 Name:

Project 6

 Instructions

***Project 6 NameSurfer***

For this project, we will pull together everything we have seen so far to make a neat application that slices, dices, and graphs data from a big database of names. Project 6 is due thursday April 26th.

Against all bureaucratic stereotypes, the Social Security Administration, provides a neat web site showing the distribution of names chosen for kids over the last 131 years in the US (http://www.ssa.gov/OACT/babynames/).

Every year, the data gives the most popular boy and girl names for kids born in the US. The data for each year is in a text file where each line has the name, followed by the sex, and the number of people born with that name in that year. Looking at a portion of the yob1880.txt we can see:

Mary,F,7065  
Anna,F,2604  
Emma,F,2003  
Elizabeth,F,1939

....

Sarah,F,1288  
Annie,F,1258  
Clara,F,1226  
Ella,F,1156  
Florence,F,1063  
Cora,F,1045  
Martha,F,1040

.....

Carl,M,372  
Lee,M,361  
Howard,M,357  
Martin,M,357  
Michael,M,354  
Bert,M,348  
Herman,M,347  
Jim,M,345

We see that there were 345 Males named Jim in 1880 and 361 Lee's. The database is for children born in the US, so ethnic trends show up when immigrants have kids.

One of the neat things about this project is that it uses real data (and lots of it!). The data just records literally what people put on the forms, so there are things like "A" and "Baby" recorded as names (the data is more cleaned up in the later years). We will not worry about that, and we will not combine names that are similar in some sense - "Cathy" and "Catherine" and "Kathryn" and "Katie" and "Kati" will all count as different names. Looking at the data, long names ("Michael") are becoming more popular compared to their short versions ("Mike").

We will take the top 1000 names for males and females from each file. The earlier files (1880, 1881, etc) only have 1000 female names and 1000 male names. As you go to the later files (2010) there are more than 19000 female names and more than 14000 male names.

To the surprise of no one, the best approach to this somewhat large problem is to decompose the problem into separate classes that can be built up gradually.

• Name - encapsulates the data for a single name (a "data model" object).

• NameCollection - the main class for the whole thing. Stores all the data and coordinates the other classes.

**Part 1  Getting the data loaded**

**Name Loading (15 points)**

The first step is creating the Name class. Each Name encapsulates the data for one name - the name and the number of births over the years. The following methods should be in the Class.

Constructor - takes a String line as in the file above and sets up the Name object.

String getName() - returns the name

 int getRank(int year) - returns the rank of the name in the given year. Use the convention that year=0 is 1880, year=1 is 1881, and so on. The Name constants START=1880 and YEARS=131 define the start year and the number of years of data.

 int bestYear() - returns the year where the name was most popular, using the earliest year in the event of a tie.  It is safe to assume that there will never be more than 150000 births in a single year for a given name.

**NameCollection (10 points)**

Create the NameCollection class to deal with getting the data loaded. Implement a read(String filename) method in NameCollection that reads the given file, looks through the existing Names to see if a record with that name already exists, if not create a Name and initialize the instance variables and set the values for the current year appropriately. NameCollection will have two instance variables male and female of type ArrayList to hold the male and female names respectively. In main() create a new NameCollection instance and send it read("yob1880.txt") - this will, in essence, read the first 1000 female and male names from the file of name data into your two ArrayLists male and female, ready for use (only 130 more files to go!).

**Part 2 - Search (10 points)**

Implement a search(String target) method in NameCollection that takes a string, and searches through all the names for any that contain that substring (not case-sensitive). Print out a line for each matching name to System.out, printing the name followed by its best year. It's fine to search the obvious way - just loop through them all and check with an if-statement.

Implement a doSearch() method that loops, calling showInputDialog() to get a string, and then calls search() with that string to print all the matching names.  Here we do a search with the string "sam" and the names and best years print on the console...

Isamar 1990

Rosamond 1910

Sam 1900

Samantha 1990

Samara 2000

Samir 2000

Sammie 1930

Sammy 1940

Samson 2000

Samuel 2000

**Part 3 - Table and Rankings (20 points)**

Now that all the files have been read, go back through the lists and calculate

a) the total number of names in a given year for a given sex (so how many females and how many males were born each year) (5 points)

b) create a new list inside the name class to hold the percentage multiplied by 10000 of females (or males) with a given name for a given year.  (5 points)

Now create a main method  to display the menu. Each menu option should call a corresponding method in the *NameCollection* class. Consider each of these in turn below:

1. Turn your attention back to the *NameCollection*class. Write a method (e.g. *displayName()* that receives a name, a gender, a start year, an end year and a timestep, then displays the corresponding information from Name. That code must go through the appropriate gender array (either male or female) and find the name, displaying the requested portion of information on the screen. (5 points)
2. Similarly in the *NameCollection* class write a method (e.g. *displayNamesInYear()* ) that receives a gender, year, a starting year, an ending year, a timestep and number of names to display. Make a copy of the appropriate gender array. Sort this copy in descending order based on the designated year. Then display the portions requested from the *n* top names from this array. (5 points)

**Extra Credit**

create a graph that will show the same data as the tables created above (5 points)

create a GUI that the user can use to enter the information to show in the graph (instead of the command line). (5 points)

**Sample Execution**

Author: Prof. Theys  
Program #6: Name Surfer   
April 2, 2012  
  
Welcome to the Name Surfer, where you can explore which names   
have been most common between 1880 and 2010.  
   
  
Reading from data files. Please be patient, this will take a few moments...  
Now creating sorted arrays...  
Sorting females list into alphabetical order...  
Sorting males list into alphabetical order...  
OK, ready.  
   
-----------------------------------------------   
 1. Search the Names for a string and best year   
 2. Search for a name and produce a table of results  
 3. Choose a year and find the n most popular names from that year   
 4. Exit   
   
 Please enter your choice: **1**  
Enter the portion of a name to search for: **sam**  
Searching among the males:  
Samuel 1880  
Sam 1900  
Samual 1882  
Sampson 1898  
Isam 1880  
Sammie 1931  
Samson 2009  
Sammy 1946  
Samie 1907  
Samir 2009  
Samantha 1989  
Searching among the females:  
Samantha 1991  
Rosamond 1911  
Sammie 1933  
Samatha 1989  
Isamar 1990  
Samara 2006  
Samira 2003  
Samiyah 2009  
   
-----------------------------------------------   
 1. Search the Names for a string and best year   
 2. Search for a name and produce a table of results  
 3. Choose a year and find the n most popular names from that year   
 4. Exit   
   
 Please enter your choice: **2**  
Enter the name and gender to find: **John M**  
Enter the starting year to display  
**1890**  
Enter the ending year to display  
**1920**  
Enter the step size  
**2**  
 1890 1892 1894 1896 1898 1900 1902 1904 1906 1908 1910 1912 1914 1916 1918 1920  
 John 7714 7490 7194 6889 6737 6666 6540 6441 6321 6199 6080 5967 6084 5916 5868 5627  
   
-----------------------------------------------   
 1. Search the Names for a string and best year   
 2. Search for a name and produce a table of results  
 3. Choose a year and find the n most popular names from that year   
 4. Exit   
   
 Please enter your choice: **3**  
Enter the gender, year, and number of most popular names: **M 1894 10**  
Enter the starting year to display  
**1880**  
Enter the ending year to display  
**1910**  
Enter the step size  
**2**  
 1880 1882 1884 1886 1888 1890 1892 1894 1896 1898 1900 1902 1904 1906 1908 1910  
 John 8761 8443 8247 8188 7710 7714 7490 7194 6889 6737 6666 6540 6441 6321 6199 6080  
 William 8650 8214 7816 7485 7257 6799 6449 6353 6557 6109 5816 5471 5097 5022 4995 4695  
 James 5378 5205 5002 4857 4637 4624 4473 4462 4407 4392 4913 4625 4652 4519 4652 4882  
 George 4651 4587 4358 4237 4095 4044 3944 3812 3627 4007 3663 3492 3351 3213 3042 2887  
 Charles 4852 4498 4219 4112 3827 3684 3578 3308 3081 2936 2780 2779 2706 2760 2606 2540  
 Joseph 2388 2358 2379 2353 2495 2422 2539 2599 2582 2617 2517 2562 2614 2698 2761 2776  
 Frank 2941 2805 2827 2836 2884 2792 2610 2500 2408 2414 2356 2279 2223 2140 2087 2001  
 Robert 2192 2208 2169 2218 2347 2306 2246 2180 2301 2421 2595 2630 2712 2781 2801 2978  
 Henry 2217 2278 2173 2180 2164 2131 2032 1989 1847 1770 1767 1780 1657 1614 1515 1539  
 Harry 1952 1971 2040 2047 2300 2127 2038 1911 1769 1757 1540 1447 1367 1247 1283 1049  
   
-----------------------------------------------   
 1. Search the Names for a string and best year   
 2. Search for a name and produce a table of results  
 3. Choose a year and find the n most popular names from that year   
 4. Exit   
   
 Please enter your choice: **4**  
Exiting program...