Final Report

Java 2, PROG24178

Summer 2022

ID 991681013

Zheng Yi Wang

**Overview:**

**This project’s overview is essentially the same as the project proposal. This app was made to keep track of songs and store different aspects of songs as an integer array. The array would describe how the song feels and the rest of the fields would be information of the song. The current app stores songs with a song name, artist name, genre and a mood array.**

**File IO:**

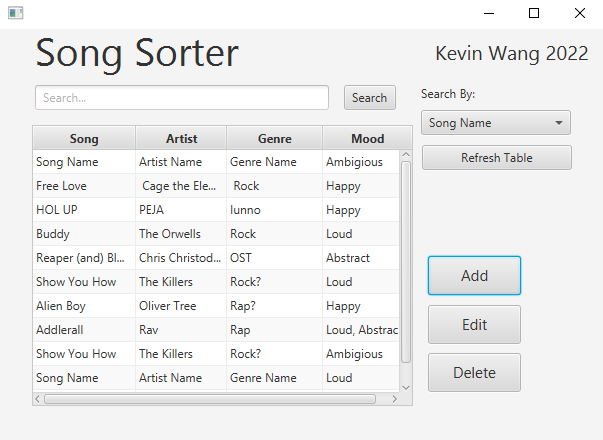
**The file IO is built into the SongList class, since it would take the text file and convert it into a static arraylist. The static arraylist would then be referenced by everything as the database, and it’s only when deleting or adding would the SongList class write to the file, then reloading the arraylist. The arraylist is important because it acts as a safety barrier by acting like a temporary database that gets initiated every time the application starts.**

**SongList and Song class:**

**The arraylist named songList is made up of Song objects which come from the Song class. The Song class is relatively straightforward, it stores all the data a song could have but I did add one more thing in there called the moodString. Humans can’t look at an int array and get the mood of a song, so the moodString takes the most applicable moods for a song and conveys it; making it more user friendly. This moodString can be seen in the screenshots of the GUI of the main page.**

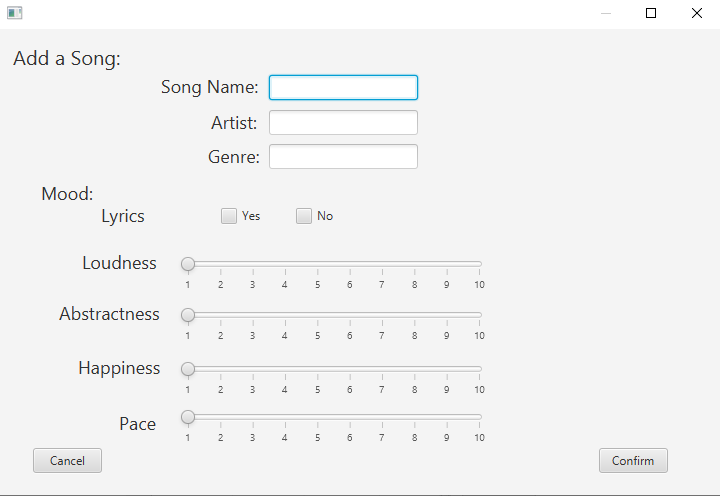
**Main Window:**

**The app itself has a main page which has a Tableview that shows the list of the songs. This table view was tricky to configure since it required making another observable array list so the GUI components can read it. This led me to refactor a lot of methods that used Song because it had to be a “SimpleStringProperty” instead of a regular string. In the end I chose tableview instead of listview is because you can sort the list easily and it looks *much* more organized. Another thing that was different from the project proposal was the search function. I used choicebox instead of combobox since I didn’t need a large list to search through. I also added a delete button for a delete function (something I overlooked) and a refresh table function to show everything again in case the search bar gets whacky.**

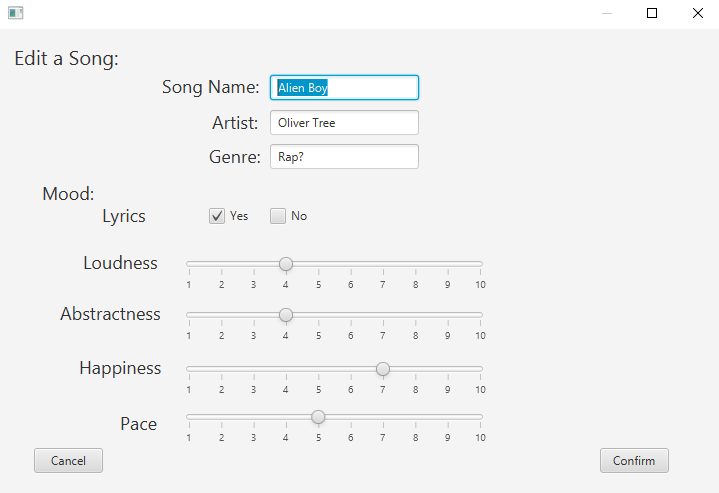
****

**Figure 1. Screenshot of the main page of Song Sorter.**

**Add and Edit windows:**

****

**Figure 2. Screenshot of the add song page**

****

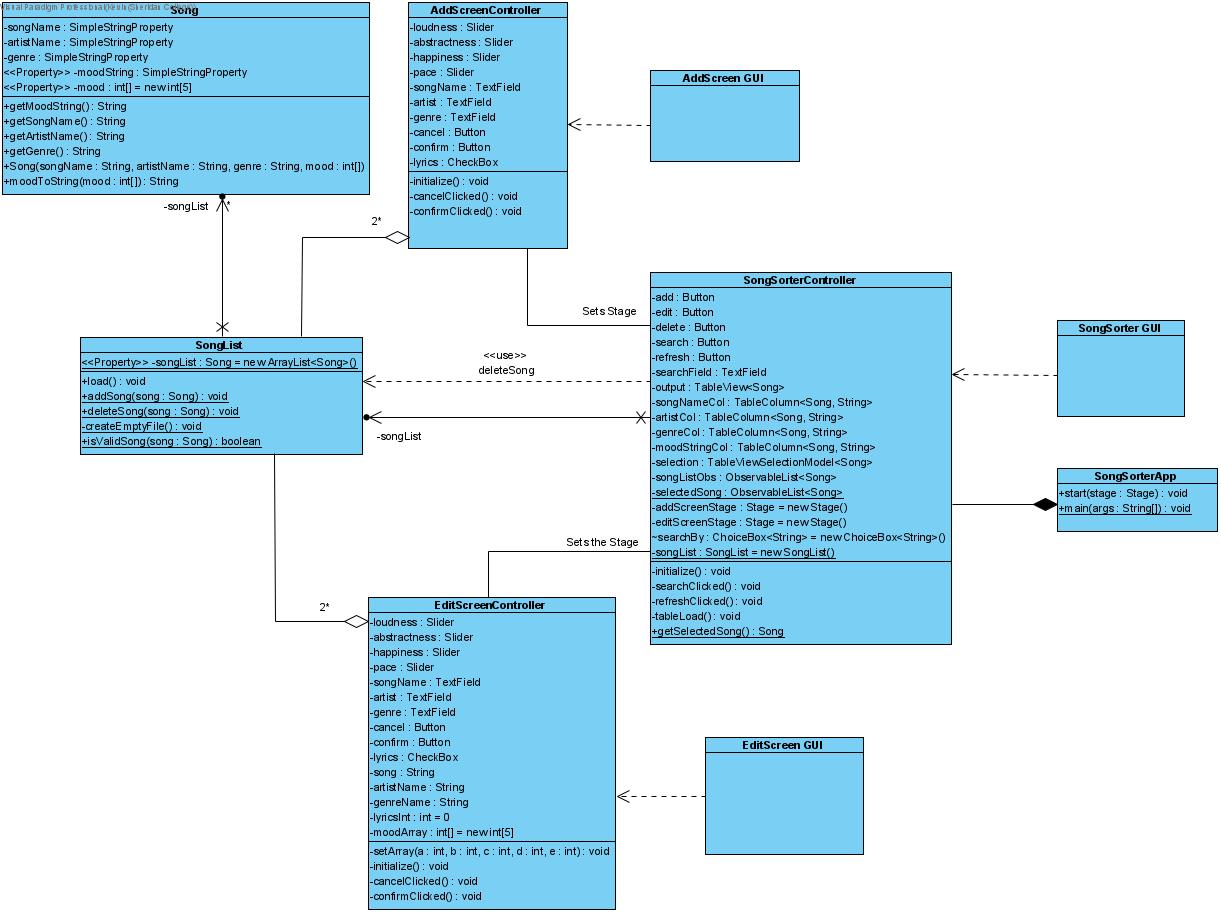
**Figure 3. Screenshot of the edit window having a previous record already selected**

**The add and edit windows are very similar in design, but their functionalities differ slightly. The only parts where they’re different from the project proposal is the project proposal used radio buttons for lyrics while I just used checkboxes since checkboxes returned Boolean results. I could then just use that Boolean and convert it to either a 1 or 0, making it easier for me to insert it into the mood array. Also, the “No” checkmark box for lyrics is only there for aesthetic reasons, it doesn’t do anything.**

**Now for the functionality of the add window. When the user hits confirm it will first check whether or not the fields are legitimate by using the method isValidSong from the SongList class. It checks the fields for any commas (since that would mess up the regex) and then if there aren’t any commas, it would add the song to the song list; overwrite the text file; then close the window with cancelClicked method.**

**With the cancelClicked function it fires off a window event which tells the main window to reload the songlist again. This part was actually sort of a pain, because it required me to implement another event handler so I can handle the window closing, and it used methods that seemed more like javascript than java.**

**The edit window is extremely similar to the add window, but it uses a select model to pass in information about what’s going to be edited. The select model is from tableview, and it’s set so you can only select one line at a time. The selected song’s information, when passed into the edit window, is pre-set in the textfield and sliders upon initialization. And when the user presses confirm, the editcontroller class calls on the songlist class to delete the old song which was selected and add the new song to the database, giving the illusion that you’re editing the current song.**

****

**Figure 4. Class diagram of my final project. The fxml files are not filled in.**

**Conclusion:**

**In conclusion, I think I learned a lot of stuff that’s outside of the curriculum such as how to use tableview, how to pass off selected values and how to integrate GUI better. I didn’t subclass or override any classes in my final program since I felt that the classes were cohesive enough as they were. The final project also showed me a lot about regex and how to use regex to my advantage. There were numerous difficulties during the project such as setting the sliders, converting between simple string property and understanding how an observable list is different from our standard array list, but in the end, I feel that through the power of my brain and stackoverflow I made it work. There are still some little changes that I could make such as making the lyrics field do something, or making the sliders go up to 10 instead of starting at 0 and going to 9; but those changes require a lot more work than meets the eye.**

**One feature that I do think would be useful would be to have a save backup button that lets you write a backup, and load the backup. That would be extremely useful in case the text file goes missing or gets corrupted. That would, of course, be above and beyond, and considering I got this much done as a single person I’m more than happy to call it here.**

**One thing I didn’t note in this document are the alerts that show up on screen every time the user needs to be notified of something. These are added into try-catch blocks so the user knows why something didn’t work.**