Sets and Maps

Programming Projects:

1) An EmailStore

- Start Eclipse and create a new Java project called ‘lab6’

- Create a class called **EmailStore** which is going to manipulate a number of unique email addresses.

- Add an attribute called ‘emailAddresses’ which uses a Set to store a number of String type values, e.g.

```
Set<String> emailAddresses = new HashSet<String>();
```

- Implement a method called **addEmail(String email)** which stores the given parameter in the underlying set. This should return ‘true’ if the email was added, or ‘false’ if it was a duplicate (already existing in the underlying Set).

- Implement a second method called **hasEmail**(**String email**) which checks whether the given parameter represents an email address already stored (this should return a boolean type).

- Finally, implement a method called **displayEmails()** which iterates over the Set (using a for loop), and displays each stored email address to the console.

- Create a **Driver** class which creates an instance of **EmailStore** and makes multiple calls to the methods for testing. Ensure the test code attempts to add duplicate email addresses, and confirm they are not added by examining the return type.

Think you’re finished? Check you have commented your code and added Javadoc to the methods and classes.

2) WordCounter

- Create a class called **WordCounter** which is going to provide a method for counting unique words within a sentence. Within the class add an attribute which will be used to manage the count of each unique word, i.e.

```
/**
 * Maps words to their occurrence count.
 */
private final Map<String, Integer> wordMap = new HashMap<String, Integer>();
```

- Implement a method called **addWord(String word)** which uses the 'wordMap' attribute to maintain a number of occurrences of that word. Hint: if the word is not already in the map then then it should be inserted with the number '1' as its occurrence value. If it is already in the map, then the occurrence value should be increased.

- Add another method called **addSentence(String sentence)** which takes a full sentence (words separated by spaces), which it splits up and then calls the addWord() method for each word present.

- Add a final method called **outputResults()** which displays each word within the map, along with its associated occurrence count.

- Update the **Driver** class so that it creates an instance of the **WordCounter** class and tests its methods, e.g.

```
WordCounter wc = new WordCounter();

wc.addSentence("This sentence has the word has in it twice");
wc.outputResults();

// would generate the output
this : 1
sentence : 1
has : 2
the : 1
word : 1
in : 1
it : 1
twice : 1
```

Think you're finished? Check you have commented your code and added Javadoc to the methods and classes.