Assignment 1

Song Library GUI Design & Implementation

Posted Tue Feb 2. Due Fri Feb 12 by 11 PM.

Worth 50 points (5% of course grade.)

You will work **in pairs** on this assignment. Read the DCS Academic Integrity Policy for Programmming Assignments - you are responsible for this. In particular, note that **"All Violations of the Academic Integrity Policy will be reported by the instructor to the appropriate Dean".**

Requirements

Design and implement an application with a graphical user interface to manage a library of songs. **A song is uniquely identified by name and artist (case insensitive).** Your application should have a SINGLE WINDOW with three functions:

1. **Song list display**, with the ability to select ONE song from the list. The list will display ONLY the names of the songs, in **alphabetical order of names**. Unless the list is empty, one song is always pre-selected, and its details shown - see the following item.
2.
3. **Song detail**, with name, artist, album, and year, of the song that is selected in the song list interface
4.
5. **Song Add/Delete/Edit**, for adding a new song, deleting a selected song, and editing a selected song information:
    1. Add: When a new song is added, the song name and artist should be entered at the very least. Year and album are optional. If the name and artist are the same as an existing song, the add should not be allowed - a message should be shown in a pop-up dialog.

The newly added song should automatically be placed in the correct position in the alphabetical order in the list. Also, it should be automatically selected, replacing the previously selected song, and its details should be shown.

2.
3.
4. **Edit**: ANY of the fields can be changed. Again, if name and artist conflict with those of an existing song, the edit should NOT be allowed. alog.
5.
6. **Delete**: Only the selected song in the list can be deleted. After deletion, the next song in the list should be selected, and the details displayed. If there is no next song, the previous song should be selected, and if there is no previous song either, then the list is empty and the detail info is all blanks.

For all of these actions, the user should be able to cancel on the action midstream if they change their mind.

Note: If you use the song detail display area for add/delete/edit as well (instead of two separate areas), then make sure the interface is not confusing to the user, particularly if the edit is disallowed or aborted by the user.

When your program is started, it should show the current list of songs in the library, in the song list display, with the first song selected by default. (The first time the program is run, there should be nothing in the display, since there won't be any songs in the library.)

The data should be persistent across different sessions of your program. You can save the song list in a file in whatever format you like, and then read it into your program when it starts up.

Your application is NOT required to play a song when it is selected. In other words, there is no requirement to have audio playing functionality, and there will not be any extra credit if you choose to implement this functionality.

The application window need NOT be resizable. When the window is closed, the application should be terminated.

Most of the aspects of design (layouts and widgets) and implementation (event handling) you need to do this assignment have been covered in class. If there is any aspect you want to add that has not been covered in class, you are expected to discover these for yourself.

You MUST use Java FX 8 for all UI aspects. (No Swing stuff.) Also, all UI layout should be done in fxml.

Grading

GUI Component	Points
Song list+selection	10
Details display	8
Add/Delete/Edit	25
Persistence	7

For the first three items, the grade will depend on correctness and completeness of functionality, as well as effective/appropriate use of JavaFX widgets. For the last part (persistence), grading is on correctness, and on ensuring this is transparent to the user, i.e. the user should not have to be concerned with where and how the song list is stored between sessions.

Effective/appropriate means the UI should be clear, the user should know exactly what to expect when events are triggered, the user should know exactly what to do to achieve their objective. If you need to explain to someone how to use the interface, then it's not a good enough interface.

IMPORTANT: Your UI should not use widgets that are an overkill for the task at hand because if a complex widget is used for a simple task, it will confuse the user. (So, for instance, do NOT use a table view to show song details because the requirements do not ask you to dynamically rearrange the display on user request.)

Finally, your UI is not required to be aesthetically pleasing (no points for this because it's highly subjective), but you are most welcome to dazzle us with your creativity (just for the heck of it!) as long as it does not get in the way of clean functionality.

You may use any of the classes in the Java 8 SDK, but **no external third-party libraries.**

Point deductions

Be aware that implementing design and functionality as specified, and submitting ("releasing") your implementation as required, are both important aspects of software

development in the real world. The point deductions listed here are intended to alert you to this.

- 5 points, if the application window does not show up at all, but we can fix the code to make it show. So make sure it actually shows when you run the application.
- 5 points, if the application window cannot be closed, or it closes but does not terminate the application
- ~~10 points, if any window other than the application frame pops up at any time (this includes dialogs)~~
- Up to 5 points, if you do not write your name at the top of all the files you submit.
- Points will be deducted for each test case that cannot be tried because either your data file or your UI does not allow for testing it. The exact number of points will depend the point value of the test case(s).
- 5 points, if you do not implement your main method in a class named SongLib (see Submission Instructions)
- 5 points, if you have any code in the default package (directly under src) (see Submission Instructions)
- 10 points, if you do not submit your implementation in a project zip file (see Submission Instructions) that can imported into Eclipse as a complete project
- **No credit**, if you used any external libraries--you are only allowed to use the standard Java SDK.

Submission Instructions

You may implement this program in as many files as you want, but you MUST call your main class SongLib (this is the class with the main method.) If you don't do this we will have to first find which of your classes has the main method so we can run your program - 5 points will be deducted!

Also, you must organize your classes into packages. There is no requirement for how many packages you need to use, as long as there is at least one. In other words, none of your classes must be directly under src (which basically amounts to a default package) - this is bad design because it severely limits the way in which you can control access to members of a class.

Make sure to write your names at the top of each of the Java source files in your project.

Export your project to a zip file called songlib.zip (see how to do this in the Eclipse page, under "Zipping up a Project"). Make sure you have done this correctly by importing it back as a project into Eclipse (see "Importing a Zipped Project into Eclipse" in the Eclipse

page). Since you are working pairs, a good test would be for one of you to zip the project, and for the other to import the zipped project into their Eclipse installation.

10 points will be deducted if we get a zip file and/or contents that are not legit, as described above

Submit your songlib.zip file to Sakai. Just ONE submission per team, please!