Count total number of occurrences of Sub-String in a String.

Example:

String str = “banana”;

String substring = “an”;

Output = 2

B**an***an*a, an - is repeated twice.

So, how to write a code for this problem?

|  |  |  |
| --- | --- | --- |
| String input | String substring | Output |
| null | Not null | -1 |
| Not null | null | -1 |
| Null | null | -1 |
| Not null | Not null | >=0 |

Now we have our corner cases we can design our test cases for the method.

Let us Assume that the class name is StringUtils and method to be implemented is countOccurencesofString(String str, String subString).

Let us first design few test cases around this method.

/\*\*

\* Input String is null and subString has length.

\* In this scenario it should return -1.

\* \*/

@Test

**public** **void** countOccurencesofStringNull(){

Assert.*assertEquals*(-1, StringUtils.*countOccurencesofString*(**null**, "a"));

}

/\*\*

\* Input String has length and subString is null.

\* In this scenario it should return -1.

\* \*/

@Test

**public** **void** countOccurencesofStringSubStringNull(){

Assert.*assertEquals*(-1, StringUtils.*countOccurencesofString*("test", **null**));

}

/\*\*

\* Input String is null and subString is also null.

\* In this scenario it should return -1.

\* \*/

@Test

**public** **void** countOccurencesofStringBothNull(){

Assert.*assertEquals*(-1, StringUtils.*countOccurencesofString*(**null**, **null**));

}

/\*\*

\* Input String has length and subString also has length.

\* String str="a",subString="a";

\* subString is repeated one time in input String.

\* In this scenario it should return 1.

\* \*/

@Test

**public** **void** countOccurencesofStringSingleLength(){

Assert.*assertEquals*(1, StringUtils.*countOccurencesofString*("a", "a"));

}

/\*\*

\* Input String has length and subString also has length.

\* String str="banana",subString="an";

\* subString is repeated twice in input String.

\* In this scenario it should return 2.

\* \*/

@Test

**public** **void** countOccurencesofLongLength(){

Assert.*assertEquals*(2, StringUtils.*countOccurencesofString*("banana", "an"));

}

Now we will write a method that satisfies our contract.

First check if str and substring is not null and has length i.e size of string must be greater than 0. Hopefully we have this method that calculates it.

/\*\*

\* Checks for length.

\* If String is not null and length of String > 0 return true

\* else return false

\*

\* Input Output

\* str=null false;

\* str="" false;

\* str="a" true

\*

\* **@param** str

\* the str

\* **@return** true, if successful

\*/

**public** **static** **boolean** hasLength(String str) {

**return** (str != **null** && str.length() > 0);

}

/\*\*

\* Checks for input and substring conditions null and length.

\*/

**if** (!*hasLength*(str) || !*hasLength*(subString)) {

**return** -1;

}

Now let us handle one more condition. What should happen when the length of substring is greater than input string?

**if**(subString.length()>str.length()){

**return** 0;

}

If the substring length is greater than input string length then return 0 as there won’t be any matching substring due to its length

Now last phase. Count number of occurrences if they exist in input string.

**int** count = 0;

**int** pos = 0;

**int** index;

**while** ((index = str.indexOf(subString, pos)) != -1) {

count++;

pos = index + subString.length();

}

Let us set count and current position, pos as 0.

Take index = str.indexOf(subString, pos)

If index == -1 then there is no such string. Return.

Else increment count as count++ and set new position as pos = index + subString.length();

Check if there is any other such substring in input string.

Entire code is as below.

**public** **static** **int** countOccurencesofString(String str, String subString) {

**if** (!*hasLength*(str) || !*hasLength*(subString)) {

**return** -1;

}

**if**(subString.length()>str.length()){

**return** 0;

}

**int** count = 0;

**int** pos = 0;

**int** index;

**while** ((index = str.indexOf(subString, pos)) != -1) {

count++;

pos = index + subString.length();

}

**return** count;

}

Now we run the test cases as against this method.