CS 211 Lab Assignment

Instructor: Dan Fleck, Ricci Heishman
Lab: Create a text High Score list and text menu to display them

Overview

In most video games you can save a high score and name. The top 10 people are shown with their “top score”. In this lab you will create a data class to hold an individual high score (an int) and the name of the person (a String).

You will also create a driver file that loads the high scores from a data file, displays a text-based menu with options to display the high score list and add a new high score to the list. Adding a high score to the list will automatically save the list to the data file. Your high score list also must be sorted.

Using your high score objects, you will also create a menu to add items to the list and to read/write a data file with the items.

1. Save new score
2. Display scores
Enter option or '0' to quit :

Sample output:

```
1. Save new score
2. Display scores
Enter option or '0' to quit :
2


High Scores
-----------
Hanke 12121
Mike 7777
Archie 3450
Charlie 2898
Zeke 1271


1. Save new score
2. Display scores
Enter option or '0' to quit :
1
Enter name:
```

```
Dan
Enter score:
5555
1. Save new score
2. Display scores
Enter option or '0' to quit :
2


High Scores
-----------
Hanke 12121
Mike 7777
Dan 5555
Archie 3450
Charlie 2898
Zeke 1271


1. Save new score
2. Display scores
Enter option or '0' to quit :
0
```

Data File Format
The data file should be a text file with formatted lines of text like this:

Sample Data File (scores.dat):

```
Hanke~12121
Mike~7777
Dan~5555
Archie~3450
Charlie~2898
Zeke~1271
```

When you read and write the file, you must parse these Strings to create HighScore objects from each line.

Specific Hints

How to sort your array of objects
In Java there are many support classes to do various things. In this lab you will have an array of 10 high scores. (NOT an ArrayList!) In order to sort this array, you should use the Java API class “java.util.Arrays”.

When sorting Objects from this class, there must be a natural ordering. Ints and String have a natural ordering (numeric and alphabetic). However your class

(HighScore) does not have a natural ordering by default – you must provide one. Objects in Java present a natural ordering by implementing the Comparable interface. This interface tells Java you have specified the natural ordering by implementing the compareTo method. So, for your HighScore class you must make sure you are Comparable so that sorting using the Array class works effectively. See information on pg 322, the example on page 508 and the Javadocs for Comparable for more info.

How to read and write a file

In Java, data is usually written and read as a stream. Reading files using an input stream and writing files using an output stream. To learn how to use Java's Standard API to read and write files, see:

http://java.sun.com/docs/books/tutorial/essential/io/index.html

HINT: Don't forget to write the newline! \n

General Hint: You're going to have an array with 100 entries, but only 5-10 of them have values. Thus, you need to keep track of the end of the array data so you can sort only that portion of the array, and know where to add new information. In this lab array.length() is not useful for anything.

Requirements

1. Create a class HighScore that holds two data values a name (String) and a high score (int).

2. HighScore shall use good encapsulation (i.e. all attributes MUST be private)

3. HighScore shall have an appropriate toString method to ensure when you print a HighScore, it displays well.

4. HighScore shall be Comparable so you can sort it.

5. HighScore should not allow any attributes to be set after the object is constructed.

6. The driver file shall read a formatted data file and create an array of HighScore objects. You may assume the file is always there. You do not need to check for a missing file. NOTE: You must create an array, not an ArrayList!

   HighScore [] scores = …

7. The driver file shall present a text based menu to the user to allow them to display the list on the screen or add a new entry

8. The list displayed by the driver file MUST be sorted with the highest score on top

9. When a new entry is added, the driver file must ask the user for a name and a score (using text input)

10. When the user enters a name and score, the driver must automatically add that to the list and save the list.

Assumptions you can make

- The user will always type in valid values (don't need to error check)
- The data file will always be present with at least one line in it
- There will never be more than 100 names in the high score list (i.e. you're safe with an array of 100 HighScores)

What to turn in:

1. A Jar file containing all Java source code and compiled code you used. (Include code that was given by the professor and your own code.
2. Output of the Jar command that shows the contents of your Jar file. Run this on the command line. (Should be done anyway to make sure you're including the source code).

Grading Rubric: This assignment is worth 10 points and will be graded based on the following rubric:

Area	Exemplary	Competent	Developing	Points
Class Header	All header components are present, with references and comments that accurately support the state of the file.	All header components are present, but references and comments are incomplete or nonspecific.	Header is missing or only partially present, and references and comments are vague or unmeaningful.	__ / 1
Coding Style	Code implementation utilizes appropriate white space, self-documentation techniques and non-obvious comments.	Code implementation exhibits minor alignment or spacing problems, some comments are missing or redundant.	Significant alignment and spacing problems, comments are generally missing or sporadic.	__ / 1
HighScore class	The HighScore class is present, has a single constructor to store the information. Has all private variables and uses accessors appropriately. Additionally, it has toString and implements Comparable.	The HighScore class is missing some parts of the specification as described above.	Much of the HighScore class is missing or is done in a different way than specified.	__ / 4
Driver Class	The driver class follows the requirements and specification above fully. Including reading/writing files, properly displaying a text based menu, and correctly sorting and sorting the file in the format described above,	The driver class is missing some parts of the specification as described above.	Much of the driver class is missing or is done in a different way than specified.	__ / 4