Marketo Music

1 Introduction

Marketo Music (MM) is a bare bones music playlist manager. MM allows a user to create and manage multiple playlists by reusing a central store of songs.

2 Data Structures and General Functionality

2.1 Central Store

You will need to store all the song data in a single structure called the Central Store. The songs will be read from a text file that lists the title and artist of each song.

You should create a class called `SongFile` that will hold the properties for each song, i.e. title, artist, `songId`. `songId` should be unique and allow the user to look up a song's associated data in O(1) time.

2.2 Sorting

MM will allow the user to sort based on `songId`, title, and artist.

2.3 Searching

MM also supports searching by artist, where songs by a particular artist should be printed out sorted by title in lexicographic order, and by title, that will print ranges of songs.

2.4 Playlists

Lastly, you must maintain a data structure that tracks an arbitrary number of playlists. You will support create, edit, insert songs into, remove songs from, and printing of the playlist. At the most basic level, a user can create an empty playlist; at which time, the playlist will be assigned a unique `playlistId`. Users can add songs to the playlist by uniquely identifying a song by its `songId`. For an addition bonus, you can allow the user to add a range of songs by searching by artist or title (detailed in 3.3.3).

Songs are unique within a playlist; therefore, you must not allow the user to add a song that already exists in the same playlist. All playlists an be indexed in O(1) time by their `playlistId`. For an additional bonus, you can make the playlist sortable by `artist` and `title` (detailed in 3.3.5).

3 Interface

3.1 SongFile

Create a class called `SongFile` to store information about each song. You will use the constructor to create the `SongFile` objects as you read them from the input file. The input file will have the following format:

```
<artist>, <title>
<artist>, <title>
<artist>, <title>
```

For example: songs.txt

```
Matt Nathanson, Mission Bells
Matt Nathanson, Headphones
Hozier, Take Me to Church
Imagine Dragons, Radioactive
```

3.2 Main Menu

From the Main Menu, you can execute a number of commands. Output the Main Menu on first execution of the application and prompt before reading a command.

3.2.1 Create Playlist

- Command: `create`
- Syntax: `create` <playlist name>
- Actions taken:
  - Add an empty playlist to the playlist data structure with <playlist name> as the name. <playlist name> may be multiple words.
  - Go to the Playlist Menu with the newly created playlist in context

3.2.2 Edit Playlist

- Command: `edit`
- Syntax: `edit` <`playlistId`>
- Actions Taken:
  - Go to the Playlist Menu with the playlist indexed by the `playlistId` in context

3.2.3 Print Song

- Command: `song`
- Syntax: `song` <`songId`>
- Actions Taken:
  - Display the contents of the `SongFile` for that song

3.2.4 Print Playlist

- Command: `playlist`
- Syntax: `playlist` <`playlistId`>
- Actions Taken:
  - Display the artist and title for each song in the playlist in its current order

3.2.5 Print All Songs or Playlists

- Command: `print`
- Syntax: `print` <print option>
  - <print option> can either be `song` or `playlist`
- Actions Taken:
  - If the option is `song`: display the contents of the `SongFile` for each song in the central store. Be sure to display the songs in the current order of the display list
  - If the option is `playlist`: display the artist and title for each song in each playlist in its current order

3.2.6 Search

- Command: `search`
- Syntax: `search` <search option><"string of words">
  - <search option> can either be `title` or `artist`
  - Example: `search title "Take Me"`
  - Search strings are case insensitive
- Actions Taken:
  - Look for the title or artist that contains the input search string
  - Print the contents of the `SongFile` for each matching record

3.2.7 Sort

- Command: `sort`
- Syntax: `sort` <sort option>
  - <search option> can either be `title` or `artist`
  - Example: `sort title`
- Actions Taken:
  - Print the sorted `SongFile`s in lexicographic order based on the sort option

3.2.8 Quit

- Command: `quit`
- Syntax: `quit`
- Actions Taken:
  - Clean up and quit the application

3.3 Playlist Menu

3.3.1 Delete Song

- Command: `delete`
- Syntax: `delete` <songId>
- Actions Taken:
  - Delete the song from the current playlist. All songs with higher indices in the playlist are shifted up one index.

3.3.2 Insert Song

- Command: `insert`
- Syntax: `insert` <songId>
- Actions Taken:
  - If the song already exists in the playlist, notify the user as such
  - Otherwise, append the song to the end current playlist

3.3.3 Insert Search (Bonus)

- Command: `insert`
- Syntax: `insert_search` <search option><"string of words">
  - <search option> can either be `title` or `artist`
  - Example: `insert_search artist "Matt Nathanson"`
- Actions Taken:
  - Search for the given option that contains the input search string and append the results of the search to the current playlist

3.3.4 Print Playlist

- Command: `print`
- Syntax: `print`
  - Notice that this is the same command as in the Main Menu with the exception of a mission parameter. The key is to know that you're in the context of the playlist instead.
- Actions Taken:
  - Display the artist and title for each song in the current playlist in its current order

3.3.5 Sort and Search (Bonus)

- Sort and Search are available from the Playlist Menu as well and are implemented as described in 3.2.6 and 3.2.7.

3.3.6 Return to Main Menu

- Command: `main`
- Syntax: `main`
- Actions Taken:
  - Exit the Playlist Menu
  - Print the Main Menu and prompt for user input

4 Logistics

Your application will be based on a number of factors. Code compiles (a must!), functionality (that it does the intended behavior correctly), completion, cleanliness, readability, robustness, optimization, and test suite (we suggest writing your own input file in addition to the given one) will all be used to determine your score.

You score will be deducted based on the number uncompleted features along with bugs in implemented features. Scores may be adjusted up or down based on other factors as well.

The bonus questions are not required but will boost your score if they correctly do their intended functions. Unlike the other features, it is all-or-nothing per feature. If the feature does not work as intended or has any bugs, it will be disregarded and will not add or subtract from your final score.

You have 48 hours to complete this coding challenge. Good luck!