Check if 2 strings are Anagrams of each other.

Two strings are anagramof each other if all the characters of one string matches another string's characters.

Example:

str = army

anagram = mary

ouput = true

str = abc

anagram = cbd

output = false

There are few ways to do it.

We will discuss this ways one by one including their time and space complexity.

1) Sort both the inputs and check for their equality.

2) Use StringBuilder to store one String into it. Now delete that char from StringBuilder that we encounter in anagram String. If StringBuilder is empty at the end of processing then return true else return false.

3) Built an int array of length 256. Now for every character in string input increment the counter in array. Now scan it for anagram array and decrement value. If the value becomes -1 then return false. If entire array is processed then just return true.

4) Using subString(). search for the character of input string in anagram string and remove it using subString().

Below are the test cases for all the methods described above.

**package** strings;

**import** **static** org.junit.Assert.\*;

**import** junit.framework.Assert;

**import** org.junit.Test;

**public** **class** StringAnagramTestCases {

//----------------------------------------------------------------------------------------------

//anagramByCharacterArraySorting

@Test

**public** **void** anagramByCharacterArraySorting1() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByCharacterArraySorting*(**null**,"aa"));

}

@Test

**public** **void** anagramByCharacterArraySorting2() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByCharacterArraySorting*("aa",**null**));

}

@Test

**public** **void** anagramByCharacterArraySorting3() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByCharacterArraySorting*(**null**,**null**));

}

@Test

**public** **void** anagramByCharacterArraySorting4() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByCharacterArraySorting*("a","aa"));

}

@Test

**public** **void** anagramByCharacterArraySorting5() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByCharacterArraySorting*("gth","thg"));

}

@Test

**public** **void** anagramByCharacterArraySorting6() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByCharacterArraySorting*("dog","dog"));

}

//----------------------------------------------------------------------------------------------

//anagramByStringBuilder

@Test

**public** **void** anagramByStringBuilder1() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByStringBuilder*(**null**,"aa"));

}

@Test

**public** **void** anagramByStringBuilder2() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByStringBuilder*("aa",**null**));

}

@Test

**public** **void** anagramByStringBuilder3() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByStringBuilder*(**null**,**null**));

}

@Test

**public** **void** anagramByStringBuilder4() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByStringBuilder*("a","aa"));

}

@Test

**public** **void** anagramByStringBuilder5() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByStringBuilder*("gth","thg"));

}

@Test

**public** **void** anagramByStringBuilder6() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByStringBuilder*("dog","dog"));

}

//----------------------------------------------------------------------------------------------

//anagramByIntArray

@Test

**public** **void** anagramByIntArray1() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByIntArray*(**null**,"aa"));

}

@Test

**public** **void** anagramByIntArray2() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByIntArray*("aa",**null**));

}

@Test

**public** **void** anagramByIntArray3() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByIntArray*(**null**,**null**));

}

@Test

**public** **void** anagramByIntArray4() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramByIntArray*("a","aa"));

}

@Test

**public** **void** anagramByIntArray5() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByIntArray*("gth","thg"));

}

@Test

**public** **void** anagramByIntArray6() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramByIntArray*("dog","dog"));

}

//----------------------------------------------------------------------------------------------

//anagramBySubString

@Test

**public** **void** anagramBySubString1() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramBySubString*(**null**,"aa"));

}

@Test

**public** **void** anagramBySubString2() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramBySubString*("aa",**null**));

}

@Test

**public** **void** anagramBySubString3() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramBySubString*(**null**,**null**));

}

@Test

**public** **void** anagramBySubString4() {

Assert.*assertEquals*(**false**, StringAnagram.*anagramBySubString*("a","aa"));

}

@Test

**public** **void** anagramBySubString5() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramBySubString*("gth","thg"));

}

@Test

**public** **void** anagramBySubString6() {

Assert.*assertEquals*(**true**, StringAnagram.*anagramBySubString*("dog","dog"));

}

}

Below is the program:

**package** strings;

**import** java.util.Arrays;

**public** **class** StringAnagram {

**public** **static** **void** main(String[] args) {

}

**public** **static** **boolean** anagramByCharacterArraySorting(String str, String anagram) {

/\*\*

\* If the input string and anagram string are null or both are same

\* string with same order of alphabets then return true.

\* \*/

**if** (str == anagram) {

**return** **true**;

}

/\*\*

\* if either of input is null then return false.

\* \*/

**else** **if** (str == **null** || anagram == **null**) {

**return** **false**;

}

/\*\*

\* This is extra check and quite important one.

\* If the length are not same then we conclude that it is not anagram.

\* \*/

**else** **if** (str.length() != anagram.length()) {

**return** **false**;

}

/\*\*

\* convert both of the inputs to character array.

\* \*/

**char**[] strArr = str.toCharArray();

**char**[] anagramArr = anagram.toCharArray();

/\*\*

\* Now sort both the array

\* \*/

Arrays.*sort*(strArr);

Arrays.*sort*(anagramArr);

/\*\*

\* Arrays.equals([] a, char[] a2) is used to check for array equality.

\* this method return true if they contain same elements in same order.

\* \*/

**return** Arrays.*equals*(strArr, anagramArr);

}

**public** **static** **boolean** anagramByStringBuilder(String str, String anagram){

/\*\*

\* If the input string and anagram string are null or both are same

\* string with same order of alphabets then return true.

\* \*/

**if** (str == anagram) {

**return** **true**;

}

/\*\*

\* if either of input is null then return false.

\* \*/

**else** **if** (str == **null** || anagram == **null**) {

**return** **false**;

}

/\*\*

\* This is extra check and quite important one.

\* If the length are not same then we conclude that it is not anagram.

\* \*/

**else** **if** (str.length() != anagram.length()) {

**return** **false**;

}

/\*\*

\* Build a StringBuilder object and push input str into it.

\* \*/

StringBuilder sb=**new** StringBuilder(str);

/\*\*

\* For every character in anagram

\* take the index of character

\* if index !=-1 then delete it

\* else return false, because we found a character that it not in string input.

\* \*/

**for** (**char** ch : anagram.toCharArray()) {

**int** index=sb.indexOf(""+ch);

**if** (index != -1) {

sb.deleteCharAt(index);

} **else** {

**return** **false**;

}

}

/\*\*

\* If the length of StringBuilder is 0 then return true as we

\* found all characters in anagram that matched with input string

\*/

**if**(sb.length()==0){

**return** **true**;

}

//else return false

**return** **false**;

}

**public** **static** **boolean** anagramByIntArray(String str, String anagram){

/\*\*

\* If the input string and anagram string are null or both are same

\* string with same order of alphabets then return true.

\* \*/

**if** (str == anagram) {

**return** **true**;

}

/\*\*

\* if either of input is null then return false.

\* \*/

**else** **if** (str == **null** || anagram == **null**) {

**return** **false**;

}

/\*\*

\* This is extra check and quite important one.

\* If the length are not same then we conclude that it is not anagram.

\* \*/

**else** **if** (str.length() != anagram.length()) {

**return** **false**;

}

**int**[] chars = **new** **int**[256];

**for** (**char** ch : str.toCharArray()) {

chars[ch]++;

}

**for** (**int** i = 0; i < anagram.length(); i++) {

**if** (--chars[anagram.charAt(i)] < 0) {

**return** **false**;

}

}

**return** **true**;

}

**public** **static** **boolean** anagramBySubString(String str, String anagram){

/\*\*

\* If the input string and anagram string are null or both are same

\* string with same order of alphabets then return true.

\* \*/

**if** (str == anagram) {

**return** **true**;

}

/\*\*

\* if either of input is null then return false.

\* \*/

**else** **if** (str == **null** || anagram == **null**) {

**return** **false**;

}

/\*\*

\* This is extra check and quite important one.

\* If the length are not same then we conclude that it is not anagram.

\* \*/

**else** **if** (str.length() != anagram.length()) {

**return** **false**;

}

**for** (**char** c : str.toCharArray()) {

**int** index = anagram.indexOf(c);

**if** (index != -1) {

anagram = anagram.substring(0, index)

+ anagram.substring(index + 1, anagram.length());

}**else**{

**return** **false**;

}

}

**return** anagram.isEmpty();

}

}