Lab5 documentation

1. Resources

For this lab I imported the CSV files from Spotify Charts. For the weeks of 10/15/2020 to 10/21/2020; 10/08/2020 to 10/14/2020 and 10/01/2020 to 10/07/2020 in the United States. I took off the header.

2. Prerequisites

In order to be able to use this program efficiently, the user must use a csv file (without the header) that follows the chart:

Position	Track name	Artist	Streams
…..	…..	….	…

The user must have four files (*week1.csv, week2.csv, week3.csv* and exampleOutput.txt)) located in the same directory as the Lab5.java file.

3. Implementation

Problem description:
A local Radio DJ wants to create a sorted playlist based on the song title name. He is in a coding class learning about data structures and tries to create the playlist using a binary search tree.

How I wrote the program:

First, I wrote a class Song which takes a songTitle(track name), streamsAverageCount (average of streaming of the song) , artistName, artistAverage (average of appearance of the artist within three weeks), and an address of the next song as instances. Then, I created a class Playlist (linked list) that represent a playlist, which takes a First (the address of the first element of the Linked list) and which has some methods such as:

- ***addSong:*** to add a song in the play list
- **isEmpty:** to check whether the list is empty or not
- **find:** to check if a song title exist in the play list
- **findAposition:** to get the index of an element inside the playlist (used for the **subset** methode)
- ***sortList():*** to sort the elements of the list by ascendant order (A.....Z) in order to get binary search tree (ordered array or ordered Linked list).
- ***display(PrintStream ps):*** to print the play list songs inside the output file (exampleOutput.txt)
- ***subset(Start , end, PrintStream ps) :*** to select song titles that fall alphabetically between start and end.

  Second, Like **Lab4** I created a class MyQueue which inherits from Playlist, it creates a linked list that stores songs from an array of text file. Then, I wrote a function called Filter which takes a string (a line of the csv file) and then filter it (the line) to get the track name, artist name, and the streams for the current song then return it.

  I created two Hasmaps streams and artist which help the total number of streams for each song during the three weeks and the number of times an artist appear during that period. Then I used those two hasmaps to calculate the **streamsAverageCount and artistAverage**

Hint: There is no redundancy inside MyQueue (it stores a track only once).