1 Exercise #1 – findNeedles

My solution to this exercise is given in the following sections:

- **Javadoc Documentation** – The existing `findNeedles` code (as-is) with the necessary Javadoc tags and documentation comments added to it.
- **Improvement and Feature Suggestions** – An email to the developer, who wrote the `findNeedles` code. The email mentions the code improvements and feature suggestions, which are implemented in the Modified-Main.java file.

NOTE: One suggestion of updating the `findNeedles` method to return a HashMap object is implemented only as code comments.

- **Updated Code** – Contents of the Modified-Main.java file with the updated `findNeedles` code.

  I have run and verified this code on Repl.it. I have referred to the Oracle Java Documentation and stackoverflow.com for the same.

Javadoc Documentation

```
/**
* Compares up to five strings from a string array with all words of another
string.
* Prints an error and returns, if the array contains more than five strings.
* Else, prints the total numbers of matches found for each string in the array.
* Matches are exact and case-sensitive.
* @param  haystack The string containing words (substrings) separated by space,
single quote(‘), double quotes (“), tab (\t), line feed (\n), word boundary (\b),
form feed (\f), and carriage return (\r).
* @param needles The array of strings to compare with the words in the haystack
string.
*/
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]);
}
}
}
```

Improvement and Feature Suggestions

The draft of the email to the developer of the `findNeedles` code is as follows:

To: mini_developer@someorg.com

Subject: Requesting review – findNeedles documentation

Hi Mini,

I have checked the `findNeedles` code with documentation into the code repository (see Javadoc Documentation). Please review and verify the same.

Also, I wanted to suggest changing the following things that I noticed in the findNeedles code.

Important Changes

- The `findNeedles` method limits the number of needles to find to five (`needles.length > 5`). Is that limit necessary, as the method is capable of finding more than five needles? Could you consider removing this limit?

  If not, it might be better to declare an integer `MAX_NEEDLES` variable at the beginning of the method, initialize it with the desired value, and use it in the condition (`needles.length > MAX_NEEDLES`).

  ```
  int MAX_NEEDLES = 5;
  ```

  ```
  if (needles.length > MAX_NEEDLES){…}
  ```

- The code for splitting the `haystack` string into the `words` array can be moved out from the `for` loop to the beginning of the first `else` block. Thus, the `split` operation will be performed only once in the block.

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

  ```
  for (int i = 0; i < needles.length; i++) {…}
  ```

- The separate `for` loop at the method's end to print the needles and the corresponding count is not necessary. The print statement can be modified and moved inside the outer for loop as follows:

  ```
  for (int i = 0; i < needles.length; i++) {
  ```

```
for (int j = 0; j < words.length; j++) {
if (words[j].compareTo(needles[i]) == 0) {
countArray[i]++;
}
} //end-inner for
System.out.println(needles[i] + ": " + countArray[i]);
}//end-outer for
```

Then the existing loop can be removed

```
for (int j = 0; j < needles.length; j++) {….}
```

In addition, do consider making the following good-to-have changes for improving the code readability and performance.

NOTE: I have attached Modified-Main.java file with the above-mentioned and following changes for your review (see Updated Code).

Good-to-Have Changes

- Consider code indentation to improve readability.
- Consider providing case-sensitive matching as an option. The method signature can include a `boolean` case-sensitivity flag parameter. Then the method can be invoked with the desired flag value to indicate whether the string matching must be case-sensitive or not. In addition, appropriate string comparison method can be used according to the flag value.
- Consider removing any duplicate strings from the `needles` array. This will reduce a few `for` loop iterations.
- Consider using `for-each` loops to iterate through the needles and words arrays. This will reduce the clutter and the need of the local variables `i` and `j`.
- Consider using a simple integer `needle_count` variable to keep the match count instead of the existing `countArray` array.
- Consider updating the `findNeedles` method to returning an object, which contains the strings from the needles array and their match counts. A `HashMap` object that stores the strings and their respective counts as key-value pairs will be a suitable candidate for this purpose. This will be more useful that simply printing the needles and counts.

I hope you will consider these suggestions and incorporate them in the code.

Best regards,

Mugdha

Updated Code

The contents of the Modified-Main.java file with the updated `findNeedles` code are as follows:

```
import java.util.Arrays;
import java.util.HashSet;

class Main {
  public static void main(String[] args) {
  findNeedles("This is super big Chaos rubber black Shed big nancy Tango", args,
false); //calling the findNeedles method with case-insensitive matching option
  }
/**
* Compares every string in a string array with all words of another string.
* Prints the total numbers of matches found for each string in the array.
* Matches are exact, and optionally case-sensitive.
* @param  haystack The string containing words (substrings) separated by space,
single quote(‘), double quotes (“), tab (\t), line feed (\n), word boundary (\b),
form feed (\f), and carriage return (\r).
* @param needles The array of strings to compare with the words in the haystack
string.
* @param casesensFlag The boolean flag indicating whether matches must be case-
sensitive. Here, false indicates case-insensitive matches.
*/

  public static void findNeedles(String haystack, String[] needles, boolean
casesensFlag) {
  //Find needles in the haystack string
  String[] words = haystack.split("[ \"\'\t\n\b\f\r]", 0); //Split the haystack
string in substrings by delimiters, save the substrings in an array

  needles = new HashSet<String>(Arrays.asList(needles)).toArray(new String[0]);
//Remove duplicate needles
  //Optionally, if casesensFlag is false, all strings in needles can be converted
to uniform case (upper or lower) before removing duplicates

 /*Optionally, initialize a HashMap object to store the strings from needles and
their count
   HashMap<String, int> foundNeedles = new HashMap<String, int>(); */

  //Compare every needle in the needles with all word strings in the haystack
string
  for (String needle : needles) {
    int needle_count = 0;  //For every new needle, set the count to 0
```

```
    for (String word : words){
      if (casesensFlag == true){ //Case-sensitive match
        if (word.compareTo(needle) == 0) needle_count++; //If a word matches a
needle, increment the  match count for the needle
      } else { //Case-insensitive match
        if (word.compareToIgnoreCase(needle) == 0) needle_count++; //If a word
matches with needle irrespective of case, increment the  match count for the
needle
      } //end-casesens if-else

    } //end-inner for words
    System.out.println(needle + ": " + needle_count); //Print the needle and its
total match count
    /*Optionally, add the needle and count to the HashMap */
                /* foundNeedles.put(needle, needle_count);            */
  }//end-outer for needles
               /* Optionally, return the HashMap foundNeedles from here. Requires
updating the method signature to public static HashMap findNeedles(...) */

  } //end-findNeedles
} //end-class Main
```