CSE 142, Autumn 2007
Programming Assignment #6: Baby Names (20 points)
Due: Wednesday, November 14, 2007, 8:00 PM

Special thanks to Stanford lecturer Nick Parlante for the concept of this assignment!
also see: http://www.peggyorenstein.com/articles/2003_baby_names.html

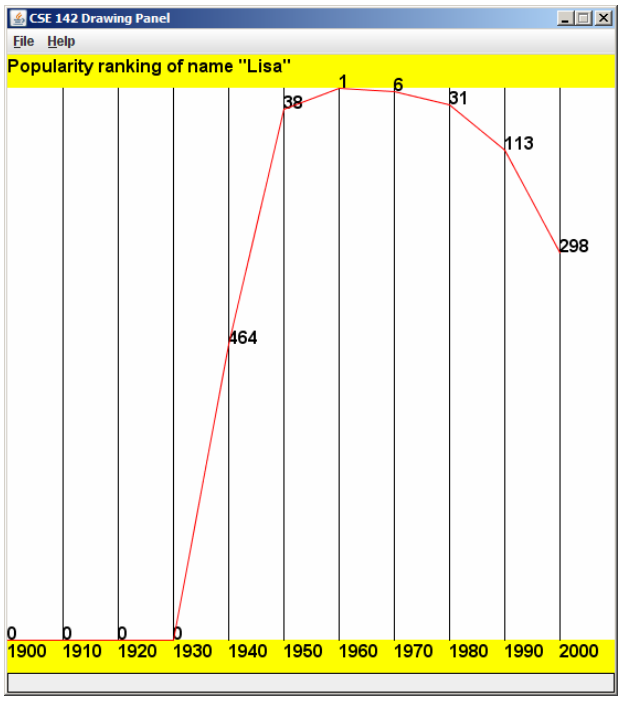
Program Description:

This assignment focuses on file processing. You will need `DrawingPanel.java`. Turn in a file named `BabyNames.java`.

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 11 decades' worth of statistics
recorded since the year 1900.

Type a name: 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
```



Your program gives an introduction and then prompts the user for a name. Then it reads a file searching for that name, ignoring case. If the name is found in the file, print the name and statistics about that name's popularity in each decade.

Your program will read its data from a file named `names.txt`. Assume that this file exists in the same folder as your program. Each line of the file has a name, followed by the 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" first 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.

The panel's overall size is 550x560 pixels. Its background is white. 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. At (0, 16), the following message appears in the top yellow bar for Lisa:

`Popularity ranking of name "Lisa"`

If the name is not found in the file, you should output that it was not found, and not draw any data. No `DrawingPanel` should appear in this case. The following is an example:

```
This program graphs the popularity of a name
in 11 decades' worth of statistics
recorded since the year 1900.

Type a name: zOIDberG
"zOIDberG" not found.
```

Graphical Output:

Each decade is represented by a width of 50 pixels. The decades are separated by vertical black lines, each of which runs from y=30 to y=530. 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" is at (0, 546) and the text "1910" is at (50, 546).

A red line connects the name ranking data over each decade. The table at right 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 y-coordinate. A rank of 0 means the name didn't appear in the top 1000, and should be drawn at the bottom of the plot range. 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

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

Implementation Guidelines:

We suggest you begin with the text output and then handle the graphical output. To handle the "name not found" case, you may want to review book section 5.2 about `boolean` flags. The red ranking line is tough and should be done after the text and simpler graphics work. The 0-ranking case is particularly tricky to draw, so you may want to do this last.

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 have capitalization matching the way it appears in the file.

Draw the text labels on the `DrawingPanel` using the `drawString` method of the `Graphics` object. Some text you'll want to write will be `int`s, but you can convert them into `String`s using the `+` operator with an empty string. For example, for an `int` named `n` with value `100`, the expression `"" + n` yields the string `"100"`. To draw this at (50, 120), you'd write:

```
g.drawString("" + n, 50, 120);
```

All text is drawn using bold `"SansSerif"` font at size 16. Set this by using the `Graphics` object's `setFont` method:

```
g.setFont(new Font("SansSerif", Font.BOLD, 16));
```

Stylistic Guidelines:

For this program you should have **four class constants** 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`)

If the constants values are changed, your output should adapt appropriately. For example, if you change the starting year constant to 1800, the program should assume the data comes from 1800, 1810, and so on. The panel's width should adjust if your width or decades constants change. For example, if you change width to 70 and decades to 5, the panel's size becomes 350x560. On the course website is a second file `names2.txt` with 8 decades of data to test your constants.

We will be especially picky about redundancy. Remember to factor `if/else` code as described in book section 4.3. Use methods for structure and to avoid redundancy. For full credit, your methods should obey the following constraints:

- The `main` method should not draw on a `DrawingPanel`, nor should it read any lines of input from a file.
- 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.
- Split the task of displaying the data into at least two methods. For example, you could have one method for text and one for graphics; or you could have one method to draw "fixed" graphics (yellow bars, vertical lines) and another for graphics that come from the file (red line, ranks).

For this assignment you are limited to the language features in Chapters 1 through 6 of the textbook. In particular, **you are not allowed to use arrays on this assignment.** 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. For reference, our solution occupies 100 lines and has 4 methods other than `main`.