CSC111 Lab 4 – Counting Words Fall 2017

In this lab, we will scan an input text and count the number of times each word appears.

Input: A text file containing English sentences.

Output: A list of words and how often each word occurs. All words must be listed in lower case, regardless of the use of upper/lower case in the input text. For example:

```
apple     3
carrot    1
grape     7
pear      2
```

Program Organization:

Class wordlist: Write a class named `wordlist`.

- Class `wordlist` has four non-static data items:
  - the allocated size of the arrays, e.g., `int allocated_size ;`
  - an integer representing the number of words stored in the String array, e.g., `int count ;`
  - an array of `String`, e.g., `String [] words ;`
  - an corresponding array of `int` representing the number of times each word occurs, e.g., `int [] word_counts ;`
- Class `wordlist` has four non-static methods.
  - Class wordlist has a **constructor** which accepts an integer `N`. The constructor allocates the arrays `words` and `word_counts` for the current object. The constructor also initializes all of entries in the `word_counts` array to zero.
  - Class wordlist has a non-static method named `print` that prints the list of words and their corresponding counts.
  - Class wordlist has a non-static method named `search` that searches the array `words` for a given word. This method returns an integer indicating the position in the array `words` where the search target is found. If the search target is not found, then return -1. E.g., `int search( String w )`
  - Class wordlist has a non-static method named `update` that updates the word list and the word counts. This method uses method `search` to determine if a word is already stored in the array; if so, then the corresponding word count is incremented. Otherwise, it is a new word, and it is added to the end of the word array. E.g., `void update( String s )`

Class main: Write a class named `main`. Class `main` contains only the main (static) method.

- You will need to import three classes from the library:

```
import java.util.Scanner ;
import java.util.NoSuchElementException ;
import java.util.regex.Pattern ;
```

- Declare and initialize a **wordlist** object Assume the input text has no more than 512 distinct words.
- Use a **Scanner** object to read the input to the end. You will need a try-catch structure to detect the end of input.
  - Study the example discussed in class to use a Java **Pattern** as a delimiter with your **Scanner** object. This approach prevents the scanner from including punctuation in the words. The simplest pattern which works well is "one or more occurrences of a non-word character". Fortunately, the **Pattern** class supports a pre-defined notation for "a non-word character", i.e., `\W`.

    A pattern may be specified as "one or more" by appending a `+` to the end of the pattern.
- Use a **String** class method `toLowerCase` to convert each word to lower case.

Running Your Program

Download the file `sample_input` from menehune.opt.wfu.edu. Run your program with:

```
gottlieb%  javac wordlist.java
gottlieb%  javac main.java
gottlieb%  java main < sample_input > word_counts
```

Turn In;

Save all your work in a directory named "Lab4". Change to your home directory (the parent directory of "Lab4"), and create a file named "lab4.tar" using the command:

```
tar cf lab4.tar Lab4
```

Use `sftp` to upload the file "lab4.tar" to your account on telesto.

I yam what I yam and that's all thats I yam.