1. **General Program Design**

To create my music player application, I began by creating a class called MusicTrack. I assigned each MusicTrack object the attributes laid out in the requirements, including the item code, song title, artist name, album name, and price. I designed my file reader so that the user can import a pre-existing library as a semi-colon delimited text file, or create a new database if the file path he/she originally specifies does not exist. If a song is added, deleted, or edited to/from this database, these changes are saved, and the modified file is saved when the user exits the application.

If a file is imported as a run-time parameter, I read specific portions of each line into an array list (for song titles), a tree map (to map song titles to item codes), and another tree map, to map item codes to MusicTrack objects. I use the array list to create an observable list of song titles, which I in turn feed into a combo box, so that the user may select a track by title. If a user chooses to create new database, I initialize this database with one MusicTrack object where all fields are set to equal “Null.” I then force the user to Add a new song before any other button option is enabled.

Each button is tied to an ActionEvent handler, which queries the MusicTrack TreeMap as warranted (depending upon the operation), and refreshes the contents of the attribute text fields. To validate user input, I also created error message Text Fields for each attribute, and I do not allow the user to “accept” changes after adding a song or editing a song until each attribute field is validated as correct.

1. **Alternative Approaches Considered**

One major challenge I encountered in designing my approach was that I had originally used song titles as keys in the TreeMap I made to hold my MusicTrack objects. As I was designing the “Edit” button functions, I realized that since the user could edit the description of a song, and I had soft-coded each title using the description, the song’s original key would not necessarily match up to the edited song title. Thus, while I considered using a single TreeMap, I realized that I needed a way to match the combo box selected value to the Item Code, which would not change even in the event that a song’s description was edited, and I decided to create an additional TreeMap that used each song’s title as the key and the ItemCode as the value.

With respect to the GUI, I originally considered using a BorderPane layout, but realized that it would be more intuitive to have the labels, text fields, and error message text fields aligned horizontally for each MusicTrack attribute. For this reason, I switched to using a GridPane, which allowed me to align each set of objects more precisely.

1. **Lessons Learned**

This project helped me to understand how soliciting and processing user input through a GUI is different than soliciting and validating user input from the console. I had a particularly difficult time figuring out how to allow the user to correct input mistakes (i.e. supply content if a field was blank) without processing the rest of the supplied content, because I originally only had event handlers attached to the buttons, but needed a way to check input before the database querying and text setting associated with each button event was carried out. I learned that one way to validate user input is to set up a series of Boolean operators, and use an if/else if loop to force all of these operators to evaluate as true before the combo box option is allowed to trigger an action event.

If I were to do this project again, one thing I would do differently would be to streamline redundancies that exist in the code by creating methods to iterate/automate the text field “setting” that is currently done one text field at a time, even though the syntax of each operation is the same. I would also consider using colors and/or text prompts in the TextField boxes to help the user understand if the input he/she had entered was correct in a more intuitive and visually appealing way. I would also try to make my code more scannable by finding a way to break some of the functionality up into methods that I could then call within EventHandler code blocks.