CSCI-290 *Project Part 6*

Ragged Array List - Search by Title Prefix

Objectives: Using your ragged array list to quickly search the songs by title prefix). Now we are going to use the Ragged Array List to organize our songs by title and then search them by title prefix.

Task 1: Song Comparator by title

- Add a new Comparator to your song class that compares songs by title.
- It should ignore capitalization.

Task 2: SearchByTitlePrefix

- Create a new class SearchByTitlePrefix. This will be very similar to your SearchByArtistPrefix except that it will use a RaggedArrayList. Most of the work is already done by your RaggedArrayList. You are just using its methods.
- It should have a constructor that takes in a SongCollection and uses the data from it to build a RaggedArrayList of songs ordered by title. Note: you are not rereading the input file! You are just taking the array of songs held in the SongCollection and adding those songs into a RaggedArrayList. Both data structures will reference the same underlying song objects.
- The ragged array list will use your song Comparator by title.
- Write a search() method that takes a title prefix (such as "Angel") and returns an array containing all of the matching songs. This will use the subList() and toArray() methods of your RaggedArrayList.
- To do a prefix search on a range, the fromElement should be your prefix and the toElement should be the next successive string that doesn't match the prefix. This is obtained by incrementing the last character, so in this case "Angem". You will need to create a couple of dummy songs containing the 2 title prefixes for your call to subList().

Task 3: Testing

- Similar to main() method used for testing SearchByArtistPrefix, create a main() method to test your SearchByTitlePrefix class . It should take two arguments: the song file and the search string. It should use SongCollection to read the file. Then it creates a SearchByTitlePrefix object, passing in the songCollection. Next if performs the search specified by the command line arguments. Finally it should print the total number of matches and the first 10 matches.
- In NetBeans change the Run configuration and add the two arguments: allSongs.txt "search"
- Perform searches for "search", "Angel" and "T". (The total matches should be 2, 23 and 1148 respectively.) You will turn in these results as part of your report.

Task 4: Statistics

- Now we are going to count how many comparison are used to build the ragged array of 10514 songs. This will allow us to verify that it meets our performance expectations.
- Make your new title Comparator extend the CmpCnt class and increment cmpCnt each time it is called.
- In your search by title prefix constructor, store a reference to the Comparator as a class variable. This will allow your main() testing routine, which is part of this class, to access that Comparator to get its cmpCnt information.
- In your main() method print the cmpCnt after the ragged array list has been built.
- Compare this measured value to what you would expect from a mathematical analysis and explain why the measured value is or is not consistent with the mathematical analysis. Clearly explain your answer. For those who used binary search be sure to explain that.

Task 5: GUI

- Your new search method should automatically plug into and work with the GUI. Check that it works.

What to turn in:

Written part:

1. Include all member names and indicate which one did the electronic submit.
2. Turn in a printout of the results from the 3 test cases (search, Angel, and T) as printed by your testing code.
3. Your comparison count and explanation from Task 4.
4. Does it work with the GUI?
5. Any incomplete parts, or any known bugs in your code.

Electronic submission:

- I will create groups for you to self enroll in. Anyone in the group can upload files.
- Submit Song.java SearchByTitlePrefix.java and your report.
- Do not in any way combine, compress, zip, tar or jar your files!

Grading:

Written part report and test results	10 points
Task 1 Song Comparator by title	10 points
Task 2 SearchByTitle prefix	20 points
Task 3 Testing Code	20 points
Task 4 Statistics and analysis	20 points
Task 5 works with GUI	20 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!