ECE 325 LAB Assignment 5: Sorting your collection

With a better understanding of the cool things you can do with Collections in Java, you decide to once more improve the implementation of your SongCollection (actually, you decide to write it from scratch again, so no need to look at the old code). Hopefully, this is the last time, but as you probably know, there are no guarantees in the life of a neoclassical jazzhopmetal artist.

- You want to use this SongCollection to easily filter duplicates in your input data. Duplicate songs (i.e. those with the same title) should end up only once in the SongCollection.
- Also, you want to use this SongCollection to easily sort your songs in the following ways:
    - By their natural ordering: ordered by title from A-Z
    - By an alternative ordering: ordered by average rating from high to low (and then for songs with equal ratings, by number of votes from high to low)

Based on these requirements, you decide to use a TreeSet instead of an ArrayList as the internal data structure for the SongCollection. In this assignment, you will implement the application.

Implement the application

1) Let Song implement the correct interface for usage in a TreeSet and finish the implementation for the sorting. The natural ordering of songs is by their titles (from A-Z).

2) Implement the loadSongs() method in the SongCollection class. You can load the songs directly into the collection. Your implementation must use a BufferedReader and a Scanner. The songs and their ratings are in the songratings.txt file (one song per line) in the following format:
Title;rating;votes
So, in comparison to the previous assignment, no need to compute the average rating – you can load it directly from the file. Make sure that your program doesn't break when there are malformed inputs in the input file. For this assignment, it is OK to just silently ignore any malformed inputs – your program should not cause any unhandled exceptions.

3) Sometimes, we want to see the songs ordered by their average rating (of course with the highest rated song first). For two songs with the same rating but with a different number of votes, the song with the highest number of votes should be ranked the highest. Finish the implementation of the sort() method (and add any class(es) you may need).

4) In the future, you may need to store the songs in a HashSet. So you have to prepare the implementation for that as well (= add the right methods, see the lecture slides for which ones). Implement the methods that are required to use your Song class in a HashSet.

5) Let the main() method (in SongCollection) of your program do the following:
- Create a new SongCollection
- Load the songratings.txt file
- Print the songs in the song collection (using their natural ordering, from A-Z)
- Print the songs in the song collection (ordered by average rating as described in (3))

- Demonstrate that you can use your Song class in a HashSet (you can just call demonstrateHashSetUsage() for this – you don't have to change this method but make sure that the output is correct though!).

Hints:
- Make sure to implement your comparison code in the right classes, and to reuse as much functionality as possible from existing classes.
- You can assume that there are never records in the txt file that have the same title but different ratings/votes (there can be completely duplicate records though, or records that do not have the right format).

Rubric

(20 points total)

Please submit:
1) A zip file containing your code and a PDF with the answers to the questions above.
Name the file 'FirstName_ID_lab_asg5.zip' and keep the exact same file structure as the zip that was provided for the assignment. So, for example:
Filename: Cor-Paul_1234567_lab_asg5.zip
| songratings.txt
|------- src
| |------- ece325
| | |------- labs
| | | |------- lab5
| | | | |------- *.java

2) A screencast/movie that shows the following steps:

- Open your eClass with your name shown
- Open your IDE
- Show your code briefly
- Execute your code and demonstrate that your program works correctly (= show the execution of the main() method).

Please submit the screencast as a **separate** file to eClass.

Please do not modify any of the names/methods we've defined in the provided *.java files.