CSC120 Week 13-14 Lab: Song List

Instructor: Mitsu Ogihara

The goal of this project is to write an application for maintaining a list of songs. Each song has two pieces of information, its title and artist.

The application allows to add data from file, save data to file, search in the data for songs with a key phrase appearing either in the title or in the artist, add new song, delete a specific song, and change title/artist of a specific song.

The application consists of three class files:

1. `Song`: the class for an individual song
2. `SongCollection`: the class for a song collection.
3. `SongMain`: This is the main class. The program execution is by way of the command `java SongMain`.

A song data file takes the following form.

- The first line is the number of songs stored in the data file.
- After the first line, the songs appear in two lines each, the first line being the title and the second being the artist.

For example, the following is a data file consisting of 5 songs:

```
5
Like A Rolling Stone
Bob Dylan
Satisfaction
The Rolling Stones
Imagine
John Lennon
What's Going On
Marvin Gaye
Respect
Aretha Franklin
```

The same for format should be used when writing to a file.

To read from a file, you use the `nextLine` method of `Scanner`. You can obtain the integer that a `String` data, say `w`, represents by calling `Integer.parseInt( w )`.

The class `Song`

Naturally, we want to use two `String` instance variables to record the title and the artist. The constructor for the class may take two `String` values and store them in their respective instance variables. There are two methods, both of which are getters, that need to be implemented:

```
public String getTitle()
public String getArtist()
```

The class must implement the interface `Comparable`. The declaration should be

```
public class Song implements Comparable<Song> { ... }
```

The method `public int compareTo( Song o )` is the method needing implementation. The method compares the title first. If the titles are different (that is, the result of `title.compareTo( o.title )` is not 0), the method returns the result as is; otherwise, it returns the result of `artist.compareTo( o.artist )`.

The class `SongCollection`

In this class, we need just one private instance variable:

```
Song[] theSongs;
```

The methods to be implemented are:

```
public int size()
public void addFromFile( File f )
public void writeToFile( File f )
public void addOneSong( String t, String a )
public void delete( int pos )
public void searchByTitle( String key )
public void searchByArtist( String key )
public void show( int start, int end )
```

The expected actions of these methods are as follows:

1. The method `size()` returns the number of elements in the array `theSongs`.
2. The method `addFromFile( File f )` reads data from file, in the following manner.
   (a) First, it creates a new scanner to read from `f`. It encases the creation in `try-catch` so that if `FileNotFoundException` occurs, then the method prints an error message and returns immediately.
   (b) Second, the method reads the first line of the file and converts it to an integer `count`. All the data files to be used by the program, have the number of songs in the first line. The acquisition of `count` is accomplished by supplying the first line to the method `Integer.parseInt`.

(c) Third, the method creates a new array of `Song` objects whose number of elements is equal to the current number of elements plus `count`. This can be achieved by

    `Song[] temp = Arrays.copyOf( theSongs, theSongs.length + count );`

(d) After that, the methods reads data from the file in pairs of lines and stores the data in the new slot.

(e) The resulting array may have duplicates. We need to eliminate duplicates. We accomplish this as follows.

   i. We first sort the elements or `temp` using `Arrays.sort` (make sure you import `java.util.Arrays`).

   ii. From the sorted list, we build an array consisting of all elements `temp[ i ]` such that
   `(*) \code{ i == 0 || i > 0 && temp[ i ].compareTo( temp[ i - 1 ] ) != 0 }`
   The latter condition mean that the element at `i` is not identical to the element at `i - 1`. For example, let us consider a sorted integer array `1, 2, 2, 4, 5, 5, 5`. Here, the second occurrence of `2` and the last two occurrences of `5` are duplicates. The list resulting from duplicate removal is thus `1, 2, 4, 5`.

   iii. The construction will create a new array `merged` of `Song` objects. It uses a position variable `pos` whose initial value is 0. Then we iterate over the elements of `temp` with an iteration variable `i`. If the condition (*) holds, we store the element `temp[ i ]` to `merged[ pos ]` and then add 1 to `pos` so that the next time such a value of `i` is found, the element goes into the next slot.

   iv. After the scanning is done, we execute
   `theSongs = Arrays.copyOf( merged, pos );`
   to extract the unique elements we have stored in a prefix portion of `merged`.

3. The method `writeToFile( File f )` writes the data to the file `f`. The same error handling as in `addFromFile` is needed.

4. The method `addOneSong( String t, String a )` adds a song specified by `t` as the title and `a` as the artist. The way the method works is very similar to the way `addFromFile` does. The differences are (a) the increase in the array length is 1 and (b) the new element shall be stored at the end of the new array.

5. The method `delete( int pos )` deletes the element at position `pos`, if the value of `pos` is valid.

6. The method `searchByTitle( Sting key )` prints all the songs whose title contains `key` along with their index values. You can use the method `indexOf( key )` on the value returned by the method `getTitle`.

7. The method `searchByArtist( Sting key )` prints all the songs whose artist contains `key` along with their index values. You can use the method `indexOf( key )` on the value returned by the method `getArtist`.

8. The method `show( int start, int end )` prints all the songs whose index values are greater than or equal to `start` and strictly smaller than `end`. Adjustments of the values may be needed when `start < 0` or when `end > theSongs.length`.

The class `SongMain`

This consists of one method, which is `main`. The method presents to the user command choices, receives the choice of command by a number (you can use `Integer.parseInt( keyboard.nextLine())`, where `keyboard` is the `Scanner` object you instantiate in your program as the `Scanner` to read from the `keyboard`). You can then use a switch statement to respond to the choice made.

Here is an execution example of the program.

```
% java SongMain
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 4
Enter file name: songs-data.txt
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 2
Enter title search key: Heaven
30: Stairway To Heaven, Led Zeppelin
189: Knocking On Heaven's Door, Bob Dylan
352: Tears In Heaven, Eric Clapton
409: Monkey Gone To Heaven, Pixies
482: Just Like Heaven, The Cure
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
```

```
6. Add one song
7. Remove one song
8. Show
Enter choice: 3
Enter artist search key: Sabbath
249: Paranoid, Black Sabbath
309: Iron Man, Black Sabbath
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 4
Enter file name: Sabbath Bloody Sabbath
*** File Not Found ***
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 6
Enter title: Sabbath Bloody Sabbath
Enter artist: Black Sabbath
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 2
Enter title search key: Blood
```

```
267: Sunday Bloody Sunday, U2
413: Young Blood, The Coasters
500: Sabbath Bloody Sabbath, Black Sabbath
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 7
Enter position: 267
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 1
***
*** Size = 500
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 8
Enter start position: 10
Enter end position: 25
10: My Generation, The Who
11: A Change Is Gonna Come, Sam Cooke
12: Yesterday, The Beatles
13: Blowin' In The Wind, Bob Dylan
```

```
14: London Calling, The Clash
15: I Want To Hold Your Hand, The Beatles
16: Purple Haze, Jimi Hendrix
17: Maybellene, Chuck Berry
18: Hound Dog, Elvis Presley
19: Let It Be, The Beatles
20: Born To Run, Bruce Springsteen
21: Be My Baby, The Ronettes
22: In My Life, The Beatles
23: People Get Ready, The Impressions
24: God Only Knows, The Beach Boys
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 5
Enter file name: foo.txt
========Select action========
0. Quit
1. Get collection size
2. Search for title
3. Search for artist
4. Add from file
5. Save to file
6. Add one song
7. Remove one song
8. Show
Enter choice: 0
```

Attached to this assignment are three text files

- `songs-data.txt`: The Rolling Stone Top 500 song of all time.
- `songs-beatles.txt`: The list of songs recorded by The Beatles.
- `songs-zeppelin.txt`: The list of songs recorded by Led Zeppelin in their official albums.

Here is the best strategy to accomplish the goal:

Step 1 Write `Song.java`.

Step 2 Write the bare-minimum version of `SongCollection.java`, which consists of the constructor, the `implements`, the private instance variable declaration, and an implementation of the method `size`. The rest of the required methods can have empty body, i.e.,

```
{
  ;
}
```

in this version.

Step 3 Write the bare-minimum version of `SongMain`. In the bare-minimum version, the program uses a do-while loop, in which the presents the command choices to the user, receives input from the user, and then uses a switch-statement with which the execution is directed but the action is yet to be typed except for the `break` at the end of each case. Make sure that the loop terminates if the user enters 0 for the action.

Step 4 Add actions one after another by editing both `SongMain` and `SongCollection`.