REVIEW

OBJECTIVES

This assignment is meant to review CS 120 material. In order to complete this assignment you must know how to deal with Strings and one-dimensional arrays.

OVERVIEW

When looking for data on the Internet, you are often interested in only a subset of the data in a particular set. If you are in a fantasy football league, for example, you might have a list of all players in the NFL but only want to see those that are classified as 'quarterbacks'. Or perhaps you are looking at a database of all known diseases but are only interested in those diseases that cause extreme fatigue. In this assignment, you are given a set of data (an array of words) that you must filter to produce a subset of those words that adhere to some criteria.

DETAILS

You must write a class named **WordFilter**. The class must have the methods listed below. Each of these methods is independent (you can complete any one of them without completing the others) and they must not access any instance variables. *None of the methods must alter the inputs in any way. Unless otherwise noted, the ordering of the output does not matter*, although duplicate entries and nulls are not allowed. Also, *each function is case insensitive*.

Most of the method descriptions also give an example of how it behaves on a specific input. Unless otherwise noted, assume that the input for each of these examples is an array of Strings containing: "aardvark", "bat", "brewers", "cadmium", "wolf", "dastardly", "enigmatic", "frenetic", "sycophant", "rattle", "zinc", "alloy", "tunnel", "nitrate", "sample", "yellow", "mauve", "abbey", "thinker", and "junk".

1. **public String[] wordsStartingWith(String[] words, String prefix):** Given an array of strings, find and return all strings in that array that start with the prefix string. If none of the strings start with the prefix, return an array that has a length of zero. All strings start with the empty string. For example, if this function is called with a prefix of "b", the result should contain: "bat" and "brewers". If this function is called with a prefix of "ba", the result should contain "bat".

2. **public String[] wordsContainingPhrase(String[] words, String substring):** Given an array of strings, find and return all strings in that array that contain the substring. If none of the strings contain the substring, return an array that has a length of zero. All strings contains the empty string as a substring. For example, if this function is called with a substring of "at", the result should contain "bat", "enigmatic", "rattle", and "nitrate".

3. **public String[] wordsContainingAll(String[] words, char[] letters):** Given an array of strings, find and return all strings in that list that contain all of the specified letters. The order of the letters in the word does not matter. If none of the strings start with the prefix, return an array that has a length of zero. For example, if this function is called with a letters of "at", the result should contain "bat", "dastardly", "enigmatic", "sycophant", "rattle", and "nitrate".

4. **public String[] wordsInBoth(String[] words1, String[] words2):** Given two arrays of Strings, return an array that contains every string that occurs in both of the input arrays. The output must not contain any duplicate elements. For example, if list1 contains {"cat", "bat", "boat", "car", "bunny"} and list2 contains {"cat", "plane", "bunny", "cat"} the output list must contain only {"cat", "bunny"}. The order of the elements in the output is not relevant.

5. **public int[] indicesOfWordsStartingWith(String[] words, String prefix):** Given an array of Strings, find and return *the indices of* all strings in that list that start with the prefix string. You must return these words as an array of ints such that if words[X] starts with the prefix string, then the resulting array must contain X. If none of the Strings start with the prefix, return an array that has a length of zero. For example, if this function is called with a prefix of "b", the result should contain the *indices of*: "bat" and "brewers". If this function is called with a prefix of "ba", the result should contain the *index of* "bat".

6. **public int[] indicesOfWordsContainingPhrase(String[] words, String substring):** Given an array of Strings, find and return *the indices of* all strings in the array that contain the provided substring. You must return these words as an array of ints such that if words[X] contains the substring, then the resulting list must contain X. If none of the Strings contain the substring, return an array that has a length of zero. For example, if this function is called with a substring of "at", the result should contain *the indices of* "bat", "enigmatic", "rattle", and "nitrate".

7. **public int[] indicesOfWordsContainingAll(String[] words, String letters):** Given an array of Strings, find and return *the indices of* all strings in that array that contain all of the specified letters. The order of the letters in the word does not matter. If none of the Strings contain all of the letters, return an array that has a length of zero. For example, if this function is called with a prefix of "b", the result should contain *the indices of*: "bat" and "brewers". If this function is called with a prefix of "ba", the result should contain *the index of* "bat".

8. **public int numberOfWordsInBoth(String[] arr1, String[] arr2):** Given two arrays of Strings, return the number of strings that occur in both of the arrays. For example, if the first array contains {"cat", "bat", "boat", "car", "bunny"} and the second array contains {"cat", "plane", "bunny", "cat"} the output list must be 2.

9. **public char[] lettersOf(String word):** Given a string, return the letters of the string as an array that is sorted in ascending order. The output must be all lower case. For example, if the input word is "Mississippi" the output must be { 'i', 'i', 'i', 'i', 'm', 'p', 'p', 's', 's', 's', 's' }.

10. **public boolean every(boolean[] flags):** Given an array of booleans, return true if every boolean is true; otherwise return false. For example, if the input is {true, true, true, false} the output is false.

11. **public boolean areAnagrams(String word1, String word 2):** returns true if the two inputs are anagrams. An anagram is defined as any word that is formed by rearranging the letters of another word. For example, "art" and "rat" are anagrams. If the two words are identical, they are not anagrams.

PLAGIARISM

You must work **individually**. You may not copy code from any other source and claim it as your own. You may not share your code with any other student for use in completing this assignment.