WORD COUNTER:

Counting the number of word occurrences in a string is a fairly easy task, but has several approaches to doing so. You have to account for the efficiency of the method as well, since you'll typically want to employ automated tools when you don't want to perform manual labor - i.e. when the search space is large.

The Collections.frequency() method provides a much cleaner, higher-level implementation, which abstracts away a simple for loop, and checks for both identity (whether an object is another object) and equality (whether an object is equal to another object, depending on the qualitative features of that object).

In this short guide, we've taken a look at how to count word occurrences for a target word, in a string in Java. We've started out by splitting the string and using a simple counter, followed by using the Collections helper class, and finally, using Regular Expressions.

In the end, we've benchmarked the methods, and noted that the performance isn't linear, and depends on the search space. For longer input texts with many matches, splitting arrays seems to be the most performant. Try all three methods on your own, and pick the most performant one.

In this tutorial, we have looked at ways to count words using several approaches. We can pick any depending on our particular use-case.

Count the number of words in the String:

In this article, you will learn how to count the number of words in a string using Java. To count the number of words in a given String in Java, you can use various approaches. One simple and common way is to split the string based on whitespace characters (such as spaces, tabs, or newlines) and then count the number of resulting substrings.

1. Using whitespace characters

In this approach, we use a boolean variable inWord to track whether we are currently inside a word or not. We iterate through the characters of the input string, and when we encounter a whitespace character, we mark the end of a word (inWord = false). When we encounter a non-whitespace character, we check if it's the start of a new word (!inWord). If so, we increment the word count and set inWord = true.

This approach doesn't rely on regular expressions or splitting the string, making it slightly more efficient for large strings with many words.

2. Using the split method

In this program, we use the split method of the String class, which splits the input string into an array of substrings based on the given regular expression pattern. The pattern "\\s+" is used, which represents one or more whitespace characters (spaces, tabs, or newlines).

Before splitting, we use input.trim() to remove any leading or trailing whitespace from the input string. This ensures that leading/trailing spaces do not affect the word count.

A string is a class in Java that stores a series of characters enclosed within double quotes. Those characters act as String-type objects. The aim of this article is to write Java programs that count words in a given string. Counting words of the given strings can be solved using the string manipulation techniques. In Java interviews, questions from string manipulation are very frequent, hence, it is important to understand this problem properly.

Conclusion

In this article, we have learned what is a string and how we can check the number of words of a given string in Java. For this string manipulation operation, we have used the in-built methods of String such charAt() and split() along with the while and for loops.