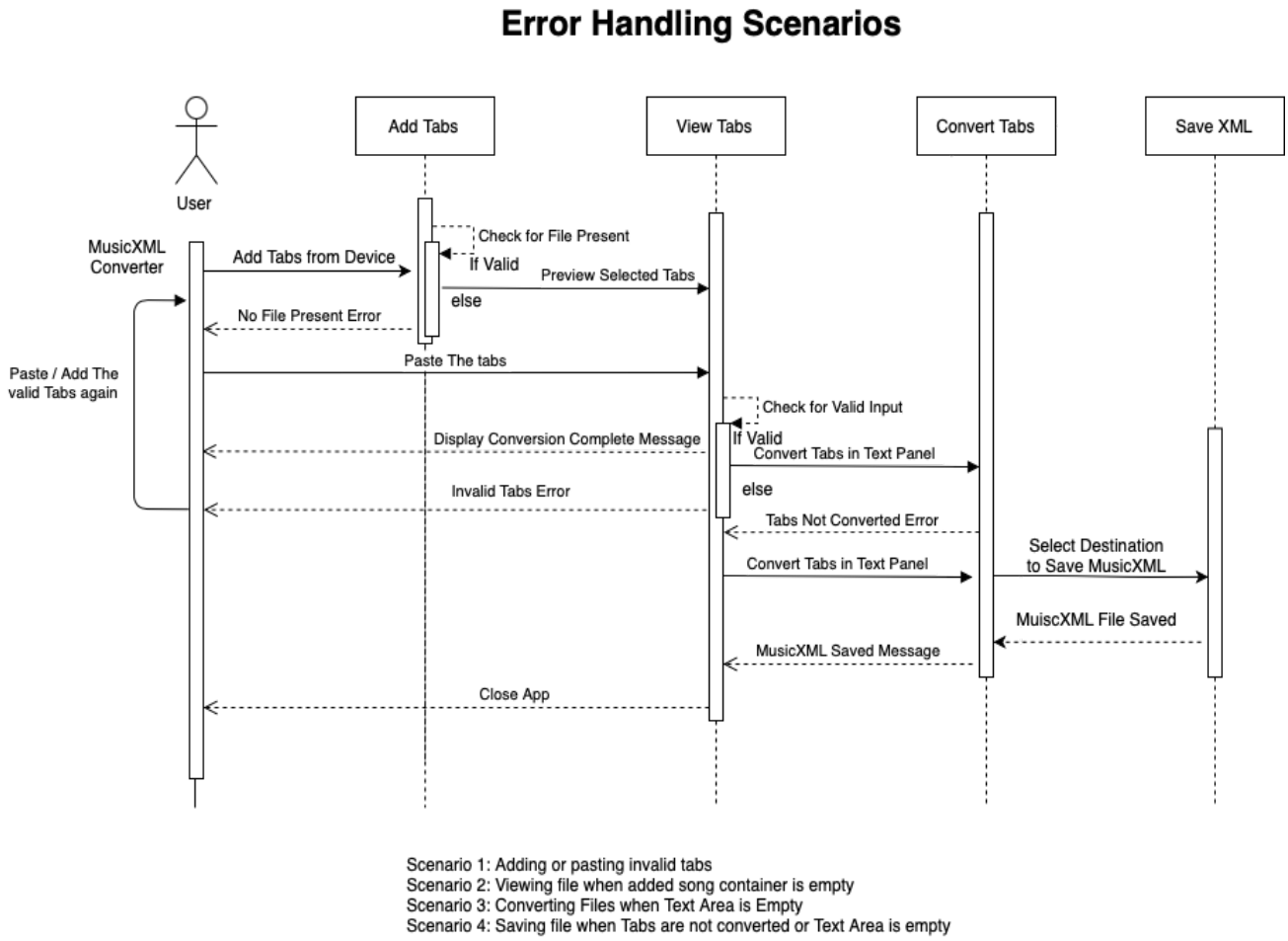
2. System Operations

The figure given below is the sequence diagrams for converting tabs into music XML format.



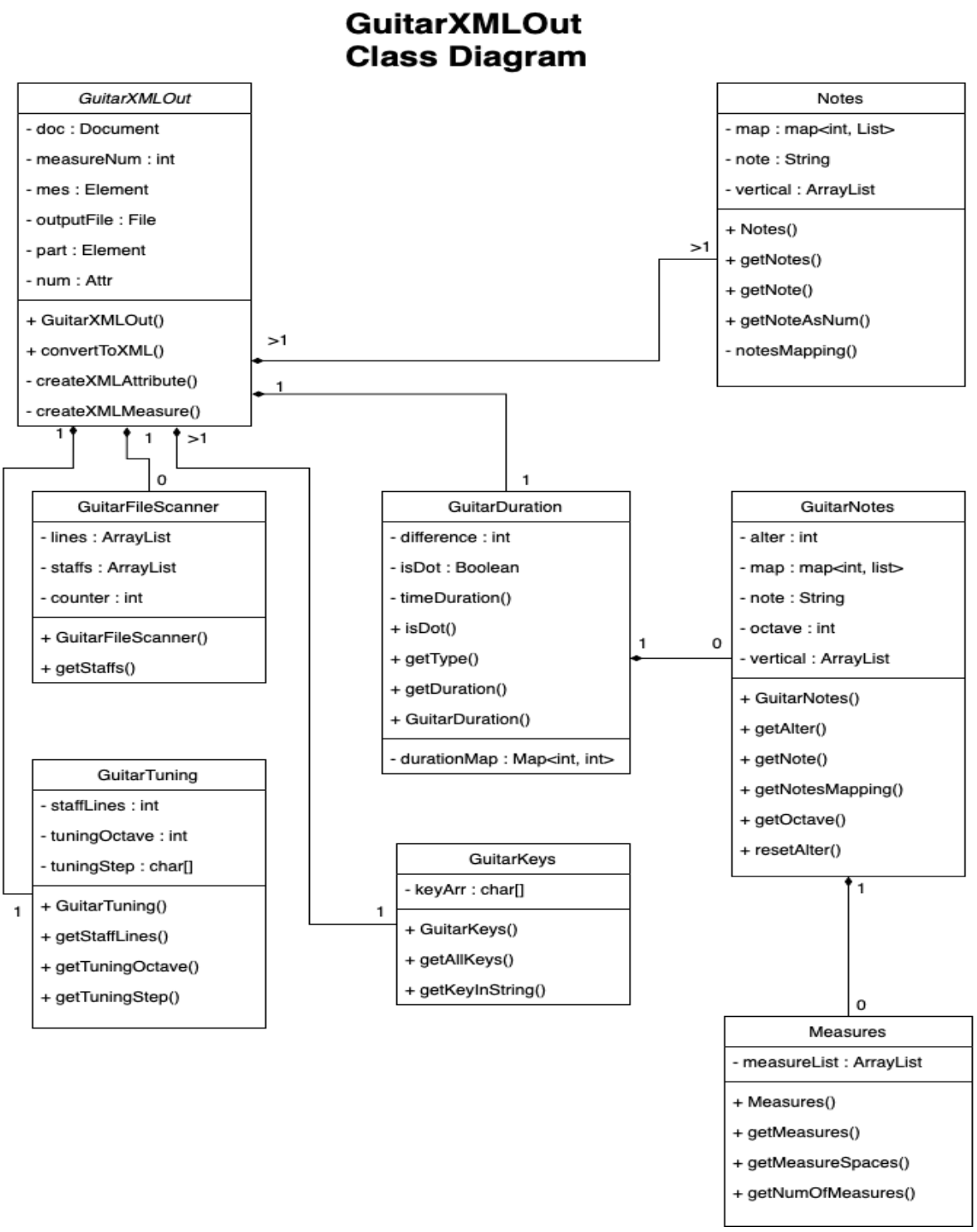
Given is a sequence diagram that depicts the steps in the way in which the user can convert the musical tabs into XML format.

Error Handling Scenarios Sequence Diagram:



This sequence diagram compresis of the error handling which users may encounter while working with System. All the specifications regarding how to deal with error can be found in the sequence diagram and users can troubleshoot the error. For detailed troubleshooting, Users can go through the USER MANUAL for the help.

3. Class Diagram



Provided UML Class diagram is for the Guitar Tabs. Class diagrams for the bass and Drums tabs are going to be similar to Guitar tabs. So, we are going to explain the design of the Guitar which will also describe the bass and drums respectively.

addSongAction(ActionEvent event)	
Input	.txt file or copy pasted tabs
Output	VOID
Description	This function is part of the MainController class. This function is specifically designed for the “ADD SONG” button on the GUI. Once you click the button you will be prompted to select a .txt file that can be imported into the GUI. Every other function in this class needs this to work correctly.

saveAction(ActionEvent event)	
Input	.txt file or copy pasted tabs
Output	VOID
Description	This function is part of the MainController class. This function is specifically designed for the “SAVE” button on the GUI. If the text area is not empty and is valid, it will read the contents of the XML text area and save it in a file which the user can save on their computer.

GuitarFileScanner(File file)	
Input	.txt file or copy pasted tabs from text area in GUI

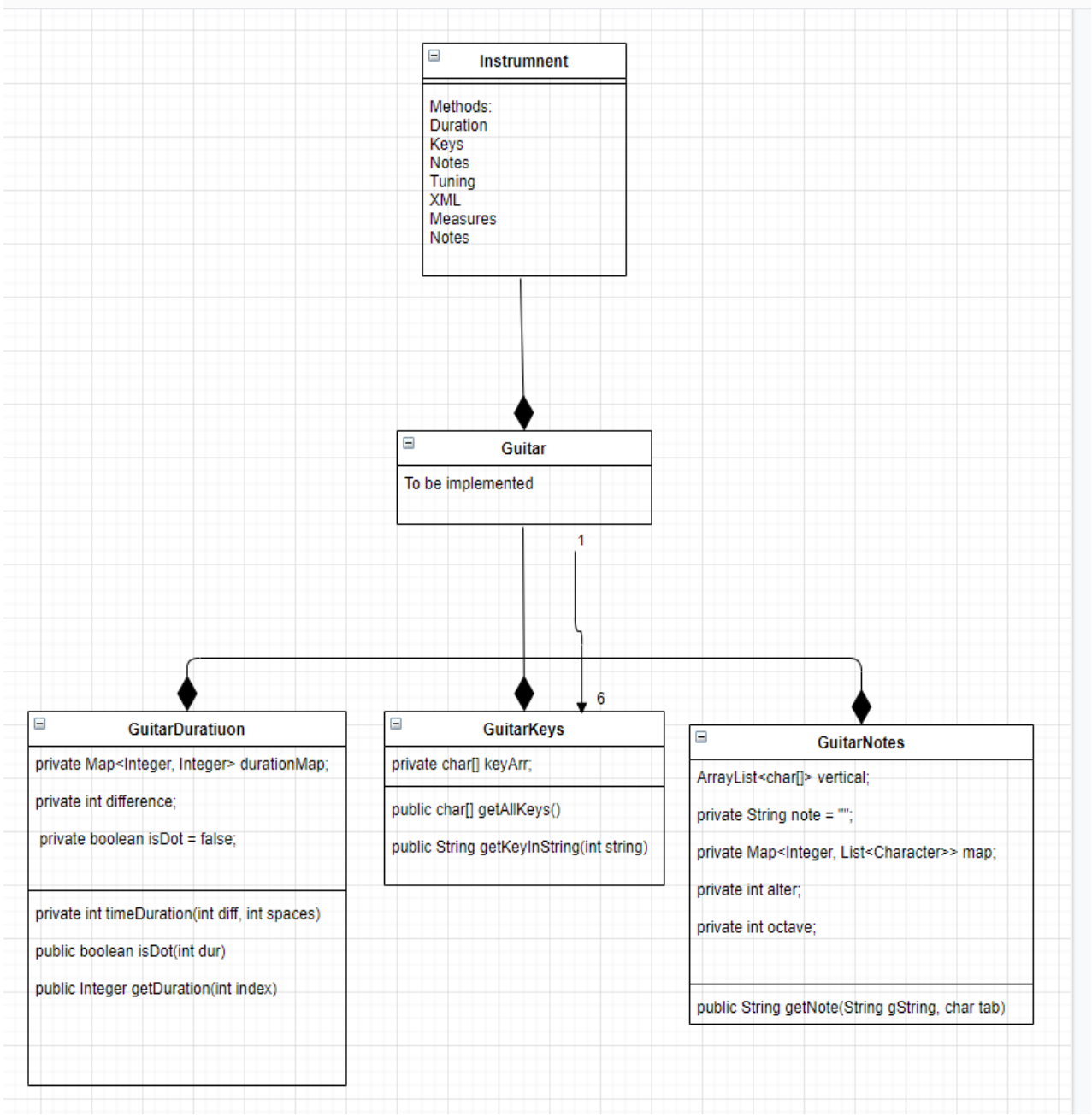
Output	VOID
Description	This function is part of the GuitarFileScanner class. This is the main function in parsing a specific tab and is used by most of the GUI functions. It parses through the input and organizes it accordingly. The new parsed file is then read by other functions such as getNotes or get Octaves.

convertToXML(FIle InputFile)	
Input	.txt file or copy pasted tabs
Output	musicXML file
Description	This function is part of the GuitarXMLOut class. This is called by the convertAction function from the main controller. This function builds the entire XML using other functions such as getNotes, getOctaves, timeDuration, etc.

getDetectedInstrument()	
Input	No Input required
Output	Name of the Instrument tabs detected (Drums, Guitar or Bass)
Description	This function is part of the InstrumentDetection class. The main use of this function is to iterate through specific tabs to determine if it is either a Guitar tab, Bass tab, Drum tab, or an improper tab.

convertAction(ActionEvent event)	
Input	.txt file or copy pasted tabs
Output	VOID (But in backend will convert the tabs into XML format for saving)

Description	This function is part of the MainController class. This function is specifically designed for the “CONVERT” button in the GUI. The main use of this function is to call several other methods such as detectInstrument, GuitarXMLOut, BassXMLOut, etc.



getNotes(String gString, int tab)	
Input	Single measure and note in terms of integer from parsed tab
Output	Specific note
Description	This function of the GuitarNotes class goes through each measure of a guitar tab and searches for integers that are defined as “tabs” each of these integers are also assigned to specific notes, which are then returned. Similar to this is the getAlter function, which searches the same thing but returns an alter, and the getOctave function which returns the octave for the specific case.

timeDuration(int diff, int spaces)	
Input	Amount of spaces from a Parsed tab
Output	Duration variable
Description	This function of the GuitarDuration class. This uses the amount of spaces in a tab to calculate the proper duration values, this is then used to determine how long a specific note will be played for using the getType function.

3. Maintenance Scenarios

The following is a perfective maintenance scenarios:

Supports Guitar, Bass and Drums in the MusicXML Convertor:

We have two fxml launchers:

Launcher.fxml: It gives the option to view the User Manual, Requirement Document, Testing Document, and last option to launch the application.

musicXML.fxml: It is a GUI for the application which allows users to carry out steps to convert musical tabs into an XML file.

3.1 Adding New Instruments:

Adding additional instruments to the software would be easy but time consuming. The user would have to be well versed with Java. In group 3's MusicXML converter, each instrument has its own individual classes and they do not share any. Therefore for someone who would want to add extra instruments would have to create whole new classes that are similar to the classes that are already implemented. Once that is done the next step would be to create a new XMLConverter that calls upon these newly made classes to properly build the XML. Finally changes would need to be made in the mainController class, so that once the program can recognize the tab and relates it to the proper instrument, the program will then call the correct converter.

3.2 Addition of New Special Notes:

Currently, special note types such as ghost notes or hammer-ons are processed inside the notes class of the specific instrument (Example: hammer-on are detected in GuitarNotes.java class) . More specifically inside this class, the method known as getNotesMapping is what detects these special symbols such as “h” or “p”... etc. If a user would want to implement any other special symbols or custom symbols (let's say a random character ‘/’, that the user will use as a slide) inside this method. If the user would want to add anything else such as “vibrato” this would also need to be added to the GuitarNotes.java class (or any other related class). Preferably the user would create a new method inside this class that will detect the desired special element.

3.3 Supporting Drag and Drop:

File and Classes which needed would be MainController Class and musicXML.fxml file. So, In MainController Class we would need to add new events to accommodate drag and drop functionality and Some more error handling methods are needed to ensure the input is processed correctly and in the musicXML.fxml file, new events needed to accommodate drag and drop functionality.

3.4 Changing the JDK of the project:

The application's JDK i.e. Java SE-15 may not have been downloaded by everyone causing an error when launching the app. To fix this error, one must be able to change the JDK on the project.

1. Right-click on the project and hover over "Build Path", then click "Configure Build Path".
2. Click on the JRE System Library then click "Edit" on the right of the window.
3. Select "Alternate JRE" and then click "Installed JREs…" in its right.
4. Choose the JRE you have installed and click "Apply and Close" and choose "Apply and Close" until you return back to the first window, then click "Finish".

These changes will require changes to be made to the build.gradle file.

5. Now open the build.gradle file from the Package Explorer.
6. Navigate to the java tag on the build.gradle file and set the sourceCompatibility and targetCompatibility accordingly to the version you wish to upgrade to.