CMPS 261: Project 7

Due Date: Check on Moodle

Instructions

1) You are allowed to discuss the problem and solution design with others, but the code you submit must be your own. Your solution **must** include the certification of authenticity as given in the course syllabus.
2) **Your code must include JavaDocs comments (see additional requirements 1).**
3) Follow the instruction on Moodle under "Submitting to Moodle".

General Requirements:

- Project submissions not including a Certification of Authenticity will not be graded until such a written and signed submission is provided by the student.
- Code, even if substantial, that fails to compile will not receive a grade higher than 25%
- Code, even if substantial, that fails to run will not receive a grade higher than 50%
- A penalty of up to 10% may be assessed for lack of required documentation, lack of descriptive identifiers, lack of indention of blocks of code, failure to follow naming requirements of the project and associated tar archive.
- Failure to submit the complete IntelliJ project may result in a penalty of up to 10%.

Problems

1) (80 Points) Create an IntelliJ Java project named "p71_yourulid". Download the file "fstein.txt" from Moodle and place it in the project folder. Write a Java program that reads the words in "fstein.txt" and displays all the non-duplicate words in ascending order. Your solution must use a class from the Java API that implements interface Set as an integral part of the solution.

2) (120 Points) Create an IntelliJ Java project named "P72_yourulid". Write a program that prompts the user to enter a year in the range [2001, 2010], a gender and a name, then displays the ranking of the name for the selected year and gender.

Technical Requirements:

a) The data for this project is to be read from by the program directly from the following URLs (DON'T store these files locally):

http://liveexample.pearsoncmg.com/data/babynameranking2001.txt
http://liveexample.pearsoncmg.com/data/babynamesranking2002.txt
http://liveexample.pearsoncmg.com/data/babynamesranking2003.txt
http://liveexample.pearsoncmg.com/data/babynamesranking2004.txt
http://liveexample.pearsoncmg.com/data/babynamesranking2005.txt
http://liveexample.pearsoncmg.com/data/babynamesranking2006.txt

http://liveexample.pearsoncmg.com/data/babynamesranking2007.txt
http://liveexample.pearsoncmg.com/data/babynamesranking2008.txt
http://liveexample.pearsoncmg.com/data/babynamesranking2009.txt
http://liveexample.pearsoncmg.com/data/babynamesranking2010.txt

b) The data files consist of lines of data, each containing a ranking (an integer), a boy's name (a string), the number of boy's that were given that name (an integer), a girl's name (a string) and the number of girls that where given that name (an integer). Values are separated by spaces and/or tabs, which java.util.Scanner will use as separators by default.. Here is an example of the first 10 lines of one of the files (babynamesranking2002.txt):

```
1 Jacob 30541 Emily 24450
2 Michael 28220 Madison 21771
3 Joshua 25965 Hannah 18802
4 Matthew 25142 Emma 16520
5 Ethan 22099 Alexis 15629
6 Andrew 21996 Ashley 15335
7 Joseph 21872 Abigail 15292
8 Christopher 21665 Sarah 14741
9 Nicholas 21380 Samantha 14652
10 Daniel 21291 Olivia 14627
```

c) Using java.util.Scanner to read from a URL: Create a java.net.URL instance object with the URL string, then pass the return of the URL method openStream() to the constructor of Scanner. For example:

```
try (java.util.Scanner input
  = new java.util.Scanner(new java.net.URL(
   "http://www.cs.armstrong.edu/liang/data/babynamesranking2010.txt"
      ).openStream())) {
```

d) The data is to be stored in two arrays of maps, one for boy's names and one for girl's names. Each array must have one element for each of the 10 years of data. Each element in the array is a map (a class from the Java API that implements interface Map) that stores key / value pairs, each pair consisting of a name and its ranking, with the name serving as the key.

Additional Requirements

[1] Doc comments are required for all public, protected and package classes, constructors and methods. JavaDocs must be generated and submitted as part of the project submission.

[2] Identifiers must be descriptive, i.e. must self-document. The only exception granted is in the case of a "for variable", that is a variable created as a simple counter as in the control variable in a "for" loop statement.

[3] Indention of all code blocks (compound statements, anything in braces), including single statements following selection or while statements, is required. IntelliJ will do this fairly automatically as you type if your syntax is correct. In IntelliJ, Ctrl+Alt+L will re-format a whole file if your syntax is correct.

[4] The main ".java" file [the one with the method *public static void main(String[] args)*] of your project must contain this minimal documentation:

```
// Your Name
 // Your CLID
 // CMPS 261
 // Program Description: description of actions of code
 // Certificate of Authenticity:   (Choose one of the two
 following forms:)

     I certify that the code in the method functions
      including method function main of this project are
      entirely my own work.

 {or}

     I certify that the code in the method functions
      including method function main of this project are
      entirely my own work., but I received some assistance
      from {name}.  Follow this with a description of the
      type of assistance. (For example, if you consulted a
      book, and your solution incorporates ideas found in the
      book, give appropriate credit; that is, include a
      bibliographical reference.)  Note:  You do not have to
      list the text, the author of the text or the
      instructor's examples.
```