Vocab FRQ

SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Assume that the classes listed in the Java Quick Reference have been imported where appropriate.

Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.

In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

Vocab FRQ

Some applications use a controlled vocabulary to describe, or tag, things. A controlled vocabulary is a limited set of keywords from which appropriate tags can be chosen.

The *Vocab* class, shown below, contains methods used to analyze words in terms of their presence in a controlled vocabulary. You will write two methods of the *Vocab* class.

```
public class Vocab
{
   /** The controlled vocabulary for a Vocab object. */
   private String[] theVocab = { /* contents not shown */ };

   /** Searches for a string in theVocab. Returns true if its String
    *  parameter str is an exact match to an element in theVocab and
    *  returns false otherwise.
    */
   public boolean findWord(String str)
   {
        /* implementation not shown */
   }

   /** Counts how many strings in wordArray are not found in
    * theVocab, as described in part (a).
    */
   public int countNotInVocab(String[] wordArray)
   {
        /* to be implemented in part (a) */
   }

   /** Returns an array containing strings from wordArray not found
    * in theVocab, as described in part (b).
   */
   public String[] notInVocab(String[] wordArray)
   {
        /* to be implemented in part (b) */
   }
}
```

Vocab FRQ

The `countNotInVocab` method returns an `int` that contains the number of words in its parameter `wordArray` that are not found in the instance variable `theVocab`.

A helper method, `findWord`, has been provided. The `findWord` method searches for an individual `String` in `theVocab`, returning `true` if an exact match between its `String` parameter and an element of `theVocab` is found, and returning `false` otherwise.

(a) Write the `countNotInVocab` method. Assume that there are no duplicates in `wordArray`. You must use `findWord` appropriately to receive full credit.

```
/** Counts how many strings in wordArray are not found in theVocab,
 * as described in part (a).
 */
public int countNotInVocab(String[] wordArray)
```

The `notInVocab` method returns an array of `String` objects that contains only elements of its parameter `wordArray` that are not found in `theVocab`. The array that is returned by `notInVocab` should have exactly one element for each word in `wordArray` that is not found in `theVocab`. Assume that there are no duplicates in `wordArray`.

The following example illustrates the behavior of the `notInVocab` method.

`theVocab`:

`"time"`	`"food"`	`"dogs"`	`"cats"`	`"health"`	`"plants"`	`"plants"`

`wordArray`:

`"dogs"`	`"toys"`	`"sun"`	`"plants"`	`"time"`

Array returned by `notInVocab`:

`"toys"`	`"sun"`

(b) Write the `notInVocab` method. Assume that there are no duplicates in `wordArray`. You must call `findWord` and `countNotInVocab` appropriately in order to receive full credit.

```
/** Returns an array containing strings from wordArray not found in
 * theVocab, as described in part (b).
 */
public String[] notInVocab(String[] wordArray)
```