

READ ALL INSTRUCTIONS BEFORE STARTING YOUR SOLUTION.

Topics: ch. 6: Selection, Iteration & Data File Processing loops, File input/output, Token/Line-based input, try/catch ifs, random numbers, and using object analysis, structured, top-down design, methods, expressions, variables, loops, class constants, formatting, Graphics, interactive programming.

Special thanks to Stanford lecturer Nick Parlante for the concept of this assignment!

also see: http://www.peggyorenstein.com/articles/2003_baby_names.html

Every 10 years, the Social Security Administration gives data about the 1000 most popular boy and girl names for children born in the US. This data on the web: <http://www.ssa.gov/OACT/babynames/>.

Specifications:

1. Create a program **named Sp8YourInitials_Names.java** (e.g. Ed Bob Hyde would use Sp8EBH_Names.java).
2. You must use copies (from the class web site) of names.txt and names2.txt to test your program.
3. You must meet all requirements in *Java_Program_Style_Requirements_GradingComponents* for full credit.
4. You must submit the .java file, printed, easy to read listing with line numbers and screen-captures of 3 runs.
5. At the top of your console output, display author, title and a brief introduction; but, no offensive remarks.
6. The main method must clearly show the overall program structure -- all major activities, not just header/body/trailer.
7. Where a program repeatedly gets user input and displays results, the main method must show that loop.
8. The main method must not read input, nor print nor draw output, itself; it must delegate that work.
9. You must have one method to do text output and a separate method to do any required graphical output.
10. The code that asks the user for input must not be in the same method as any code to read data from a file.
11. You must put any data files and DrawingPanel.java in the same directory as your program.
12. You must structure your solution using static methods that accept parameters and return values where needed.
13. You must have **at least 3 non-trivial methods other than main** in your program – see the next specification.
14. You should define other methods as needed for structure or to eliminate redundancy.
15. You must have one method to draw the "fixed" graphical content (yellow bars, vertical lines, etc.) and a different method to produce the content that comes from the file (red line, ranks, etc.).
16. Your program must prompt the user for a name, and then read through a data file searching for that name. If it is found display popularity statistics about that name for each decade since 1900. See the Example below.
17. You must display both text output on the console and a graphical line chart of this data on a DrawingPanel.

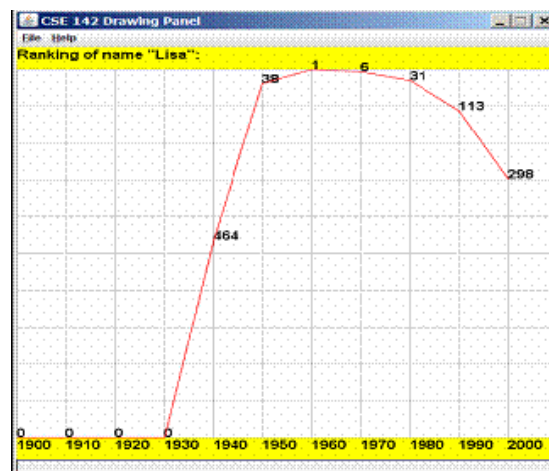
BabyNames program by Ed Hyde

This program graphs the popularity of a name in Social Security baby name statistics recorded since the year 1900.

Type a name (or press Enter to quit)? **Lisa**

Popularity ranking of name "Lisa"

```
-----
1900: 0
1910: 0
1920: 0
1930: 0
1940: 464
1950: 38
1960: 1
1970: 6
1980: 31
1990: 113
2000: 298
```



18. You must display the introduction and then the prompt for a name to display. Then provide output and re-prompt until the user presses ENTER without a name.

19. Your program will read its data from a file such as names.txt. Each line of this file has a name, followed by the popularity rank of that name in 1900, 1910, 1920, and so on. The default input file has 11 numbers per line, meaning that the last number represents the ranking in the year 2000. A rank of 1 was the most popular name that year, while a rank of 999 was not very popular. A rank of 0 means the name did not appear in the top 1000 that year at all. Below is a sample of the data:

```
...
Lionel 387 344 369 333 399 386 408 553 492 829 972
Lisa 0 0 0 0 464 38 1 6 31 113 298
Lise 0 0 0 0 0 997 0 0 0 0 0
Lisette 0 0 0 0 0 0 0 816 958 0 864
...
```

"Lionel" was #387 in 1900 and is losing popularity. "Lisa" made the list in 1940 and was #1 in 1960.

20. If the name is found, you must also construct a DrawingPanel to graph the data.

21. Your panel must exactly reproduce the window appearance of the examples for the same user input.

22. The panel's overall size is 550x560 pixels.

23. Its background is white.

24. It has yellow filled rectangles along its top and bottom, each being 30 pixels tall and spanning across the entire panel, leaving an area of 550x500 pixels in the middle.

25. If the name is not found in the file, you should simply output that it was not found, and not print or draw any data. No DrawingPanel should appear if the name is not found. The following is an example:

```
BabyNames program by Ed hyde
```

```
This program graphs the popularity of a name in Social Security baby name statistics recorded since
the year 1900.
```

```
Type a name (or press Enter to quit)? Zoidberg
```

```
"Zoidberg" not found.
```

Graphical Output:

26. Each decade must be shown in a width of 50 pixels.

27. Decades are separated by **vertical** (Color.LIGHT_GRAY) lines, each of which runs from y=30 to y=530.

28. The bottom yellow rectangle contains black text labels for each decade, left-aligned and with the text's bottom at y=546. For example, the text "1900" has coordinates (0, 546) and the text "1910" has the coordinates (50, 546). The heading "Ranking of name ..." is located at (0, 16).

A red line connects the data about the name's ranking over each decade.

The table below shows the mapping between rankings and y-values. The y-values start at 30, and there is a vertical scaling factor of 2 between pixels and rankings, so you should divide a ranking by 2 when calculating its onscreen y-coordinate. The red lines appear on top of any other elements that might occupy the same pixels.

Rank	y
1	30
2, 3	31
4, 5	32
...	...
998, 999	529
0	530

29. A rank of 0 means the name didn't appear in the top 1000, so it should be drawn at the bottom of the plot.

30. There are light gray **horizontal** lines every 50 pixels to indicate every 100 ranking points. For example, a line is drawn from (0, 80) to (550, 80) to mark the ranking of 100.

31. To the right of each line endpoint (at the same coordinate), **black** text must show the name's rank for that decade. For example, "38" appears to the right of the 1950 endpoint because "Lisa" had a rank of 38 in 1950.

32. The black text appears under the red line but on top of any lines or other elements that would otherwise occupy the same pixels.

33. Your program should work correctly regardless of the capitalization the user uses to type the name. For example, if the user asks you to search for "LISA" or "lisa", you should find it even though the input file has it as "Lisa". The name displayed on the console and DrawingPanel should match the capitalization in the file.
34. To draw the text labels on the DrawingPanel, you will need to use the drawString method of the Graphics object. Some of the text labels will be ints; convert them into Strings using the + operator with an empty string. For example, if you have an int named x with value 1900, the expression (" " + x) yields the string "1900". All onscreen text is drawn using the font named "SansSerif", in bold, at a size of 16.
35. You must use the Graphics object's setFont method to set its text size. See Section 3G of your textbook.
36. You must have 4 variables to represent the following values:
- the name of the input file (default of "names.txt")
 - the starting year of the input data (default of 1900)
 - the number of decades' worth of data in each line of the file (default of 11)
 - the width used for each decade on the drawing panel (default of 50)
37. After you get the other parts working, make it possible for the user to enter other values for these four variables and have your output adapt appropriately. For example, if the user asks to start at year 1800, the program will now assume the file data comes from the years 1800, 1810, and so on. The panel's overall width should adjust if the width or decades variables are changed; for example, if the width is 70 and the decades are 5, the panel's size should become 350x560. A second input file named names2.txt has 8 decades worth of data that you can use to test your file name and decades variables, as well as the width and starting year.
38. Your program should work if the user specifies fewer decades than data available in the specified file. How can you gracefully handle the situation where a user asks for more decades than the available data?