CSE142—Computer Programming I
Programming Assignment #6
due: Tuesday, 5/19/20, 11 pm

Thanks to Nick Parlante of Stanford for the original version of this assignment
also see: http://www.nytimes.com/2003/07/06/magazine/06BABY.html

This assignment will give you practice with file processing in conjunction with all of the other constructs we have learned this quarter (loops, if/else, methods, and graphics). We will be processing a file with data obtained from the Social Security Administration. They provide a web site showing the distribution of names chosen for children over the last 130 years in the US (http://www.ssa.gov/OACT/babynames/).

Every 10 years, the data gives the 1000 most popular boy and girl names for kids born in the US. The data can be boiled down to a single text file as shown below. On each line we have the name (a single token), followed by the sex ("f" or "m"), followed by the rank of that name in 1880, 1890, 1900, ..., 2010 (14 numbers). A rank of 1 was the most popular name that year, while a rank of 997 was not very popular. A 0 means the name/sex combination did not appear in the top 1000 that year at all.

```
...
Irene F 102 60 31 21 17 33 73 92 187 275 292 385 467 696
Fulton M 0 0 0 797 0 0 0 0 0 0 0 0 0 0
Sam M 36 41 34 40 84 110 150 207 228 321 428 396 462 472
Eulalia F 695 768 869 693 715 782 0 0 0 0 0 0 0 0
Ulysses M 317 310 384 413 440 487 464 518 640 787 799 752 772 0
Leigh F 0 0 0 0 0 0 0 658 344 213 218 469 0 0
Abbie F 178 199 316 449 729 806 952 0 0 0 725 685 591 822
Ruthe F 0 0 0 0 736 0 0 0 0 0 0 0 0 0
Napoleon M 303 355 400 402 648 698 826 781 889 0 0 0 0 0
Kadeem M 0 0 0 0 0 0 0 0 0 0 0 538 0 0
...
```

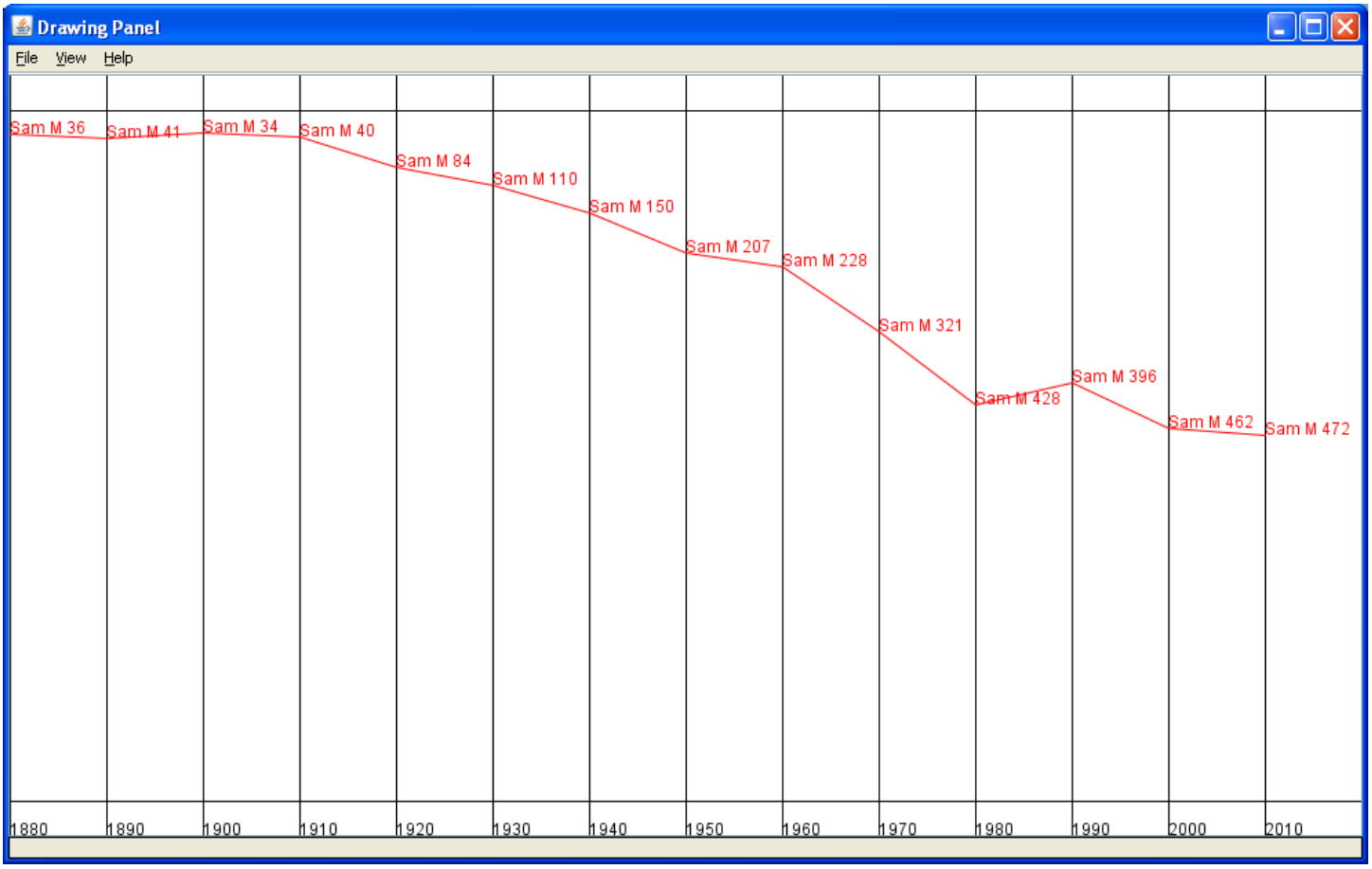
We see that "Sam" was #36 in 1880 and is slowly moving down. "Leigh" popped up on the scene in 1950 and got more popular for a while, but then headed back down. "Kadeem" shows up suddenly in 1990 and then disappears again. "Ulysses" has had an almost steady decline and dropped off the list in 2010.

Your program is to give an introduction and then prompt the user for a name and sex combination to display (remember that names like "Leslie" can be given to both boys and girls). Then it will read through the data file searching for that combination. If it finds it, it should graph the data for that combination. If not, it should generate a short message indicating that the combination is not found. Look at the sample log of execution at the end of this write-up. You are to exactly reproduce this format. Notice that we print a message if the combination is not found, but when it is found, we don't print anything. Instead we graph the data.

If the combination is found, you are to construct a DrawingPanel to graph the data. Don't construct this object until you verify that the combination is in the file. If the combination is not found, you shouldn't construct a DrawingPanel at all. For the cases when the combination is found, you should construct the panel and then give a series of drawing commands that produce output like the following. You are to exactly reproduce this output.

The background color is the standard white (in other words, you don't have to set the background color). There are a series of horizontal and vertical lines drawn in black. The overall height of the panel should always be 550 pixels. The horizontal line at the top should be 25 pixels below the top and the horizontal line at the bottom should be 25 pixels above the bottom. The panel is divided into 14 sections of equal width to represent the 14

different decades for which we have data (starting with 1880 through 2010). These particular sections have a horizontal width of 70 pixels each. You should introduce class constants for the number of decades (14), the starting year (1880) and the horizontal width per decade (70). It should be possible to change these values and have your program adapt appropriately (more on this later). You are to draw a vertical line to the left of each of the 14 sections, including at the x coordinate of 0 (you do not need a vertical line on the far right).



Notice that at the bottom of the panel you need to label each decade. To do so, you will need a new drawing command. The Graphics class includes a drawString method that takes 3 parameters: a String, an x-coordinate and a y-coordinate. The (x, y) coordinate is the position of the lower-left corner of the text. For example, the text "1880" above has coordinates (0, 550) while the text "1890" has the coordinates (70, 550). You will obviously want to write a loop to draw these various labels, especially since your program has to adapt properly if the constants are changed. At some point you will be faced with the problem of turning an int into a String (e.g., how do you turn an int value like 1880 into the String "1880"?). There are two ways to do this. You can call the method String.valueOf passing it the int and it will return a corresponding String or you can concatenate the int with an empty string as in:

```
"" + 1880
```

Then you'll need to plot the actual data for the individual name/sex combination. As noted earlier, the panel will always be 550 pixels high with the upper and lower 25 pixels not part of the plot area. That leaves exactly 500 pixels for plotting these values. The numbers range from 1 to 1000, so each pixel will represent two different rankings. Thus, a rank of 1 or 2 should be drawn at a y-coordinate of 25. A rank of 3 or 4 should be drawn at a y-coordinate of 26. A rank of 5 or 6 should be drawn at a y-coordinate of 27. And so on up to ranks of 999 and 1000 which should be drawn at a y-coordinate of 524. A rank of 0 (which means the name didn't appear in the top 1000 at all) should be drawn at the very bottom of the plot range, at a y-coordinate of 525.

If your code to convert ranks into pixels has a bug, then your lines are likely to be slightly off. You can use the table below to investigate. It shows the different ranks for "Lisa" along with the corresponding pixel values that should be used for the y-coordinate. You can use the jGRASP debugger or a println to see your y-coordinates.

Rank	0	0	0	0	0	0	733	220	6	2	16	64	295	718
y-coordinate	525	525	525	525	525	525	391	134	27	25	32	56	172	383

You are to draw lines connecting the different decades. In addition, just to the right of the line you are to include a string that has the name followed by a space followed by the sex followed by a space followed by the rank. Notice, for example, that the string "Sam M 36" appears to the right of the line for 1880. That is because Sam with sex M had rank 36 in 1880. The text is to appear just to the right and just above the point you are plotting. In other words, you can use the same coordinates for drawString that you use for drawLine. The lines and text for the actual plot should be drawn in red. You should draw the entire black grid first and then draw the red lines over the grid.

You should ignore case when comparing the name and sex typed by the user with the names and sexes in the input file. The strings that you display on the drawing panel should use the name and sex from the input file, even if the user types the name or sex in a different case. For example, if the user asks you to search for "SAM" with sex "m," you should find it even though the input file has it as "Sam M" and the drawing panel should use the input file's "Sam M" rather than what the user typed when it displays the name/sex/rank information in the plot. You should not make any assumptions about the capitalization in the input file. For example, the input file might contain a name with mixed case like "McKinley" and even though this input file uses capital letters for "F" and "M", your program should not assume that will always be the case. You should also not make any assumptions about spacing within a line of the input file and whether or not a line has extra data. This won't be a problem for your program if you use a Scanner to pull apart each line of input (versus, for example, searching for a string that has exactly one space between the name and the F/M sex notation) and as long as you use the constant for the number of decades to determine how many values to process.

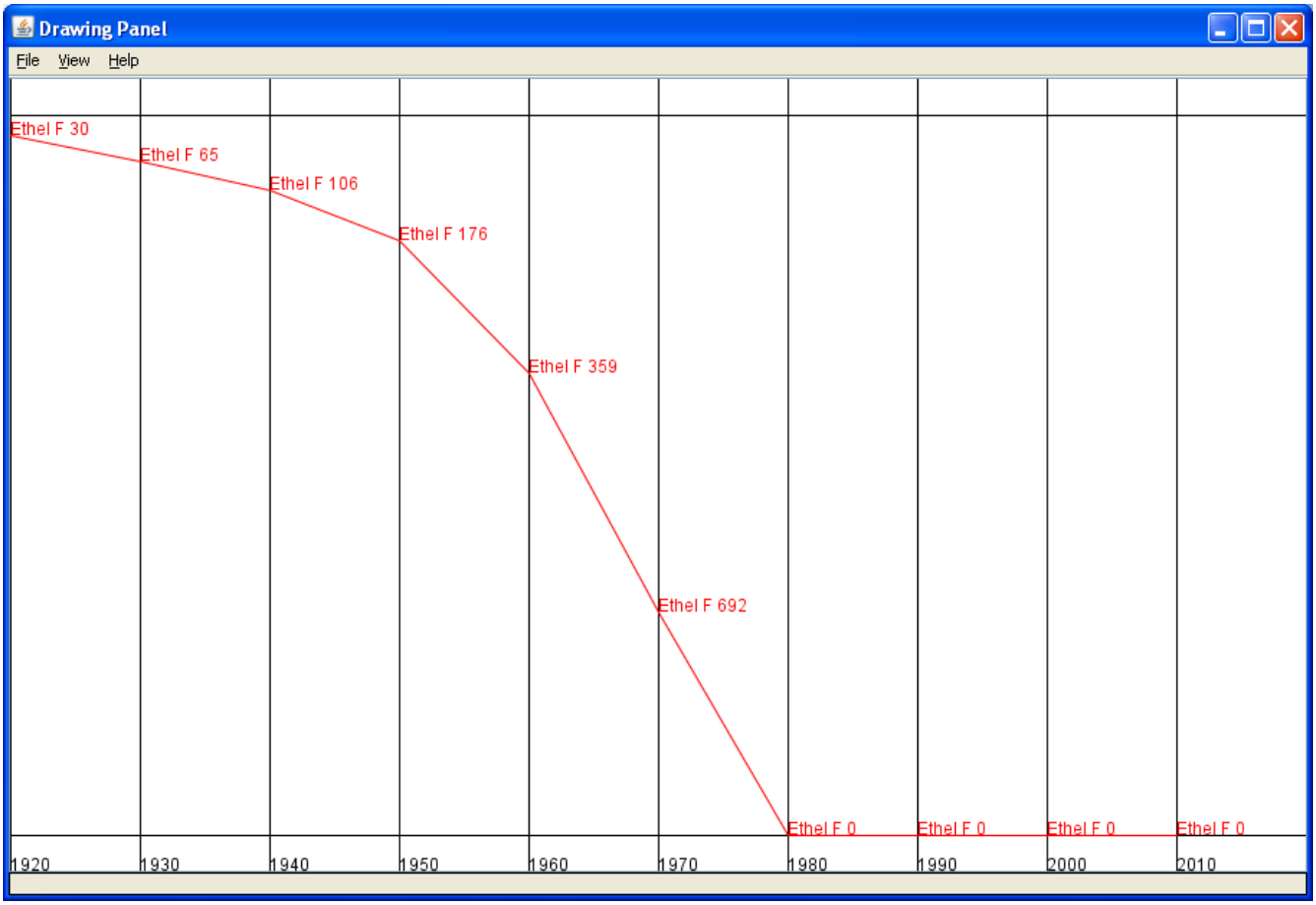
You will be using several different Scanner objects for this program. You will have one Scanner that you use to read information from the console. You will use a different Scanner to read from the input file. And because the input file is line-based, you should construct a different Scanner object for each line of the input file as explained in section 6.3 of the book. You should write your code in such a way that you stop consuming lines of input once you find one that has the name/sex combination you're searching for.

In terms of style points, we will be grading on your appropriate use of control structures like loops and if/else statements, your ability to avoid redundancy and your ability to break down the overall problem into methods that each solve part of the overall problem. No one method should be overly long. You are required to come up with at least four different methods other than main that each perform some nontrivial part of the problem. Avoid "chaining," which is when many methods call each other without ever returning to main. For this assignment you are limited to the language features in Chapters 1 through 6 of the textbook.

You will need to have DrawingPanel.java in the same directory as your program for it to compile properly. In addition, you will need to have the file names.txt in the same directory as your program for Java to find it. You may assume that the input files are legal (i.e., that they match the format described earlier). As with the Café Wall assignment, your output must match within 500 pixels of the sample output to be considered correct.

As noted earlier, your program has to have three constants for the number of decades, the start year and the horizontal width to use for each decade. It should be possible to reset these constants and to change the file it is reading from to have it work with a similar dataset. Your program doesn't have to figure out which kind of file it is working with. The idea is that you can reset the constants appropriately for a particular data file that you are using. You can make sure your program works properly by changing the number of decades to 10, the start year to 1920 and the horizontal width to 90 and changing the file name to names2.txt (which has just 10 decades

worth of data). You must also store names2.txt in the same directory as your program. Below is a sample of the output for "Ethel". You can introduce other class constants in addition to the three required constants.



We strongly recommend that you develop the program in stages. First write a program that just locates the appropriate line from the input file and prints it. Then have it use that input line to compute and print coordinates. Then add code to draw the grid. Then write code to draw the red plot on top of the grid.

Your program should be called Names.java. All of the files that you will need are included in a file ass6.zip on the class web page (you have to unzip it). These files should be put in the same folder as your program.

Log of execution for names2.txt and name found (user input bold/underlined)

```
This program allows you to search through the
data from the Social Security Administration
to see how popular a particular name has been
since 1920.

name? ethel
sex (M or F)? f
```

Log of execution for names.txt and name not found (user input bold/underlined)

```
This program allows you to search through the
data from the Social Security Administration
to see how popular a particular name has been
since 1880.

name? kumar
sex (M or F)? m
name/sex combination not found
```