

**NAME: ALISHBA ARHAM**

**REGISTRATION NO: BSE-B-030**

**COURSE: OOP**

**SUBMITTED TO: SIR SHAHID BHATTI**

**PROJECT: CHESS GAME IN JAVA**

***Title: Java Music Player Project***

**Abstract:**

First of all, a login page appears that works to login into the playlist. After that,username and password option appears that is different for every user. It is helpful for privacy and it works like a Spotify.

The project involves the development of a sophisticated music player application using Java and a graphical user interface (GUI). This music player offers a comprehensive set of features aimed at enhancing the user's music listening experience and organizational capabilities. Users can efficiently manage their music library, create personalized playlists, and exercise full control over audio playback.

**Packages used in this:**

* import javafx.application.Application;
* import javafx.scene.Scene;
* import javafx.scene.control.Button;
* import javafx.scene.control.Label;
* import javafx.scene.layout.BorderPane;
* import javafx.scene.layout.HBox;
* import javafx.stage.Stage;

**Functions used:**

* **setBackground(Color.WHITE):** This function will set the background color of the UI component.
* **setLayout(layout):** This function will set the layout of the frame or panel. Layout can be grid, flow, gridbag, etc.
* **setText(“your text”):** This function will set the text of the label, button, etc.
* **setVisible(true):** This function will set the frame/window to be visible to the user. By default, it is false.
* **setSize(int width, int height):** This function is used to set the size of frame, panel, etc. It takes two parameters such as width and height.
* **setTitle(“any text”):** This function is used to set the title of the java music player window.
* **add(obj):** This function is used to add the component object in frame or panel.
* **setCurrentDirectory(new File(“C:\\Users”)):** This function is used to set the default directory path of the file chooser.
* **setDialogTitle(“any text”):** This function is used to set the title of the file chooser window.
* **setFileSelectionMode():** This function is used to set file selection mode of file chooser such as files only, folder only.
* **setFileFilter():** This function is used to set the file extension allowed to be chosen by the user.
* **getSelectedFile():** This function gets the selected file by the user in the application memory.
* **getName():** This function returns the name of the selected music file.
* **getPath():** This function returns the directory path of the selected file.
* **play():** This function is used to play the mp3 music.
* **close():** This function is used for pausing & stopping the mp3 music in java mp3 music player.
* **skip():** This function will skip that particular part of the mp3 music file at the time of playing. Use at the time of resume() function.

Key functionalities of the music player include traditional playback controls such as play, pause, stop, and skip, along with the ability to adjust volume levels. Users can create, edit, and delete playlists, add or remove songs from playlists, and save or load playlists for future use. The application also features a robust search functionality, allowing users to quickly find specific songs or albums within their library.

The graphical user interface (GUI) of the music player is designed to be intuitive and user-friendly. It includes interactive elements such as buttons, menus, and display areas for song information and album art. The interface provides a visually appealing experience while ensuring ease of navigation and functionality access.

Optional features can be incorporated to further enrich the application. These may include an equalizer for adjusting audio settings, a lyrics display feature synchronized with the playback, visualizations such as waveforms or spectrum analyzers for a dynamic visual experience, and integration with online music sources or platforms for streaming capabilities.

Challenges in developing this project include implementing robust audio playback functionality using Java libraries such as JavaFX or Java Sound API. Efficient management of playlists and music files within the application is crucial for a seamless user experience. Designing a responsive and visually appealing GUI that caters to user interactions and preferences is another key aspect.

By working on this project, developers can gain valuable skills in GUI design using JavaFX or Swing, file handling for managing music files and playlists, event handling for user interactions, and multimedia programming for audio playback and control. The project offers a practical and hands-on learning experience while delivering a functional and enjoyable music player application for users.