Faculty of Electrical Engineering
and Computer Science

Testing Document
Cover Page

Winter of 2022
EECS 2311 – Section Z, Lab 01
Software Development Project

Testing Document
Due Date: March 6, 2022,

Group Members (Name and Student ID)	
Aleksander Weinberger	216627994
Harsimran Saini	215604960
Chirag Sardana	215225642
Shaharyar Choudhry	218027326
Hoshner Tavadia	217828567

"The document describes the test cases run, the way they were derived, and the way they were implemented. Discussion on why this should be sufficient testing is included. Professional presentation. The important components that need to be tested are the GUI, Audio(play music capabilities). This will be done using JUnit 5 and run via the Gradle task".

Testing commands

GUI testing

- Test → buildPane()
  - Checking if the staff is appended to the scroll pane
- Test → ConstructBarLine()
  - Checking if the bar-line is constructed right after a measure
- Test → constructClef()
- Test → constructTimesig()
  - Checking the time sig
- Test → constructMeasureGrid()
  - Check to see if notes are placed on the right strings of the staff
- Test → staffHasSpace()
  - These were checks to make sure that measure can still fit on the staff

Play music Testing

- Test → playMusicSheetButtonHandle()
  - Checking to see if the music is played
- Test → pauseMusicSheetButtonHandle()
  - Checking to see if the music is paused

Saving Music sheet

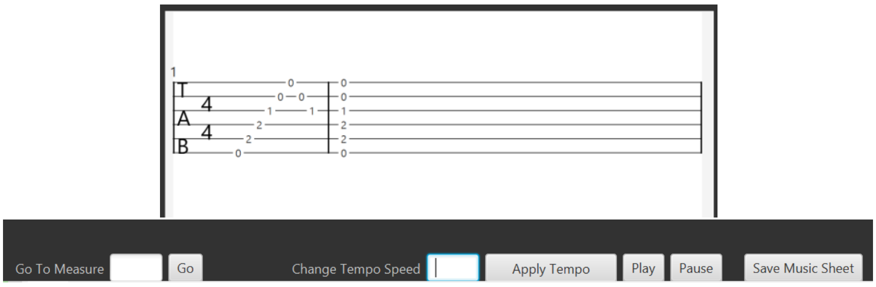
- Test → saveMusicSheetButtonHandle()

Music Player Test Cases

1) Test Case 1: Test to see if the "Play Music" button works when a valid input in entered

How was this derived: It was a function that needed to be implemented as stated by the "customer" and the way we implemented it was to use a play button in which, when pressed, would play music audio. Initially, the "play music" button was able to be pressed even if the user entered no input. Therefore, clicking the play music button would create a null pointer exception. We fixed this problem by only allowing the user to play music once the user has entered a reasonable input

How this was tested: We ran the code with a valid input and clicked the "Play Music" button. This worked because now expected the button to be pressable and not grayed out. We then compared the outputs with what we were looking for.

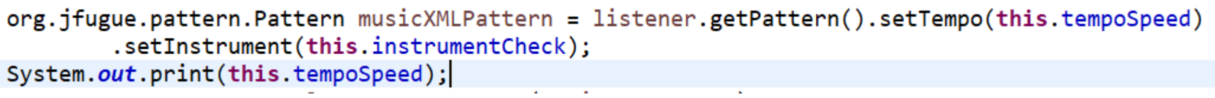


2) <u>Test Case 2</u><u>: Test to see if the new tempo is applied to the music player</u>

How was this derived: It was a function that needed to be implemented as stated by the “customer” and the way we implemented it was to use a textfeild in which, the user enters a tempo speed integer and clicks “Apply Temp” to update it. Upon pressing play, the program should play the music again with the newly applied tempo. It is important to test this as the customer will most likely want to change the tempo of the music

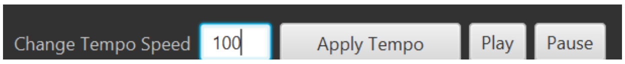
How this was tested: We ran the code with a valid input, set a new tempo of 100, clicked the “Apply Tempo” button and clicked play. Now the program should print the inputed tempo speed in the console. This is to check if the private varibale “this.tempoSpeed” has been updated or not. If the number outputted to the console was not 100 then we would know that the user inputted tempo has not been applying. But if the console does have “100” ouputed and the music has slowed down, then we know for sure that this test has passed. Below is the code for setting and printing the tempoSpeed.

Code Used For Test:



```
org.jfugue.pattern.Pattern musicXMLPattern = listener.getPattern().setTempo(this.tempoSpeed)
        .setInstrument(this.instrumentCheck);
System.out.print(this.tempoSpeed);
```

Input:



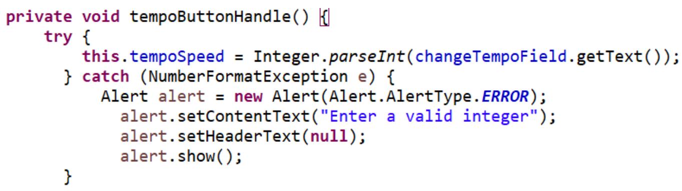
Change Tempo Speed 100

3) <u>Test Case 3:</u> <u>Test to see if program handles a incompatible input for temp speed</u>

How was this derived: Since the button for apply tempo is always clickable the user can enter a wrong input and run the program. If the user enters a letter or decimal this will result in an error. Therefor it was important for us to handle this potential issue.

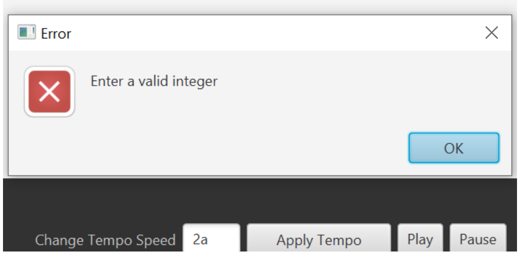
How this was tested: We ran the code with a valid input, set a new tempo of 2a, and then clicked the “Apply Tempo.” We then waited for a message to pop up saying “Please enter a valid integer”. This should notify the user to change their input. The code below handles the error and was created for us to visualize our tests.

Code Used For Test:



```
private void tempoButtonHandle() {
    try {
        this.tempoSpeed = Integer.parseInt(changeTempoField.getText());
      } catch (NumberFormatException e) {
          Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setContentText("Enter a valid integer");
            alert.setHeaderText(null);
            alert.show();
      }
```

Input + Output:



Error

Enter a valid integer

Change Tempo Speed 2a

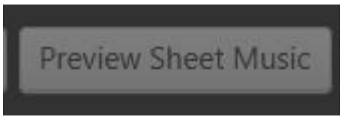
Sheet Music Creator Test Cases

1) **Test Case 1**: Test to see if the “Preview Sheet Music” button becomes enabled when a non-empty output in entered

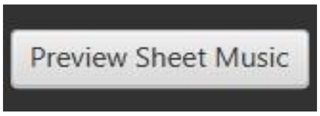
How was this derived: Initially, the “Preview Sheet Music” button was disabled if the user entered input. When the user enters a ASCII music tab. It enables the “Preview Sheet Music” button along with other buttons.

How this was tested: We ran the code with an empty input and found the button to be disabled. When a correct tablature was inserted from the test cases, the button became enabled, and we were able to click it.

Disabled Button:



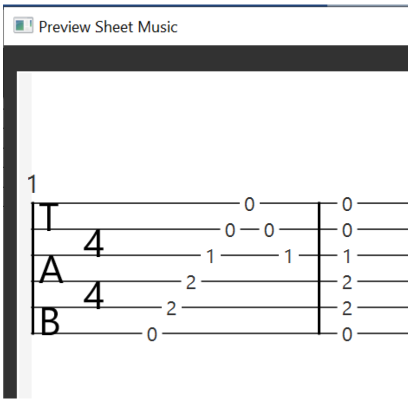
Enabled Button:



2) **Test Case 2:** Test to see if the “Preview Sheet Music” button, on the main screen, works when an input of Guitar text-tablature is used.

How was this derived: It was a function that needed to be implemented as stated by the “customer” and the way we implemented it was when it was clicked another window would open and display the music tablature.

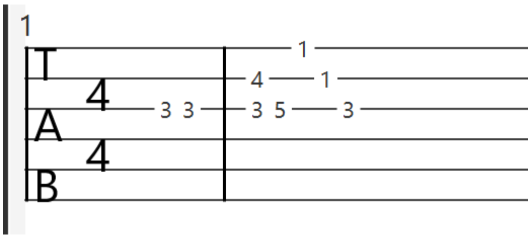
Output:



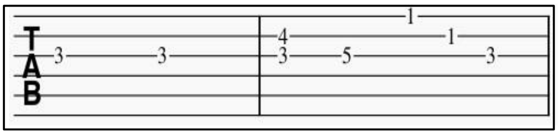
3) **Test Case 3:** Test to see if the Music Tablature Produced by the software is accurate.

How was this derived: We compared the music tablature created by our software to a music tablature created by MuseScore3.0 and SoundSlice web application. Our results were 90% similar to these applications.

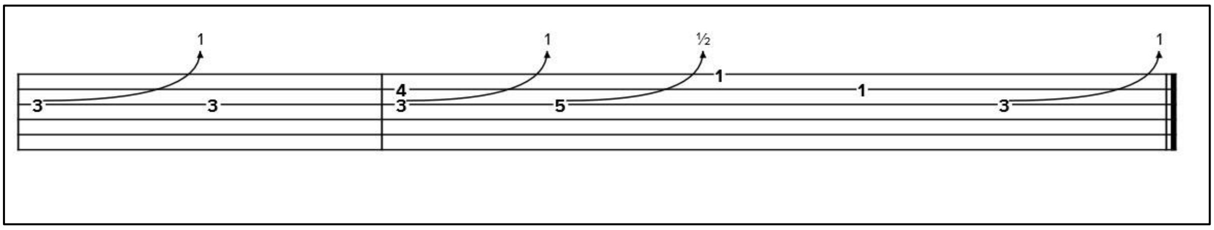
Our Result:



MuseScore3 Results:



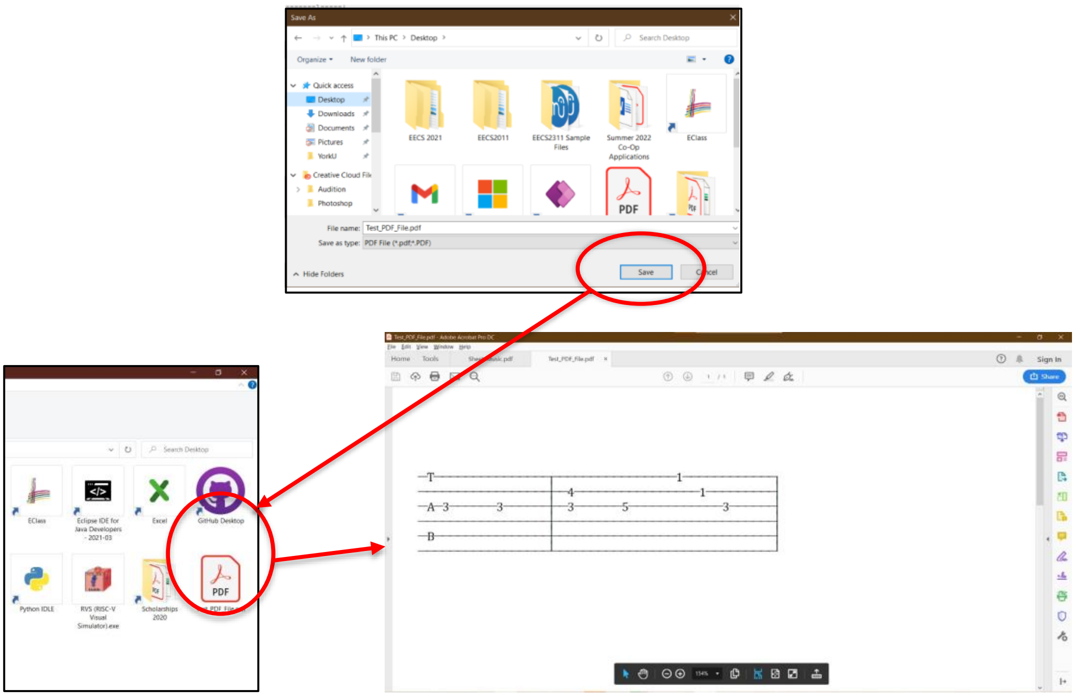
SoundSlice Results:



4) **Test Case 4:** Test to see if our program can export the file as a PDF document.

How was this derived: We tested the feature in our software to export the music tablature as a PDF.

The software was able to open a window to enter the file name and then would open up a file directory to allow the user to select a destination and save the file. Output:

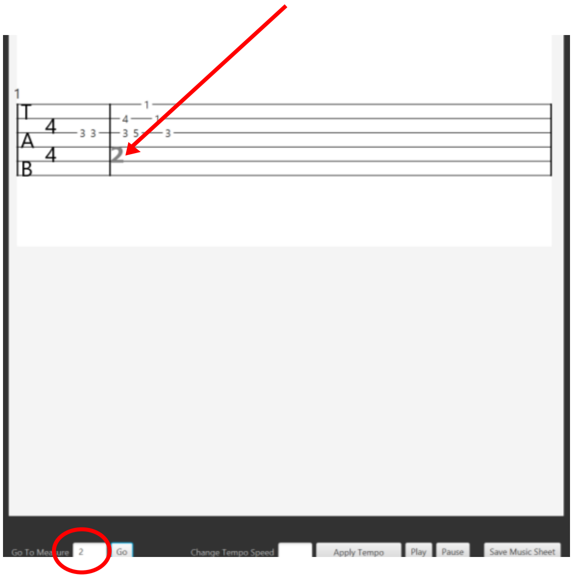


5) **Test Case 5:** Test to see if a visual output appears when "go to measure" is used

How was this derived: It was expected that the previewer should allow the user to see what measure they are looking at, as well as support a go to measure function as in the text input. This is a required featured by the client so it was crucial for us to test it thoroughly.

How We Tested: After entering a valid input such as bendtest.txt. We then clicked the preview button to view the sheet music. From there we entered different inputs and numbers to see if the correctly measure is focused. We confirmed the correctness by counting the measures ourselves to see if they matched with the output.

Input = 2, Output:


1
T
A
B
4
4
2
Go To Measure
2
Go
Change Tempo Speed
Apply Tempo
Play
Pause
Save Music Sheet