CSE 142, Autumn 2006
Programming Assignment #6: Baby Names (20 points)
Part A due: Sunday, November 12, 2006, 11:59 PM (+1 late day)
Part B due: Wednesday, November 15, 2006, 8:00 AM (20 points)

Special thanks to Stanford lecturer Nick Parlante for the concept of this assignment!
also see: http://www.farfilm.com/peggy/articles/wherehaveallthelisas.htm

Problem Description and Program Behavior:

This assignment will give you practice with file processing. It generates graphical output, so you will need to have `DrawingPanel.java` in the same directory as your program for it to compile properly. Turn in files named `BabyNamesA.java` (Part A) and `BabyNamesB.java` (Part B).

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 is provided on the web at http://www.ssa.gov/OACT/babynames/.

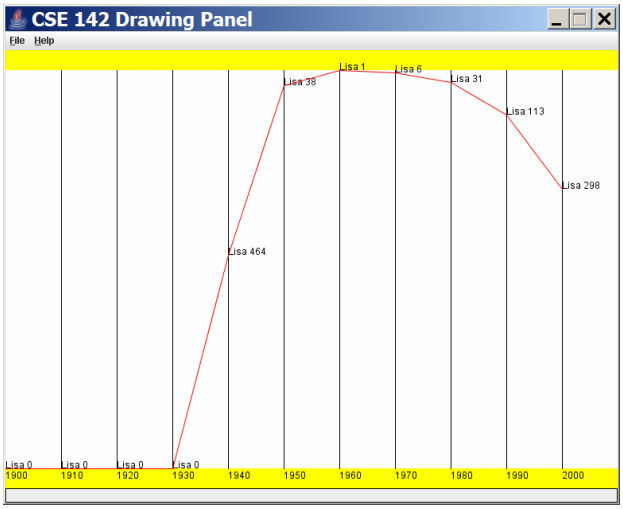
Your task in this program is to prompt the user for a name, and then to display popularity statistics about that name for each decade since 1900. You will display both a text output of this data and a graphical line chart of this data on a `DrawingPanel`.

```
This program graphs the popularity of a name
in Social Security Administration statistics
recorded starting at 1900.

Type a name (or 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

11 decades' worth of data were found.
```



Your program is to give an introduction and then prompt the user for a name to display. Then it will read through a data file searching for that name. If the name is found in the file, you should print the name and the statistics about that name's popularity in each decade.

Your program will read its data from a file named `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. Here 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 slowly decreasing. "Lisa" made the list in 1940 and peaked in 1960 at #1.

If the name is found, you must also construct a `DrawingPanel` to graph the data. Your panel must exactly reproduce the window appearance of the examples for the same user input.

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:

```
This program graphs the popularity of a name
in Social Security Administration statistics
recorded starting at 1900.

Type a name (or Enter to quit)? Zoidberg
"Zoidberg" was not found.
```

Text Output (Part A) (optional):

To earn an extra late day, turn in a version of this program that produces only the text output (no `DrawingPanel`). Your code will receive the late day if it has correct logic and approximately matches the expected output; style will be ignored.

Graphical Output (Part B):

The panel's overall height is exactly 550 pixels. Its background is white. It has yellow filled rectangles along its top and bottom, each being 25 pixels tall and spanning across the entire panel, leaving an area 500 pixels tall in the middle.

Each decade is represented by a width of 70 pixels. The decades are separated by vertical black lines from y=25 to y=525. The bottom yellow rectangle contains black text labels for each decade, left-aligned and with the text's bottom at y=538. For example, the text "1900" has coordinates (0, 538) and the text "1910" has the coordinates (70, 538).

A red line connects the data about the name's ranking over each decade. The red lines appear on top of any other onscreen elements that might occupy the same pixels. 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. A rank of 1 should be drawn at y=25. A rank of 2 or 3 should be drawn at a y-coordinate of 26. A rank of 4 or 5 should be drawn at a y-coordinate of 27. And so on, up to a rank of 998 or 999, which should be drawn at a y-coordinate of 524. A rank of 0 means the name didn't appear in the top 1000 at all, and should be drawn at the very bottom of the plot range, at a y-coordinate of 525.

To the right of each decade's line endpoint (at the same coordinate), black text shows the name followed by a space followed by its rank for that decade. For example, the text "Lisa 38" appears to the right of the line for 1950, because the name "Lisa" had a rank of 38 in 1950.

The panel's overall width should be 70 pixels times the number of decades shown. There are 11 decades in the default input file `names.txt`, so the panel to show this data would have a width of 770, but if the number of decades changes in the file, the panel's width should adjust itself accordingly. For example, there is a second input file named `names2.txt` that has 8 decades worth of data; when displaying this file, the panel should have a width of 560. You may assume that the input file has at least 2 decades' worth of information on each line and that each line has the same number of decades.

Implementation Guidelines:

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 that is displayed on the console and `DrawingPanel` should appear with standard capitalization as it appears in the file.

To draw the text labels on the `DrawingPanel`, you will need to use the `drawString` method of the `Graphics` object, as described in Supplement 3G of the textbook. Some of the text labels you'll want to write will be `int`s, but can convert them into `String`s 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"`.

Stylistic Guidelines for Part B:

Use methods to provide structure and avoid redundancy. No one method should be overly long. You should decide for yourself exactly what methods to use, but for full credit, your methods should obey the following constraints: The `main` method in particular should not perform drawing operations on a `DrawingPanel`, nor should it directly read lines of input from a file; perform these tasks in other methods. The code that asks the user for a name must not be in the same method as any code to read lines of input from a file. No single method may produce both graphical and text output. That is, any method that performs operations on a `DrawingPanel` or `Graphics` object should not contain any `println` statements.

Use a class constant in your program for the starting year (1900). It should be possible to change this value and have your output adapt appropriately. For example, if you change the constant to 1800, the program will now assume that the first number on every line of the file represents data from the year 1800, the second number from 1810, and soon. Note that after changing the starting year constant or changing the input file, the ending year will not necessarily be 2000.

Follow past stylistic guidelines about indentation, whitespace, identifier names, and localizing variables, and commenting at the beginning of your program, at the start of each method, and on complex sections of code.