Joanna Klukowska
joannakl@cs.nyu.edu

Project 1: Inheritance and Baby Names

Due date: February 13, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own . Any sharing or copying of assignments will be considered cheating (this includes posting of partial or complete solutions on Piazza, GitHub or any other public forum). If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own). You are responsible for every line in your program: you need to know what it does and why. You should not use any data structures and features of Java that have not been covered in class (or the prerequisite class). If you have doubts whether or not you are allowed to use certain structures, just ask your instructor.

In this project you will provide a tool for visualizing popularity of baby names in New York State in the last decade. Your program will use the open data set provided by the Department of Health that contains names given to babies born in NYS each year. Using this data and a name specified by the user, your program will need to generate a histogram showing the fraction of children who were given that name in each year.

Objectives

The goal of this programming project is for you to master (or at least get practice on) the following tasks:

- working with multi-file programs
- reading data from input files
- working with large data sets
- using the `ArrayList` class
- writing classes
- working with existing code
- extending existing classes (inheritance)

Most, if not all, of the skills that you need to complete this project are based on the material covered in cs101. But there might be certain topics for which you did not have to write a program or that you forgot. Make sure to ask questions during recitations, in class and on Piazza.

Start early! This project may not seem like much coding, but debugging always takes time.

For the purpose of grading, your project should be in the package called `project1`. This means that each of your submitted source code files should start with a line:

```
package project1;
```

Dataset

In this project you will be working with open data. Wikipedia has a good description of open data: ”Open data is the idea that some data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control.”

The data set that you need can be found at New York State Open Data portal. For your convenience a csv file is also located on the course website.

The file that you download is CSV (comma separated values) file - it is a simple text file and is processed like text file (but it can also be opened by most of the spreadsheet programs and displayed column-wise based on the locations of commas on each line).

The file is organized as follows. The first line in the file contains the column headings. Each consecutive line (except for the first one) contains the data:

```
YEAR, NAME, COUNTY, GENDER, COUNT
```

(see the NYSDOH_NYSBabyNames_DataDictionary.pdf file for the description of format and restriction of each of the fields).

Joanna Klukowska
joannakl@cs.nyu.edu

User Interface

Your program has to be a console based program (no graphical interface) - this means that the program should not open any windows or dialogs to prompt user for the input.

Program Usage

The program is started from the command line (or run within an IDE). It expects one command line argument.

This program should use command line arguments. When the user runs the program, he/she will provide the name of the input file (containing data formatted exactly as described above) as a command line argument. (This way the user can specify a data set from another state or a different version of the data set).

The user may start the program from the command line or run it within an IDE like Eclipse - **from the point of view of your program this does not matter**.

If the name of the input file provided as a command line argument is incorrect or the file cannot be opened for any reason, the program should display an error message and terminate. It should not prompt the user for an alternative name of the file. If the program is run without any arguments, the program should display an error message and terminate. It should not prompt the user for the name of the file. The error messages should be specific and should describe exactly what happened, for example:

```
Error: the file names.txt cannot be opened.
```

or

```
Usage Error: the program expects file name as an argument.
```

The above error messages generated by your code should be written to the `System.err` stream (not the `System.out` stream).[1]

Input and Output

The program should run in a loop that allows the user to check popularity of different names in different counties. On each iteration, the user should be prompted to enter a name (for which the program computes the results) or the letter 'Q' or 'q' to indicate the termination of the program.

The user should not be prompted for any other response.

If the name entered by the user cannot be found in the list of names stored in the dataset, the program should print a message

```
No such name in the dataset.
```

to the output stream and the program should continue into the next iteration.

Histogram format:

If the name entered by the user is found in the dataset for at least one year, a histogram showing the popularity of this name should be displayed.

For each year's data, the program needs to determine the percentage of babies given a particular name

$$\frac{\text{number of babies with a given name}}{\text{total number of babies in that year}} \times 100$$

For each year there should be a line of output matching the following format:

```
YYYY (F.FFFF): HISTOGRAM_BAR
```

where

- `YYYY` is a four digit year,

[1] If you are not sure what the difference is, research it or ask questions.

Joanna Klukowska
joannakl@cs.nyu.edu

- `F.FFFF` is a percentage calculated according to the above formula (it has to be printed with exactly one digit before the decimal point and four digits after the decimal point[2]),
- `HISTOGRAM_BAR` is a visual representation of the percentage; it should consist of a sequence of vertical bars '|', one bar for each 0.01 percent (rounded up to the nearest integer); the number of bars can be calculated by

$$\left\lceil \frac{\text{number of babies with a given name}}{\text{total number of babies in that year}} \times 10000 \right\rceil$$

  where the symbol $\lceil \ldots \rceil$ means the ceiling function (in Java you can use `Math.ceil()` to compute it, see https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#ceil-double-)

Here are the example lines calculated for 'Joanna' for three different years:

```
2007 (0.0129): ||
...
2010 (0.1989): ||||||||||||||||||||
...
2015 (0.0616): ||||||
```

(Note, that the program should not be printing ..., but rather the information for each year.) There should not be any blank lines between the lines for each year.

A few things to consider:

- For a name that does not occur in a particular year, the count should be set to zero, which will result in no vertical bars printed.
- For a name that is the most popular in a given year, the number of bars may exceed the width of the display window and (depending on the settings) may wrap to the next line. This is fine and you do not need to modify this behavior.
- Some names may occur in the data file more than once (since they are given to both female and male babies). Your histogram should combine the data for both occurrences.
- The program should be case in-sensitive. The name in the data file is always capitalized, for example 'Joanna'. Your program should produce exactly the same results regardless if the user types 'joanna', 'JOANNA', 'JoAnNa' or any other variation of cases.[3]
- Some names have several different spellings in common use, for example 'Joanna', 'Johana', 'Johanna'. For the purpose of this program these are considered to be completely different names.
- Any lines in the input file that do not contain the required five fields (year, name, county, gender and count) should be silently ignored and the program should move on to reading the next line from the file.

Sample interaction:

```
asia@zeppo$ java -cp bin project1_jk.BabyNames ../csci102/projects/project1/Baby_Names__Beginning_2007.csv
Please enter a name:    Jayden
2007 (1.0629): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2008 (1.2191): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2009 (1.2283): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2010 (1.2364): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2011 (1.2287): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2012 (1.1426): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2013 (1.0173): ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2014 (0.8155): ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2015 (0.4731): ||||||||||||||||||||||||||||||||||||||||||||||||
Please enter a name:    Isabella
2007 (1.1042): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2008 (1.0452): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2009 (1.1427): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2010 (1.2535): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2011 (1.2365): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2012 (0.9413): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2013 (1.0955): |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2014 (0.9306): ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
2015 (0.5542): ||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

[2] You should be using `printf` for that. If you are not familiar with formatted output in Java, research it or ask questions.

[3] You may want to explore the methods of the `String` class that ignore the case when comparing two objects.

Joanna Klukowska
joannakl@cs.nyu.edu

```
Please enter a name:    Gerard
2007 (0.0000):
2008 (0.0000):
2009 (0.0000):
2010 (0.0000):
2011 (0.0000):
2012 (0.0000):
2013 (0.0059): |
2014 (0.0072): |
2015 (0.0035): |
Please enter a name:    Smoke
No such name in the dataset.
Please enter a name:    Sabrina
2007 (0.0939): ||||||||||
2008 (0.1041): ||||||||||
2009 (0.1270): ||||||||||||
2010 (0.1132): ||||||||||
2011 (0.1210): ||||||||||||
2012 (0.0811): |||||||||
2013 (0.0505): ||||||
2014 (0.0728): ||||||||
2015 (0.0376): ||||
Please enter a name:    q
```

Data Storage and Organization

Your need to provide an implementation of several classes that store the data and compute the results when the program is executed.

In particular, your program must implement and use the following classes. You may implement additional classes as well, if you wish.

`Name` Class

The `Name` class stores information about a particular name for a particular year. It should store information about the name itself, the gender and the count (how many babies have been given that name).

This class should provide a three parameter constructor:

```
public Name ( String name, String gender, int count )
```

If the constructor is called with an empty string for `name`, invalid `gender` indicator (valid values are single characters 'f' for female, 'm' for male in either lower- or uppercase), or a negative value for `count`, then an instance of `IllegalArgumentException` should be thrown carrying an appropriate error message.

There should be no default constructor.[4]

This class should implement `Comparable<Name>` interface. The comparison should be done by the `name` as the primary key (using alphabetical order), by the `count` as the secondary key (i.e., when two objects that have the same value of `name` are compared, the comparison should be performed by `count`) and by the `gender` as the ternary key.

This class should override the `equals` methods. The two `Name` objects should be considered equal if the `name`, `count` and `gender` data fields are identical.

The class should override the `toString` method. The details are up to you, but you should make sure that it returns a `String` object that is a meaningful representation of the object on which it is called.

`YearNames` Class

The `YearNames` class should be used to store all the `Name` objects for a particular year. You should use `ArrayList` class for that purpose (i.e., this class should have a data field of type `ArrayList<Name>`).

[4] A default constructor is one that can be used without passing any arguments.

Joanna Klukowska
joannakl@cs.nyu.edu

The class needs to provide the one parameter constructor

```
public YearNames ( int year )
```

The constructor should throw an `IllegalArgumentException` if the `year` argument is not a four digit positive number between 1900 and 2018 (inclusive).

The class should implement

- `public void add ( Name name )`

  method that adds a new `Name` object to the list. This method should throw an instance of `IllegalArgumentException` if it is called with a `Name` object that already exists (i.e., the `name` argument is equal to an existing element based on the `equals` method for the `Name` class).

- `public int getYear ( )`

  method that returns the year number associated with this object.

- `public int getCountByName ( String name )`

  method that returns the number of babies that were given the `name` specified by the argument. If there are two `Name` objects matching the specified `name` string (one male, one female), the sum of two counts should be returned.

- `public double getFractionByName ( String name )`

  method that returns the fraction of babies that were given the `name` specified by the argument (this is the number of such babies divided by the total number of babies born in the the year).

This class should override the `equals` methods. The two `YearNames` objects should be considered equal if their years are identical.

The class should override the `toString` method. The details are up to you, but you should make sure that it returns a `String` object that is a meaningful representation of the object on which it is called (it may or may not contain the listing of all of the elements).

You may implement other methods, if you wish.

You will need to instantiate one `YearNames` object for each year in the dataset.

`NYSBabyNames` Class

The `NYSBabyNames` class is the actual program. This is the class that should contain `main` method. It is responsible for opening and reading the data files, obtaining user input, performing some data validation and handing all errors that may occur (in particular, it should handle any exceptions thrown by your other classes and terminate gracefully, if need be, with a friendly error message presented to the user).

You may (and probably should) implement other methods in this class to modularize the design.

Given Code

To simplify parsing of the data from the input file, you can download from the course website the code for the following function:

`splitCSVLine` method allows you to parse the lines from the input file into an `ArrayList<String>` object containing all of the entries. This function produces empty strings to represent entries that were blank within the input line.

Programming Rules

You should follow the rules outlined in the document *Code conventions* posted on the course website at https://joannakl.github.io/cs102_s18/notes/CodeConventions.pdf.

The data files should be read only once! Your program needs to store the data in memory resident data structures.

You may not use any of the collection classes that were not covered in cs101 (for this assignment, do not use `LinkedList`, `Stack`, `Queue`. `ColorSet`, `PriorityQueue`. or any classes implementing `Map` interface).

Joanna Klukowska
joannakl@cs.nyu.edu

You may use any exception-related classes.

You may use any classes to handle the file I/O, but probably the simplest ones are `File` and `Scanner` classes. You are responsible for knowing how to use the classes that you select.

Working on This Assignment

You should start right away!

You should modularize your design so that you can test it regularly. Make sure that at all times you have a working program. You can implement methods that perform one task at a time. This way, if you run out of time, at least parts of your program will be functioning properly.

You should make sure that you are testing the program on much smaller data set for which you can determine the correct output manually. You can create a test input file that contains only a few names.

You should make sure that your program's results are consistent with what is described in this specification by running the program on carefully designed test inputs and examining the outputs produced to make sure they are correct. The goal in doing this is to try to find the mistakes you have most likely made in your code.

You should backup your code after each time you spend some time working on it. Save it to a flash drive, email it to yourself, upload it to your Google drive, do anything that gives you a second (or maybe third copy). Computers tend to break just a few days or even a few hours before the due dates - make sure that you have working code if that happens.

Grading

If your program does not compile or if it crashes (almost) every time it is run, you will get a zero on the assignment.

If the program does not adhere to the specification, the grade will be low and will depend on how easy it is to figure out what the program is doing.

40 points class correctness: correct behavior of methods of the required classes and correct behavior of the program

50 points design and the implementation of the required classes and any additional classes

20 points proper documentation, program style and format of submission

How and What to Submit

For the purpose of grading, your project should be in the package called `project1`. This means that each of your submitted source code files should start with a line:

```
package project1;
```

Your should submit all your source code files (the ones with .java extensions only) in a single **zip** file to Gradescope.

You can produce a zip file directly from Eclipse:

- right click on the name of the package (inside the `src` folder) and select Export...
- under General pick Archive File and click Next
- in the window that opens select appropriate files and settings:
  - in the right pane pick ONLY the files that are actually part of the project, but make sure that you select all files that are needed
  - in the left pane, make sure that no other directories are selected
  - click Browse and navigate to a location that you can easily find on your system (Desktop or folder with the course materials or ...)
  - in Options select "Save in zip format", "Compress the contents of the file" and "Create only selected directories"
- click Finish