**Nojus Sinkevicius**

**x22535216**

**Data Structures & Algorithms**

**CA1 Report**

Design Summary:

For this music manager application, I adopted an object-oriented approach and tried to keep the design of the application somewhat appealing. I kept applications such as Spotify and Soundcloud in the back of my mind to guide me throughout the process. .The design consists of several key components: abstract data types (ADTs), interfaces, and classes, as requested in the brief.

Abstract Data Types (ADTs):

I used ADTs to encapsulate the data structures used for storing the song information. I used Arraylists of String arrays (ArrayList<String[]>) to represent the songs in each playlist. These Arraylists provide flexibility in managing song data, allowing for dynamic addition, removal, and manipulation of songs. I used the songs that were stored in Arraylists to update the jTable. I created an Arraylist for each jTable, one for the liked playlist, party playlist and the sad playlist.

Interfaces:

The SongManager interface defines some operations for managing the songs like, adding, removing, moving up & down, and retrieving songs. By defining these operations in an interface, I made sure that multiple classes (such as MainApp) could use the same functionality, promoting code reuse and flexibility.

Classes:

MainApp: This is my main class and serves as the core of the application, responsible for the logic. This class manages songs across different playlists (liked, party, sad). It implements the SongManager interface and provides methods for adding, removing, and manipulating songs within each playlist. Additionally, it includes methods for searching songs and updating tables and labels.

Menu: The Menu class represents the graphical user interface (GUI) of the application. It interacts with the user and provides actions to the MainApp class for song management. The class includes event handlers for different user actions, such as adding/removing songs, moving songs within playlists, and searching for songs. It also updates the GUI components based on changes in the tables.

Data Management:

To manage song data, I used ArrayLists to store songs in each playlist. This approach offers flexibility and efficiency in handling dynamic lists of songs. By incorporating the song data within ADTs and providing methods for manipulation, the application maintains a clean separation between data and functionality, adhering to the principles of object-oriented design.

Retrospect:

Looking back, I would incorporate more swing ui elements and try to keep design as consistent as possible. I had lots of issues with this project, even small ones. I had wrote half the application and realised I have not used interfaces so I decided to scrap what I had done and restarted. The 2 major issues I faced was implementing a function that updated the Jtable from an arraylist instead of just directly updating from user input like joptionpane. This definitely took the most time out of everything during the development of this project. The second issue I faced was creating the search function for each playlist.

Conclusion:

Overall, the design of the music manager application emphasizes flexibility, design, and maintainability. By using ADTs, interfaces, and classes effectively, this application provides a great way for managing song data and interacting with users through a user-friendly interface.