CSC 220 – Project 4: Name Surfer

Due Date:
Friday, December 10, 2021 before 11:59 PM

Objectives:
Analyze data from a large database of names.
Assess student's ability to function effectively on a team to accomplish a common goal.

Development Process:
Students will work in self-selected pairs.

Students will divide the development of classes and methods in a fair manner and work together to integrate their code.

Students should employ agile software development techniques. Plan daily scrums and bi-weekly duration sprints.

Project Details:
Design and implement a program that analyses baby name popularities in data provided by the Social Security Administration.

For each year of birth YYYY from 1890, Social Security created a comma-delimited file called yobYYYY.txt. Each record in the individual annual files has the format "name,sex,number," where name is 2 to 15 characters, sex is M (male) or F (female) and "number" is the number of occurrences of the name. To safeguard privacy, names are restricted to those with at least 5 occurrences.

Each file is sorted first on sex and then on number of occurrences in descending order. When there is a tie on the number of occurrences, names are listed in alphabetical order. This sorting makes it easy to determine a name's popularity rank, i.e. its popularity in that year: the first record for each sex is rank 1, the second record for each sex is rank 2, and so forth.

```
Emma,F,18688
Olivia,F,17921
Ava,F,14924
Isabella,F,14464
Sophia,F,13928
Charlotte,F,12940
Mia,F,12642
Amelia,F,12301
Harper,F,10582
Evelyn,F,10376
...
```

Important: be sure to understand the difference between popularity rank and number of occurrences. **The third value is the number of occurrences of the name. It is NOT the popularity rank.**

The set of files from yob1880.txt through yob2020.txt is available on Canvas as names.zip. In your Project4 folder, unzip it into a subfolder called “names”.

Classes Required:

- NameRecord – encapsulates the data for one name
    - Variables
        - String `name`
        - int array `rank` to store the popularity rank value for each year; **do NOT store the number of occurrences!**
        - constant int variables START=1880 and END=2020 define the start and end years in the data. **You MUST use START and END throughout your code!**
    - Methods:
        - Constructor – takes a String name. Initializes the NameRecord object.
        - String getName() – returns the name
        - void setRank (int year, int rank) – sets the popularity rank of the name in the given year. Use the convention that year=0 is 1880, year=1 is 1881, and so on.
        - int getRank(int year) – returns the popularity rank of the name in the given year. Use the convention that year=0 is 1880, year=1 is 1881, and so on.
        - int bestYear() – returns the year where the name was most popular, using the earliest year in the event of a tie. Returns the actual year, for example 1920, so the caller does not need to adjust for START. It is safe to assume that every name has at least one year with a non-zero popularity rank.

- NameSurfer – the driver. The main method should read all the data from the files and store it in two ArrayLists – one for male names and one for female names – of NameRecord objects. It should then offer the following menu to the user:

```
1 – Find best year for a female name
2 – Find best rank for a female name
3 – Find best year for a male name
4 – Find best rank for a male name
5 – Quit
Enter your selection.
```

  If 1, 2, 3, or 4 is entered, the program should prompt the user for a name, search for that name in the ArrayList **(search should ignore case)**, print the desired information, and display the menu again. If the name is not found in the ArrayList print an error message and display the menu again.

Note:
Remember to keep your methods and objects encapsulated.

Formatting Requirements

- Follow indentation rules as discussed in class.
- Use descriptive variable names.
- Comment your code.
  - Every method should have a comment containing a short description on top.
  - Every instance variable should be followed by a comment that describes it.
  - Also comment portions of code that solve a specific subproblem.
- Don't forget to include your names, the course number, title of the assignment, today's date, and an overview comment near the top.

What to turn in:

- **Group submission**: jar your two Java source files and a screenshot and submit them to Canvas by the deadline. Do NOT include the "names" folder in your submission. <u>Students should submit only one project per team.</u>

```
jar cvf Project4.jar NameRecord.java NameSurfer.java screen.jpg
```

- **Individual submission**: each student shall describe their individual contribution to the project as well as reflect on how the two students facilitated each other's contributions. Use the following outline. This writeup shall be a minimum of one page.
  - What worked well?
  - What challenges did you encounter?
  - What do you wish you had done differently?
  - Was the software development process effective?
  - Was the effort evenly divided?
  - What was your overall experience working on a small team?

Grading:

- NameRecord class has the correct instance variables and constants – 5p
- NameRecord constructor correct – 5p
- getName() correct – 5p
- setRank() correct – 5p
- getRank() correct – 5p
- bestYear() correct – 5p
- driver correctly reads data from yobYYYY.txt files and stores it in an ArrayList – 15p
- driver prints user menu and reads user's option – 5p
- search for name correct – 10p
- option 1 correctly handled – 5p
- option 2 correctly handled – 5p
- option 3 correctly handled – 5p
- option 4 correctly handled – 5p
- option 5 correctly handled – 1p
- Proper comments and indentation – 4p
- Individual writeup– 15p