WRITING EXERCISE #1

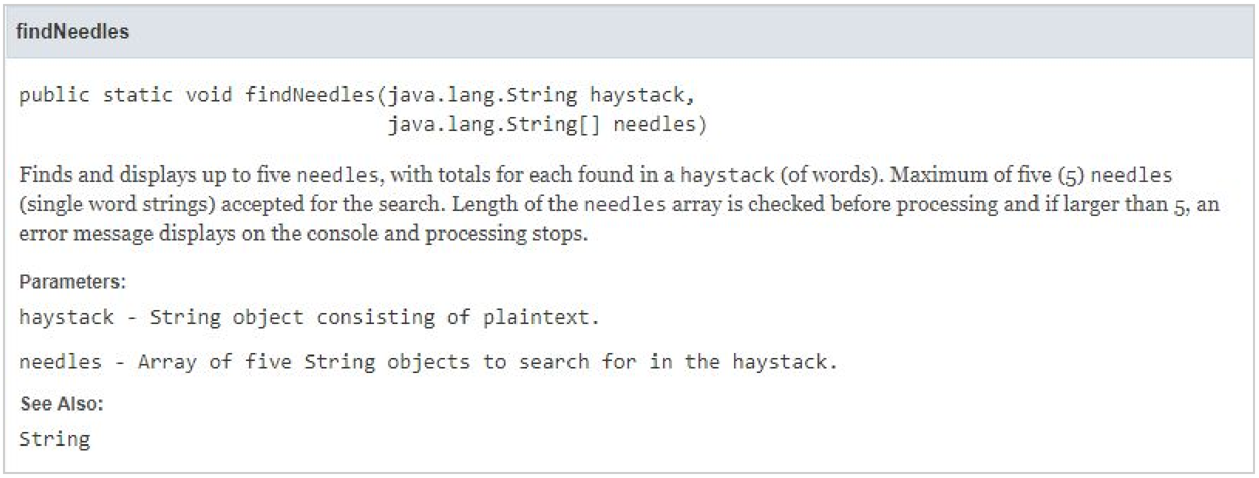
The findNeedles Method

Reference Documentation

```
/**
*  Finds and displays up to five <code>needles</code>, with totals for each
*  found in a <code>haystack</code> (of words).
*  Maximum of five (5) <code>needles</code> (single word strings) accepted
*  for the search. Length of the <code>needles</code> array is checked
*  before processing and if larger than 5, an error message displays on
*  the console and processing stops.
*
*  @param haystack String object consisting of plaintext.
*  @param needles Array of five String objects to search for in
*  the <code>haystack</code>.
*  @see java.lang.String
*  @author Google
*/

public static void findNeedles(String haystack, String[] needles) {
if (needles.length > 5) {
    System.err.println("Too many words! ");

} else {
    int[] countArray = new int[needles.length];
    for (int i = 0; i < needles.length; i++) {
        String[] words = haystack.split("[ \"\'\t\n\b\f\r]", 0);
        for (int j = 0; j < words.length; j++) {
          if (words[j].compareTo(needles[i]) == 0) {
              countArray[i]++;
            }
        }
    }
for (int j = 0; j < needles.length; j++) {
    System.out.println(needles[j] + ": " + countArray[j]);
    }
  }
}
```



findNeedles

```
public static void findNeedles(java.lang.String haystack,
                               java.lang.String[] needles)
```

Finds and displays up to five `needles`, with totals for each found in a `haystack` (of words). Maximum of five (5) `needles` (single word strings) accepted for the search. Length of the `needles` array is checked before processing and if larger than 5, an error message displays on the console and processing stops.

Parameters:

`haystack - String object consisting of plaintext.`

`needles - Array of five String objects to search for in the haystack.`

See Also:

`String`

About the findNeedles Method

The findNeedles method takes a String object (the haystack) and an array of Strings (the needles. No more than 5 needles are allowed, and this method checks the length of the String array before doing anything else.

Assuming the array contains 5 or less needles, the number of needles (the length of the array) is assigned to an array of integers, the variable named countArray. This will be used to count the number of instances of any given needle found in the haystack.

But first, the haystack is broken up into distinct words using the split method from Java class String. The split method is invoked on the haystack object, applying a regular expression that separates the text on word boundaries (space band), quotation marks, apostrophe, tab, newline, form-feed, and return characters. Trailing empty strings are removed and the result is the new array of Strings:

```
String[] words = haystack.split("[ \"\'\t\n\b\f\r]", 0);
```

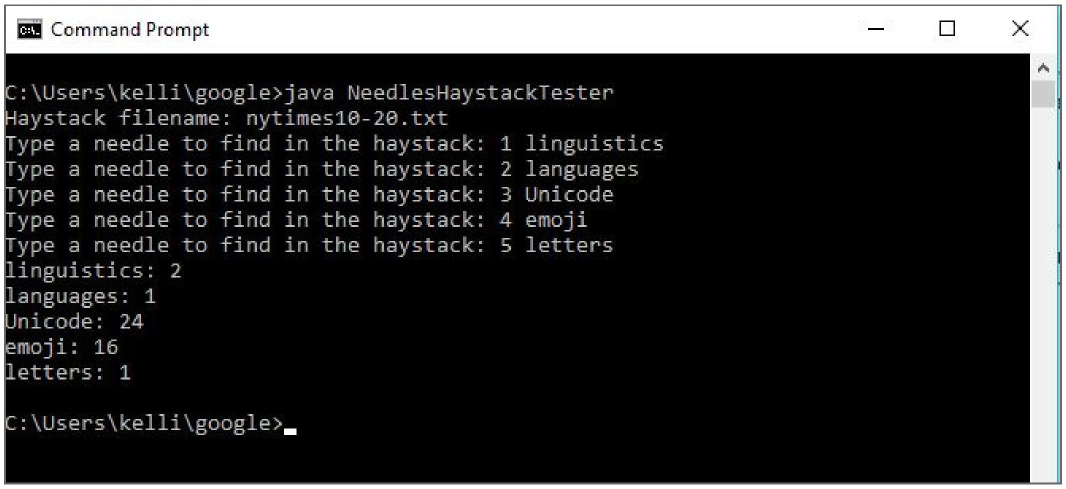
The length of this array—the number of words derived from the original haystack string—guides the nested for-loop that compares each word against each needle in turn, one by one. Every word from the haystack source text (words[j] is compared to the given needle needle[i] and for each match, the counter (countArray[j] is incremented for that specific needle. When the inner comparison loop finishes with the given needle, the outer loop moves to the next needle until the last needle has finished being processed and the words and their totals found in the haystack are printed.

Questions and Suggestions

1. I think it would better to check the array length for ‘needles’ elsewhere, before passing it to findNeedles. A separate method that allows users to enter the needles they are looking for could simply prompt for the series of up to five ‘needles’ or words, and when user is finished entering, that array could be passed to findNeedles without checking the length except to set the counter (i in the for loop, used to keep track of which needle is being compared to the words the haystack.)

2. Similarly, it might be better to break-up the haystack into the array of words before passing into findNeedles. That way, the processing inside findNeedles could simply process the two arrays. So in other words, create a separate method to break up the haystack String object into an array of Strings and pass that array to the findNeedles. The prompt that asks users to enter the needles could also prompt for a filename containing the text (haystack) to be searched, and that can be broken up in the separate method and only after this preliminary process is done would two arrays be send to the findNeedles method, one array consisting of all the words in the haystack and the second array containing the needles to find in the haystack. For example, prompt the

user for inputs as shown here (not the best UI, too wordy):



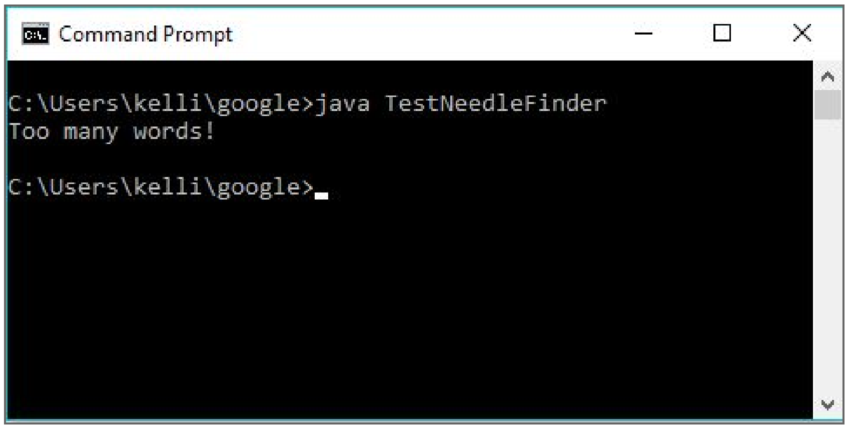
```
C:\Users\kelli\google>java NeedlesHaystackTester
Haystack filename: nytimes10-20.txt
Type a needle to find in the haystack: 1 linguistics
Type a needle to find in the haystack: 2 languages
Type a needle to find in the haystack: 3 Unicode
Type a needle to find in the haystack: 4 emoji
Type a needle to find in the haystack: 5 letters
linguistics: 2
languages: 1
Unicode: 24
emoji: 16
letters: 1

C:\Users\kelli\google>
```

3. Is there a need to limit the needles to an array length of 5?

4. Is there any limit to the size of the String object that can be passed into this method? I setup a test method that let me pass in text file containing almost 3,000 words, so I'm curious if using regular expressions to break up the text into the array of words might have issues beyond a certain size? Would StringTokenizer be better than regular expressions for this type of work?

5. If the limit of 5 needles does indeed need to be hard-coded into this method exactly as is, I'd suggest a friendlier message without the exclamation mark, such as "Too many words, try again," or "I can only look for 5 needles in the haystack."



```
C:\Users\kelli\google>java TestNeedleFinder
Too many words!

C:\Users\kelli\google>
```

6. This kind of string handling and counting seems like it might be a better fit for Python. An exercise for when i have more time.