PROGRAMMING ASSIGNMENT 4

Due: Thursday, February 27th at 11:59pm

This assignment will provide you experience working with

- Obtaining and using real-world data
- File input/output
- Exceptions
- Auto web page creation (ok, not a huge deal)

Name popularity

The Social Security Agency has kept track of baby names based on Social Security card applications since 1879 and through the end of February 2014. All of the data is available on line and there are webscripts for searching through the data. Please read http://www.ssa.gov/OACT/babynames/background.html for additional information about the names.

The top 1000 names are available on the web, and in this assignment you will write a program that works similarly to 'Popularity of a Name' feature but using all baby names (or names with at least 5 cards issues for the year.) (See http://www.ssa.gov/oact/babynames/#ht=2).

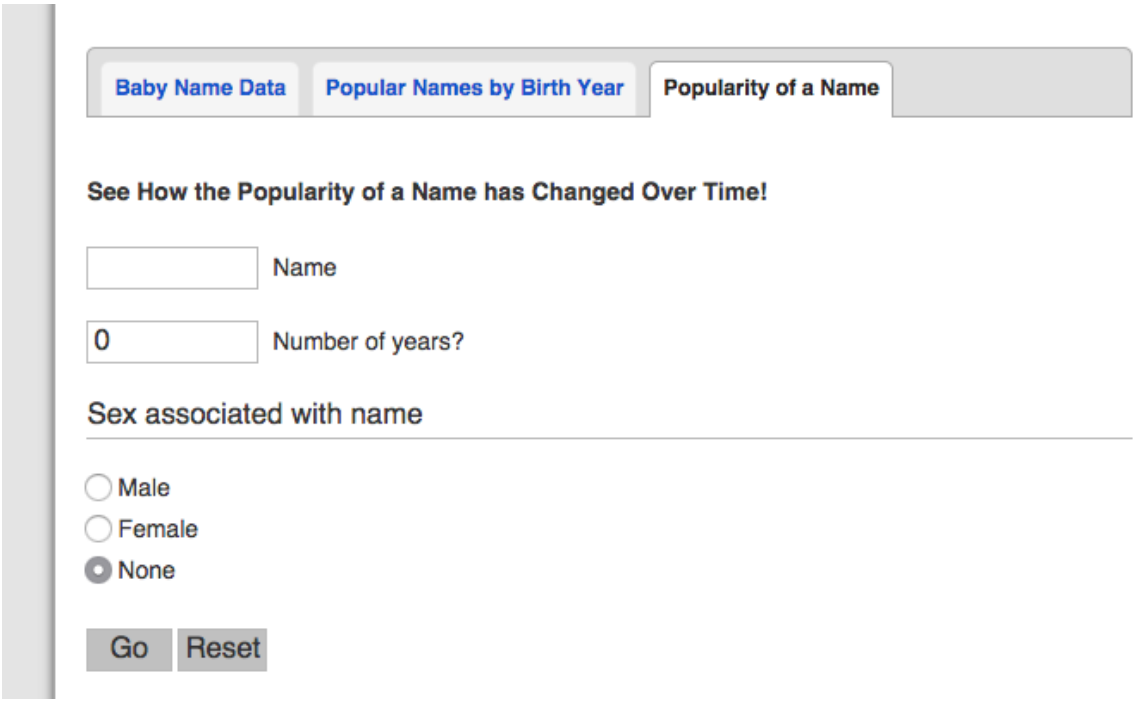

Figure 1. User Input

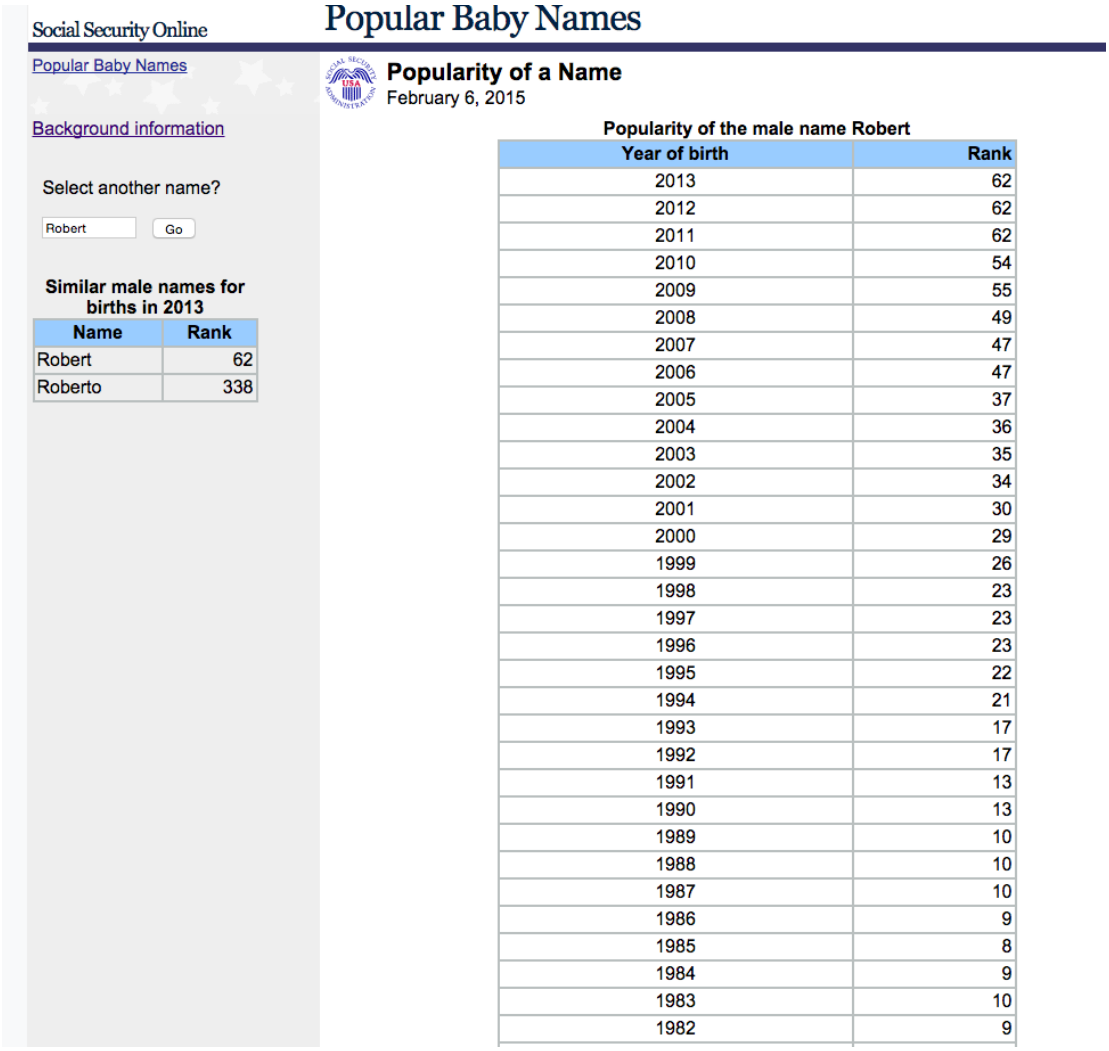

Figure 2. Output

Assignment

In this assignment, you will download the national data of SSN names from http://www.ssa.gov/OACT/babynames/limits.html. Your program should read in the files and allow retrieval for names, number of births, gender and rank for each year. You will design the structure(s) and storage for this information. Please prompt user for the pathname to the files. You should use exceptions to ensure that the files are readable and exist.

You should then design and develop a Java program that prompts a user for a Name, number of years to go back, say we call it `yearsBack`, and sex. Your program should then print out the name's popularity for the past `yearsBack` years. You should then prompt the user to try another name, without changing sex or years, and or prompt the user to exit. You must properly use conditionals to validate/verify user input.

You should write out the Name, on one line, and then Year and Rank for each year to the screen similar to the following:

```
Name? Rebecca
Years back? 10
M/F? F



        Rebecca          Years: 10
---------------------------------

        Year                     Rank
        2013                     1773
        2012                     2004
        2011                     2107
        2010                     2280
        2009                     2457
        2008                     2902
        2007                     3212
        2006                     3556
        2005                     3995
        2004                     4450
```

Figure 3. Output Example

Agile Design and Implementation

You will be designing your solution for this assignment. Below are guidelines for your 'Sprints'

*Sprint 1 - Read in the file and store it (I suggest using an ArrayList of ArrayLists_*

Good OO design will make this much easier to do. This is the lion's share of the work. It will be helpful for you to

- Review ArrayList, iterators and focus on Scanner and File in Chapter 12.
- Use JUnit to test that you are reading in the files correctly. You can check by printing on the screen, querying for a year's data.
- Be sure to verify path name as a directory and file names prior to trying to read in code! Erroneous path and/or files should raise exceptions.
- There are two options to reading in the file

1. Work with file as a file of Strings Read in a line at a time and then use one of the String methods to split the line up into its fields (name, M/F, number of births)
2. Use more than one Scanner object, using Regular Expressions to define your delimiters (Liang, 10.10.4).

Sprint 2 – Robust and correct User Input Queries

You must reasonably validate user input and design for known user errors. Do not use Exceptions for validating user input. The 'easiest' way to do this is with regular expressions to simplify this process. (See http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html) You will get extra credit if you validate using regular expressions.

- Name
  - Only alphanumeric input (no blanks) allowed
- Years Back
  - Verify years back is in the range of data
    - If too large, just print out back to 1884
    - If negative or 0, allow for new input
- Sex – Use the term used by SSA
  - Male, Female or None options
  - When None is selected at the SSA site, it defaults to male. This is the easiest thing to do, although not the right thing to do. The correct solution is to keep a total for all names independent of M/F, however for this assignment when None is selected then print out one showing the name results for male and one for female.

Sprint 3 – Formatting output

- Use either String.format() and/or System.out.printf()
- Neatness will count, so have your ranks line up right justified
- This is not hard, but it is critical to your projects success.

You should keep in mind that for full credit:

- Good OO Design including
  - toString() and equals() overridden when appropriate
  - Include JUnit tests for each class

Submission

Your submission should be 1 zipped file uploaded to blackboard containing:

- UML Class diagram as a PDF file
- All *.java files

Extra Credit Options:

Print out the table in a web page named “Name”.html where “Name” is the name the user gives you. You should use valid HTML5 tags and a decent style (use open source CSS style sheets). Include a screen shot of your web page in your delivery as well as the .css file if you are using an external style sheet. You must handle exceptions for writing to a file.

Use regular expressions for your data validation