1. (10 points) Chapter 6: Self-check problems #7, #8, #12, #13, #20

For your benefit, I suggest you complete all of the self-check problems on your own. However, I will just collect the answers to the above five problems. Please type the answers into a document.

a. NO NEED to copy original questions, just put down the question number and the answer.
b. Name the file as "***LastnameFirstname6.doc***" (or ".docx" or ".pdf", where "Lastname" is your last name, and "Firstname" is your first name), submit it online.

2. (40 points) Chapter 6: Programming Project #4 "Baby Names"

a. Download the attached "`BabyNames.java`", and "names.txt" file.
b. Modify the "BabyNames.java" so it will do the following:

i. The "names.txt" file contains data about popular baby names over the last 80 years in the United States (http://www.ssa.gov/OACT/babynames). Every 10 years, the data gives the 1000 most popular boy names and girl names for children born in the US. The data can be summarized in the following format:

```
...
Sam 99 131 168 236 278 380 467 408 466
Samantha 0 0 0 0 272 107 26 5 7
Samara 0 0 0 0 0 0 0 0 886
Samir 0 0 0 0 0 0 920 0 798
...
```

Each line has a name followed by the rank of that name in 1920, 1930, …, 2000 (9 numbers). A rank of 1 was the most popular name for that decade, while a rank of 907 was not that popular. A 0 means that the name was out of the 1000 popular names for that decade.

ii. Your program will give an introduction and then prompt the user for a name. Then it will read through the data file searching for that name. The search should be case-insensitive, meaning that you should find the name even if the file and the user use capital letters in different places. As an extreme example, SaMueL would match sAMUeI.

iii. If your program finds the name, it should print out the statistics for that name onto screen. You are to reproduce this format **exactly**:

```
** Popularity of a baby name since year 1920 **
name? ada
1920: 154
1930: 196
1940: 244
1950: 331
```

```
1960: 445
1970: 627
1980: 962
1990: 0
2000: 0
```

Then, you need to save the statistics into a result file for that baby name. For the example above, the result file should be named as "**Ada.txt**", which should contain contents in the following format **exactly**:

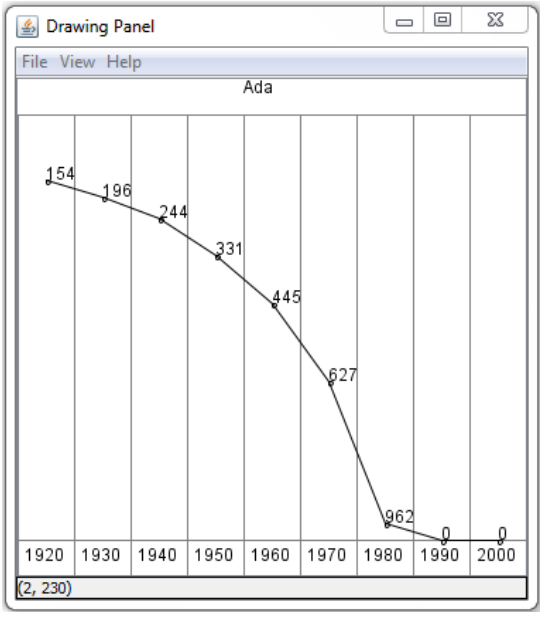
```
Ada,
1920: 154,
1930: 196,
1940: 244,
1950: 331,
1960: 445,
1970: 627,
1980: 962,
1990: 0,
2000: 0
```

iv. If the name is not in the data file, your program should generate a short message onto screen indicating the name was not found. You are to reproduce this format **exactly**:

```
** Popularity of a baby name since year 1920 **
name? kumar
name not found.
```

In this case, you should **not** generate the result file.

v. **AFTER you successfully find the name**, print out to screen and save to the result file, try drawing a graph for the ranks of that name over time. One example is shown below. Of course, you can be creative here.



vi. Keep in mind, **part v (the graph part) is worth only 5 points.** If you cannot get part v work, that's ok – however, before submission **you need to make sure your program still compiles and works for the previous parts**. (Remember, if your code does not compile, you will lose a significant portion of your points!)

vii. Keep in mind, my computer directory is different from yours, so please **DO NOT use absolute path for the files** (e.g. c:\temp\1.txt) - see **Tip** below. Otherwise, when I grade on my computer, it won't work (then you will lose points!)

viii. For this assignment, you are limited to the language features in Chapters 1 through 6 of the textbook. In particular, ***do not use arrays on this assignment***.

ix. Remember, your program will be graded both on "external correctness" and "internal design and style" (whether your source code follows the **style guide**).

c. Submit the final "`BabyNames.java`" file **(DO NOT change the file name)** online.

Tip:
You need to have the file "names.txt" in the same directory as your java program is running from (i.e. the current directory). Otherwise, you will get "FileNotFoundException". For people using IDE (like NetBeans or Eclipse), it's probably not obvious where the current directory is. You can use the following code to find out:

```
String currentDir = System.getProperty("user.dir");
System.out.println(currentDir);
```

Run it, look at the output and figure out where the current directory is. Copy "names.txt" there, and move on!