CST2550 Coursework 2

Dr Barry D. Nichols

July 18, 2019

1 Brief Task Description

Design and implement a Karaoke application which allows searching a large number of songs by title and adding songs to a 'play-list'. When playing it should play the video of the songs in the play-list in order.

You should consider the most appropriate data structure(s) and algorithm(s) to implement these features. Your design, including analysis of the time complexity of your solution, will be presented as a written report.

2 Submission

You must submit a single **PDF** file of your report and a single **zip** file of your source code by **Friday, 24th April 2020**.

Note:

- **Only source code should be included in the zip file, not IDE files**
- **Code must compile and run on the Linux lab setup, code which does not compile on the lab setup will achieve a *maximum* of 40%**
- **As anonymous marking will be applied, you should *not* include your name in your source code or report**

3 Scenario

A company are launching a new karaoke business and they require software which allows customers to search for songs and add them to the play-list.

At initial release, it is expected that there will be several million songs, and the number will continue to grow with time.

The list of songs (and karaoke videos) will be sourced separately, so you do not need to focus on creating these (a sample song list and single video file will be provided for development and testing). This data should not be hardcoded in your program and it should be possible to load another file instead of this sample file without recompilation (e.g. using a command line argument).

The song list file will be formatted as one song per line, and each line will be a with the format that should be loaded into the program. The file will have details of one song per line, and will include

- song title
- artist
- playing time
- video file name

You can assume that no two songs have the same title.

The application should have an intuitive user interface which allows the user to search for a song by title and add songs to the end of the play-list.

The user should be able to play, pause, stop and skip the songs in the play-list. When a song is played it should be removed from the play-list. When the song is playing its video should be displayed in the GUI window.

The most common functions are

- search song in library (by title)
- add song to play-list
- play songs in play-list (in order)
- add songs to library

therefore, performance should be taken into consideration.

It should also be possible to delete a song from the play-list, but as users normally simply skip the song when it starts playing, this will be used far less often, so performance is not as important.

4 Detailed description

It is recommended that you complete the tasks in the following order as the later sub-tasks will require the earlier ones.

4.1 Song class

Implement a class to hold the details of a song, each line from the data list will be used to initialise an object of type 'Song'.

4.2 Design and Implement Logic

Select appropriate data structures and design the class(es) and methods to use these data structures to achieve the required functionality:

- add song(s) to song library
- search song library for a song by title
- add song to end of play-list
- play next song in (and remove from) play-list
- retrieve songs in play-list (to view)
- delete song from play-list

When selecting data structure(s) to use in your solution, you should consider the time complexity, especially the most frequently used functionality. Analysis of the time complexity should be included in your report with your design.

You should then implement your design and test it with the sample data.

4.3 Graphic user interface

Design and implement a GUI using JavaFX. The GUI should make use of the class(es) you implemented to enable required functionality of the system.

The GUI will have an intuitive user interface to execute the functionality previously mentioned. It will also display the video when playing, along with necessary controls to play, pause and skip songs.

5 Report Layout

- Abstract
  - A paragraph describing the work, results and conclusions
- Introduction

  - brief description of what the paper will be about **(not the coursework task specification)**
  - brief (paragraph) describing the layout of the rest of the report

- Design, including
  - pseudo code
  - analysis of time complexity of solution
  - GUI mock-ups/wireframe diagrams (not screen shots of final application)

- Testing
  - description of the testing approach(es) used
  - evidence of testing, e.g. testing result tables

- Conclusion
  - summary of work done
  - limitations of your approach and critical reflection of your work
  - how would you approach a similar project differently in future

- References
  - any references here should have matching in-text references
  - any work which is not your own must be referenced
  - using the Harvard reference system

- Appendices
  - Code listings of the class(es)

6 Academic Misconduct

This is individual work and you should complete it yourself. You should not work as a group and each submit the same work (even with minor changes) as your own. Any material or ideas found online, in textbooks, etc should be properly referenced.

You should familiarise yourself with the university's academic integrity and misconduct policy:

https://www.mdx.ac.uk/about-us/policies/university-regulations

7 Extenuating Circumstances

There may be difficult circumstances in your life that affect your ability to meet an assessment deadline or affect your performance in an assessment. These are known as extenuating circumstances or 'ECs'. Extenuating circumstances are exceptional, seriously adverse and outside of your control. Please see link for further information and guidelines:

https://unihub.mdx.ac.uk/your-study/assessment-and-regulations/extenuating-circumstances

8 Marking

The report and included code will be marked according the to attached marking scheme. Marking will be **anonymous**, i.e. the marker will not see the name of the student while marking.

9 Feedback

Provisional marks and written feedback will be available on Moodle within 15 working days of your submission. If you would like clarification or more detailed feedback on your coursework contact your module tutor.

10 Marking Scheme

10.1 Report

Item	**Marks**
Abstract	5
Introduction	7
Design (including time-complexity analysis)	10
Testing	8
Conclusion	10
Layout and clarity of writing	5
Correct referencing	5

10.2 Code

Item	**Marks**
Code quality (e.g. comments, naming, layout, exception handling, etc.)	5
Search for song (by title)	8
Add song to play-list	7
Play song (play video of next song in play-list and remove song from play-list)	8
Controls while playing (skip to next song, pause, stop)	5
Display play-list	5
Delete a song from play-list	5
Intuitive user interface	7