Java Project

Student Name: Panashe Madakasi

Student ID:30000916

Date:2/12/2020

Contents

[Implementation 1](#_Toc58177032)

[A detailed explanation of what the program is required to do, 1](#_Toc58177033)

[Analysis 2](#_Toc58177034)

[Test Data and Evidence 4](#_Toc58177035)

# Implementation

### A detailed explanation of what the program is required to do,

-Must contain dynamic data structures (e.g. doubly linked list or a binary tree)

A doubly linked list is used to store the Objects that holds the song names data and the song path data.

- Must contain hashing techniques

A Hashing technique is used on the login page, the stored password is hashed, and the user typed password is hashed and then compared to the stored on.

- Must contain sorting algorithm

A Bubble sort is used as the sorting algorithm, the sorting algorithm is used to sort the song name and paths, this can be seen when the user clicks the sort button the names in the table view will be sorted.

- Must contain searching technique

A binary search technique is used to search through the song name data, this can be used by typing the song name and clicking the search button.

- Must contain 3rd party library

The 3rd party library used is the csv reader, the csv reader is used to read the csv file which contains the song names and song path

- Must have a GUI

With the use of scene builder which is an easy GUI builder and the javafx libbers two GUIs where created the login page and the music player, scene builder allows for easy drag and drop of GUI tools.

- Must adhere to coding standards

Through out the program the java code conventions were used.

Java Code Conventions:

<https://www.oracle.com/technetwork/java/codeconventions->150003.pdf

- Must have help files

Help files are provided with the help button with will open a html page with test to assist the user with using the program

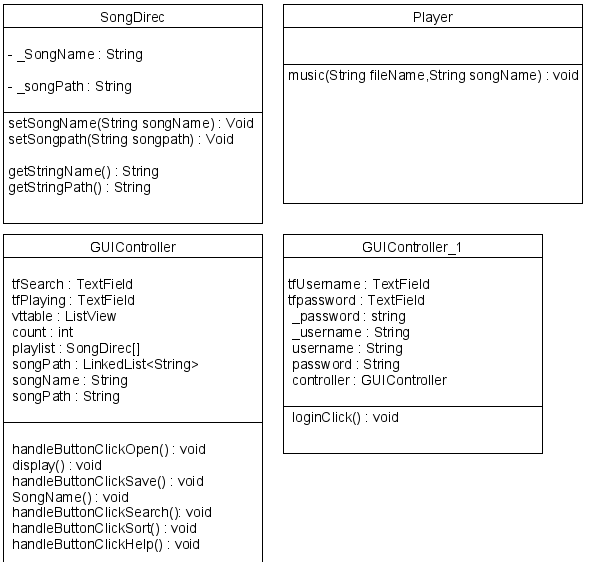
# Analysis

A statement and explanation for each of the following;

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Music Player   |  | | --- | | LoginClick() | | Input | Process | Output | | User name – user input  Password – User input | The user name and password are matched and the password is hased | If the passwords are matched the music player will open |  |  | | --- | | handleButtonClickOpen() | | Input | Process | Output | | CSV File | Information from csv file is stored in an Song Object which saves the song name and song file | Song names are dispayed in a table view and music starts playing |  |  | | --- | | display() | | Input | Process | Output | | Song object | Song names stored in the song object are displayed in the table view | Songs are displayed in table view |  |  | | --- | | bubblesort | | Input | Process | Output | | An array of unsorted random numbers | This bubble sort method sorts through the array of numbers, while the array is being sorted a timer is in use to time how long it takes to sort | The output is the time the method took to sort through the array and the new sorted array |  |  | | --- | | handleButtonClickSearch() | | Input | Process | Output | | Songs from song object  And the song the user is searching | User typed song is searched through the song object | The song searched will start playing on the music player |  |  | | --- | | hashing | | Input | Process | Output | | The user’s username and password typed | The user’s username and password are matched with a hashed username and password in the client | If the users username and password are matched the music player will be presenting. |  |  | | --- | | HandleButtonClickHelp() | | Input | Process | Output | | Button click | Opens a html help file | A html file is opened which contains help notes to help the program |  |  | | --- | | HandleButtonClickSave() | | Input | Process | Output | | Button click | Saves name and path | Writes a csv file with the song name and path |  |  | | --- | | SongName () | | Input | Process | Output | | Playlist object | Loops song names and paths into array | Loops song names and paths into array | |

Algorithm Design

A UML diagram.



# Test Data and Evidence

|  |  |
| --- | --- |
| The login screen for music player which applies hashing |  |
| The music player will open once have logged in |  |
| When the user presses the open button a dialog window appears allowing the user to choose a csv file |  |
| When the csv file has been chosen the songs will display in the table view |  |
| if the user presses the save button the songs and the song paths are saved into a csvfile |  |

User guide

|  |  |
| --- | --- |
| The User choses between the three inputs which corresponds with three different sorting algorithms, depending on how much numbers there are the sorting algorithms will take different amounts of time to sort through them | C:\Users\student\AppData\Local\Microsoft\Windows\INetCache\Content.Word\q3wrong input.PNG |
| If the user inputs (A)  The bubble will begin to sort the array 100 times and display the times on the console | C:\Users\student\AppData\Local\Microsoft\Windows\INetCache\Content.Word\q3 bubble sort.PNG |
| If the user inputs (C)  The bubble will begin to sort the List 100 times and display the times on the console | C:\Users\student\AppData\Local\Microsoft\Windows\INetCache\Content.Word\q3 mergesort.PNG |
| If the user inputs (b)  The bubble will begin to sort the List 100 times and display the times on the console | C:\Users\student\AppData\Local\Microsoft\Windows\INetCache\Content.Word\q3 built in sort.PNG |