API Reference

API Reference: `findNeedles` Method

Overview

The `findNeedles` method scans a given string (`haystack`) and counts occurrences of specific words (`needles`). If the number of words to search for exceeds five, it logs an error message. Otherwise, it counts how many times each "needle" (word to search for) appears in the haystack and prints the result for each needle.

This method is useful for scenarios requiring word counting from a body of text. The target audience for this document is experienced Java developers familiar with Java standard libraries and common data structures.

Method Signature

Java

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

Parameters:

- `haystack` (`String`): The input text where you are searching for occurrences of specific words. The text is tokenized based on whitespace and punctuation.
- `needles` (`String`[]): An array of words you want to search for within the `haystack`. If more than 5 words are provided, an error message is printed and no further processing occurs.

Returns:

None: This method has no return value. It prints the count of each word (needle) in the console or an error message in case of too many words.

Behavior:

1. **Word Limit Check**: If the number of elements in the `needles` array exceeds 5, the method immediately prints:

   ```plaintext
   Too many words!
   ```

   No further execution occurs.

2. **Text Splitting**: The `haystack` string is split into words using a regular expression based on whitespace and common punctuation (`"`, `'`, tabs, newlines, etc.). This results in an array of words.

3. **Needle Counting**: For each word in the `needles` array, the method counts how often it appears in the `haystack`. Comparisons are case-sensitive, meaning "word" and "Word" will be considered different.

4. **Output**: After counting, the method prints a result for each word in the `needles` array in the following format:

```plaintext
needle[i]: count
```

where `needle[i]` is the word and `count` is the number of times it appeared in the `haystack`.

Example Usage:

```java
public class Example {
                public static void main(String[] args) {
                String haystack = "The quick brown fox jumps over the lazy dog. The fox is clever.";
                String[] needles = { "fox", "dog", "cat" };
                findNeedles(haystack, needles);
                }
            }
```

Output:

```plaintext
fox: 2
dog: 1
cat: 0
```

Example with Too Many Needles:

java

```java
public class Example {
    public static void main(String[] args) {
        String haystack = "This is a test sentence for counting words.";
        String[] needles = { "this", "is", "a", "test", "sentence", "extra" };
        findNeedles(haystack, needles);
    }
}
```

Output:

plaintext

```
Too many words!
```

Edge Cases:

- **Case Sensitivity**: The word comparison is case-sensitive. If you need case-insensitive matching, consider converting both `haystack` and `needles` to lowercase before comparing.
- **Special Characters**: The regular expression used for splitting words may not handle all punctuation or special cases, so further adjustments might be necessary for more complex tokenization.

- **Empty Strings**: If the `haystack` is empty or contains no words, the count for each needle will be `0`.

Considerations for Optimization:

Efficiency: The method runs a nested loop over `needles` and words in the `haystack`. This can be inefficient for large inputs, so consider optimizing by using more advanced data structures (e.g., HashMap) for faster lookups in cases of large `haystack` strings or multiple `needles`.

Conclusion

The `findNeedles` method is a simple utility for counting word occurrences within a string. It's straightforward to use, but developers should be aware of its limitations regarding case sensitivity, word tokenization, and performance for large datasets.