Check if String contains unique elements or not.

Given : We are or we will be given a String and we need to check whether the string contains all unique characters or not. If all characters in String are unique then return true else return false.

Example:

|  |  |
| --- | --- |
| Input String | Output |
| null | false |
| Mississippi | false |
| Work | true |
| amazon | false |
| Amazon | true |

For this question we are considering that the input string is case sensitive. So Amazon and amazon will have different outputs.

So now we can design our test cases accordingly.

@Test

**public** **void** testForNull() {

Assert.*assertEquals*(**false**, UniqueCharacters.*isUniqueChar*(**null**));

}

@Test

**public** **void** testForEmptyString() {

Assert.*assertEquals*(**true**, UniqueCharacters.*isUniqueChar*(""));

}

@Test

**public** **void** testForDuplicateCharacters() {

Assert.*assertEquals*(**false**, UniqueCharacters.*isUniqueChar*("amazon"));

}

@Test

**public** **void** testForCaseSensitivity() {

Assert.*assertEquals*(**true**, UniqueCharacters.*isUniqueChar*("Amazon"));

}

Now let us code for this question.

First we check for corner condition regarding string null and length is 0.

/\*\*

\* Is the input string is null then return false.

\* \*/

**if** (str == **null**) {

**return** **false**;

}

Now let us write code for string length = 0.

/\*\*

\* If String length is 0 then we return true. Actually this can work as

\* per the question requirements.

\* \*/

**else** **if** (str.length() == 0) {

**return** **true**;

}

A word will consist of combination of 256 characters as there is ASCII for only 256 characters.

Now we take a Boolean flag array and set the ASCII for the character as true. And if we encounter that bit again we return true. Else if condition will never execute and after traversing of string it will return true. Let us put this thing in code.

/\*\*

\* Now we declare the flag array which can be used to set the ASCII of

\* the index as true.

\*

\* If we encounter that character again then we will simply return false.

\* If we do not encounter that bit till the end then we return true.

\* \*/

**boolean** character[] = **new** **boolean**[256];

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

/\*\*

\* int ascii = str.charAt(i);

\* ascii will contain the ASCII for the character.

\* a - 97

\* so character[97] = true;

\* So we will set the ascii bit of boolean array.

\* \*/

**int** ascii = str.charAt(i);

**if** (character[ascii]) {

**return** **false**;

} **else** {

character[ascii] = **true**;

}

}

**return** **true**;

Code for entire method is as follows:

/\*\*

\* The following method is used check if the string has all unique

\* characters or not.

\*

\* **@param** str

\* the str

\* **@return** if is unique string, return true else, return false.

\*/

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

/\*\*

\* Is the input string is null then return false.

\* \*/

**if** (str == **null**) {

**return** **false**;

}

/\*\*

\* If String length is 0 then we return true. Actually this can work as

\* per the question requirements.

\* \*/

**else** **if** (str.length() == 0) {

**return** **true**;

}

/\*\*

\* Now we declare the flag array which can be used to set the ASCII of

\* the index as true.

\*

\* If we encounter that character again then we will simply return false.

\* If we do not encounter that bit till the end then we return true.

\* \*/

**boolean** character[] = **new** **boolean**[256];

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

/\*\*

\* int ascii = str.charAt(i);

\* ascii will contain the ASCII for the character.

\* a - 97

\* so character[97] = true;

\* So we will set the ascii bit of boolean array.

\* \*/

**int** ascii = str.charAt(i);

**if** (character[ascii]) {

**return** **false**;

} **else** {

character[ascii] = **true**;

}

}

**return** **true**;

}