CSCI-290 *Project Part 10*

Choose Your own Data Structure

Objectives: Searching by lyrics phrase (making data structure choices)

In this last assignment we will add another search class **SearchByLyricsPhrase**. Its search method will return an array of all the songs that contain the phrase. The result will be ordered by ranking, so that the best matches are shown first.

In program 9 you wrote a ranking function for ranking a single song, and then tested it by ranking every song. If you want an easy solution you could do the same for this assignment. If it works correctly this will receive full credit. The challenge for this assignment is to make searching faster through use of data structures. Is there a way you could avoid ranking every song? Is there a data structure that you could build in the search class‘ constructor to make the actual searches faster? How can you quickly order the songs by rank? You are not required to use your ranking function from program 9. It was assigned to get you thinking about the issues involved in ranking. On the other hand, if it works, weigh how much time it might take to make something better versus how busy you are at the end of the semester.

In the Part10 kit, there are two files. One is full of test data and results. The other is actually a tester for your program with timers built in. You should use P10timer.java as the driver to test this part. I will be using it to grade. The elapsed time for all tests should be well under 1 second.

Task 1: SearchByLyricsPhrase

- Create a new search class SearchByLyricsPhrase.
- As with the other searches, the constructor will take a SongCollection as its argument. This time, however, you get to decide what data structures to build. You might design something simple, something elaborate, or reuse previous data structures. It is your choice. The only requirements are that the searches are accurate and reasonably fast.
- Implement the method public Song[] search(String lyricsPhrase) whose argument is a string containing the lyrics phrase. Searches are case insensitive, and all words must be matched, even if they are common words.
- Your search method should return an array of matching songs ordered by rank: best matches first.

Task 2: Testing

- Add a main() to your search class, similar to those in the previous assignments. As before the first command line argument is the name of the song file, and the second argument is now the lyrics phrase. In NetBeans, to pass in more than one word as a single command line argument you will need to enclose the lyrics phrase in quotes.
- Your testing routine should print out the total number of matches, and the RANK, ARTIST, and TITLE for the first 10 matches. Test your program thoroughly. I will

be using the testing code in P10timer.java when I grade your programs.

- Sample results: The lyrics phrase search "she loves you" should output:

```
Total songs = 58, first 10 matches:
rank artist title
13 Aerosmith, "Beyond Beautiful"
13 Beatles, The, "She Loves You"
13 Beatles, The, "That Means A Lot"
13 Billy Joel, "Modern Woman"
13 Blondie, "Little Girl Lies"
13 Elvis Presley, "Take Good Care of Her"
13 Elvis Presley, "The Meanest Girl in Town"
13 Fleetwood Mac, "Silver Springs"
18 Bonnie Raitt, "Any Day Woman"
21 Pink Floyd, "Apples And Oranges"
...
```

  As in program 9, your rankings may vary slightly due to differences in the way you dealt with end of line characters.

- Run these test cases and show the results in your report:

"she loves you"	should find 58
"love love love"	should find 1760
"school's out"	should find 45

Task 3: Works with the GUI

- Check that this new search works with the GUI.

What to turn in:

Written part:

1. Describe your solution and any data structures that you used
2. Turn in your search results from Task 2.
3. Any incomplete parts, or any known bugs in your code.

Electronic submission:

- Submit your write up part10.pdf file with all team members listed.
- Zip the entire project folder this time and submit it.

Grading:

Written explanation of your ranking algorithm and ranking results	20 points
Correctly calculating rankings	80 points

I expect your programs to be properly commented and use good style. Points may be deducted for egregious disregard of these matters. Every method in every class must have complete documentation including authorship. A full an complete revision history should be at the top of every class file!