CSCI 2073 – Programming Assignment 4 – Fall 2019

The Social Security Administration publishes lists of the most popular baby names on their web site (http://www.ssa.gov/OACT/babynames/). When querying the most popular names for a given decade, the browser displays the result on the screen. The results can also be copied and pasted onto text files for processing (see sample files babynames60s.txt, babynames70s.txt, etc.). Each line in the text files contains five entries:

• The rank (a positive integer)
• The name and frequency of the male name of that rank
• The name and frequency of the female name of that rank

For example, the line below from the babynames60s.txt file shows that the 7th most common boy's name was William, with 421,592 births during that period. The 7th most common girl's name was Linda.

```
7  William  421,592  Linda 225,412
```

Your assignment is to write a Java class **`BabyNames`** to store and process name information. You are free to store team and game information in any suitable data structure(s) of your choice, but keep in mind that the data files may have different lengths.

The **BabyNames** class is required to contain the following public methods:

`BabyNames(String filename)`: a constructor to read and store data

`String processQueries(String queryFile)`: method to process a file containing a number of queries. The file contains one request per line. The possible requests are:

FIND name

RANK number

TOP minpct

UNISEX

Each request requires that the program produce specific output on a line by itself, as follows:

- FIND name: displays the name, gender, and corresponding rank and number of births. If the name appears on both the male and female lists, two lines of output will be generated (if the name is not found, the program should display NOT FOUND instead of gender, rank, and number of births).
  ```
  NAME: Name (FEMALE)              RANK: ##     BIRTHS: ##,###
  ```
- RANK number: displays two output lines with the same information as shown in the FIND name transaction for the male and female names with the corresponding rank.
  ```
  NAME: Name (MALE)                RANK: ##     BIRTHS: ##,###
  NAME: Name (FEMALE)              RANK: ##     BIRTHS: ##,###
  ```

- TOP minpct: displays two lists of names (one for female names, one for male names) with enough births to exceed the specified percentage of total births for that gender, using the format below (if either list is empty, just display "NONE" instead of the empty list).
  ```
  TOP MALE NAMES:   nameX (x.xx%), nameY (y.yy%), …, nameZ (z.zz%)
  TOP FEMALE NAMES: nameX (x.xx%), nameY (y.yy%), …, nameZ (z.zz%)
  ```
- UNISEX: displays a list of names that appear in both the male and female lists, using the format below. If no name meets the criteria, just display "NONE" instead of an empty list.
  ```
  UNISEX NAMES: nameX, nameY, …, nameZ
  ```

The *BabyNamesTest.java* program, as well as the files *100babynames.txt* and *queries.txt* are provided for basic testing. The sample output below should result from executing *BabyNamesTest* with your solution and the data files provided. Once you are satisfied with your results, submit *BabyNames.java* and any other .java files developed as part of your solution. Do not submit any of the files provided by the instructor for testing.

Sample input file contents and corresponding output:

`FIND Paul`	`NAME: Paul (MALE)   RANK: 17      BIRTHS: 1,333,140`
`FIND Eleanor`	`NAME: Eleanor NOT FOUND`
`RANK 10`	`NAME: Thomas (MALE)       RANK: 10       BIRTHS: 2,188,254`
`TOP 4.0`	`NAME: Sarah (FEMALE)      RANK: 10       BIRTHS: 997,279`
`UNISEX`	`TOP MALE NAMES:   James (5.06%), John (4.91%), Robert (4.84%), Michael (4.46%)`
	`TOP FEMALE NAMES: Mary (5.53%)`
	`UNISEX NAMES: NONE`

For this assignment, your class comments must include a detailed description of the data structures you chose as well as a rationale for your choices.