**BSHCIFSC2 - Data Structures & Algorithms**

**Continuous Assessment (CA) 50% of overall module**

**Author: Morris Ouedraogo X20461702**

**MusicInt:** The interface defines the contract for interacting with the music collection. It includes methods like **isEmpty**, **isFull**, **push**, **pop**, **size**, **emptyStack**, **enqueue**, and **dequeue**.

**MyStack class:** This class implements the **musicInt**. It utilises the **ArrayList** to maintain a stack structure for storing the songs the user inputs and their corresponding genres. The stack is utilised using the **push** and **pop** methods. The **enqueue** and **dequeue** methods are also provided for additional functionality although they are not typically associated with a stack data structure.

**ManageApp class:** This class contains the **main** method and serves as the entry point for the application. It initializes the GUI and makes it visible to the user.

**MusicGUI class:** This class extends **javax.swing.JFrame** and represents the graphical user interface of the application. It provides components for adding songs, displaying categorized songs, and performing actions such as removing songs and displaying total song count. The GUI interacts with the **MyStack** class to manage the music playlist.

**Design Summary**: The design follows a simple model-view-controller architecture.

The **MusicGUI** class acts as the view, responsible for displaying the user interface and handling user interactions. The **MyStack** class serves as the model managing the underlying data structure for storing songs and genres. The controller aspect is not explicitly defined but is embedded within the **MusicGUI** class where action listeners are used to handle user input and trigger corresponding operations on the data model.

The method of managing data involves using a stack data structure to store songs and genres. Each song and its corresponding genre are pushed onto the stack together. Additionally, songs can be categorised into specific genres (pop or RnB) and moved from the main playlist to genre-specific playlists using the **moveBtnActionPerformed** method in the **MusicGUI** class.

Overall the design provides a basic framework for managing a music playlist through a graphical user interface. However there are areas for improvement such as encapsulating data more effectively, separating concerns more clearly and problems I encounter during the development of the project was setting the playlist to be on repeat. I did not fully understand what exactly the brief wanted the playlist to do in this section. I conducted some research on YouTube to allow me this to do this but unfortunately I had no time to complete this part of the project, A way I could of sorted this issue out would have been starting the project on time.