Programming Challenge- Java

SUMMARY

Your objective will be to write the fastest, most efficient implementation of weight-checking and string-sorting algorithms in Java. Your code will be assessed for creativity, elegance, speed, scalability and pass rate.

Comments and Javadocs will be appreciated! If your code fails to compile, you will be disqualified.

NOTE: You may use not use any external libraries or JARs. Libraries are limited to the standard Java libs. Java SDK **8u121** should be used for this project. Follow the data structures presented in the objectives. In the case of sorting, it is mandatory to receive an array as a parameter and use that structure to sort.

RESTRICTIONS: The JVM settings will be as follows: **-Xms256m –Xmx1g**

All methods should all reside in a **StringManipulator** class.

OBJECTIVE 1: STRING NORMALIZATION

Implement a String function **cleanString(String str)** which removes all non-alphabetic characters from a string and converts all characters to upper case.

OBJECTIVE 2: STRING UNIQUE CHARACTERS FUNCTION

Implement the Boolean function **hasUniqueChars(String word)** to check if a word only contains unique characters. Your function should not consider non-alphabetic characters, and it should not be case-sensitive. This function will return true only if the word doesn't have any character that is repeated, e.g. **hasUniqueChars** (“a2p’}}pl66e”) should return **false**, and **hasUniqueChars**(“co%m$Puter”) should return **true**.

OBJECTIVE 3: STRING WEIGHT FUNCTION

Implement the Real function **getWeight(String word)** to check the weight of a string. The weight is defined as a numeric value that is the average of each char ascii value contained in the string. Your function should not consider non-alphabetic characters, and it should not be case-sensitive, e.g. **getWeight** (“Ab5-:C”) should return **66**.

OBJECTIVE 4: SORTING

Implement a function **sortStrings(String[ ] words)** that will sort a large array of strings based on the value obtained by the weight function in ascending order. Please ensure to keep scalability in mind when designing this method. The output of this function can either be the sorted array, or sending the output directly to the file mentioned in the objective 5. The candidate should pick whichever option will be more optimal with the working solution.

OBJECTIVE 5: TESTING

Implement a **main()** function which will read in a .csv file located in src/atdd/challenge_tests.csv and execute the **hasUniqueChars** () function against each of the words in the file. The main function should also sort all the words in challenge_tests.csv in ascending ordered by string weight and output the cleaned, sorted list to src/atdd/ challenge_sorted.csv. Words should be completely alphabetic and in all upper-case letters.

NOTE: The first entry in challenge_tests.csv will be the column name 'Words'. The challenge_sorted.csv should contain 3 columns, with the first column header as 'Words', the second column header as 'Unique', and the third column header as 'Weight'. The first column should contain the cleaned and sorted list of words from challenge_test.csv, the second column should contain whether the word contains only unique characters in it or not, and the third one should contain the average ascii value of each character in the string. Please see sample files in the attachment that was emailed with this document in samples.zip.

OBJECTIVE 6: VERSION CONTROL & CODE SHARING

Please setup a BitBucket account and commit your code to a personal repository. Share your repository with the following users:

- Sebastian Meneses (sebastian.menesespilco@bbva.com)
- Rohit Mukherjee (rohit.mukherjee.contractor@bbva.com)
- Terry Pitman (terry.pitman@bbva.com)
- Narayana Chiluka (narayanareddy.chiluka@bbva.com)

Any commits occurring after June 31st 11:59 AM CST will not be accepted.

Good luck!