12.JUnitTestforStringComparison

Aim:

TodevelopaJUnittestthatcomparestwostringsandvalidatesiftheyareequalusing assertEquals assertions.

Algorithm:

1. **Step1:**Createamethod tocomparetwostrings.
   * Themethodwilltaketwostringsasinputandcomparethemusingtheequals method.
2. **Step2:**WriteaJUnittestclasstoverifythecorrectnessofthestringcomparison functionality.
   * Thetest willuse theassertEqualsmethodfrom JUnittocomparethestrings.
3. **Step3:**Definetestcaseswithvariousinputcombinations,includingmatchingstrings, different strings, and edge cases like empty strings or null values.
4. **Step4:**Executethetestsandvalidatetheoutput against theexpected results.

CodeImplementation:

1. *StringReversalMethod*

**package** ex3.strcmp;

**publicclass** Strcmp {

// Method to compare two strings (case-sensitive)

**publicboolean** areStringsEqual(String s1, String s2) {

**if** (s1 == **null** || s2 == **null**) **returnfalse**;

**return**s1.equals(s2);

}

// Method to compare two strings ignoring case

**publicboolean** areStringsEqualIgnoreCase(String s1, String s2) {

**if** (s1 == **null** || s2 == **null**) **returnfalse**;

**return**s1.equalsIgnoreCase(s2);

}

// Method to check if one string contains another

**publicboolean** containsSubstring1(String full, String part) {

**if** (full == **null** || part == **null**) **returnfalse**;

**return**full.contains(part);

}

**publicboolean** containsSubstring(String full, String part) {

// **TODO** Auto-generated method stub

**returnfalse**;

}

}

*JUnitTestClass*

**package** ex3.strcmp;

**importstatic** org.junit.jupiter.api.Assertions.\*;

**import**org.junit.jupiter.api.Test;

**publicclass** StrcmpTest {

@Test

**void** testCase1\_sameStrings() {

Strcmp util = **new** Strcmp();

*assertTrue*(util.areStringsEqual("hello", "hello")); // Expected: true

}

@Test

**void** testCase2\_differentStrings() {

Strcmp util = **new** Strcmp();

*assertFalse*(util.areStringsEqual("hello", "world")); // Expected: false

}

@Test

**void** testCase3\_oneNull() {

Strcmp util = **new** Strcmp();

*assertFalse*(util.areStringsEqual("hello", **null**)); // Expected: false

}

@Test

**void** testCase4\_bothNull() {

Strcmp util = **new** Strcmp();

*assertFalse*(util.areStringsEqual(**null**, **null**)); // Expected: false

}

@Test

**void** testCase5\_emptyStrings() {

Strcmp util = **new** Strcmp();

*assertTrue*(util.areStringsEqual("", "")); // Expected: true

}

}

Sample Input:

# TestCase1:

* + Input:"hello","hello"
  + ExpectedOutput: true

# TestCase2:

* + Input:"hello","world"
  + ExpectedOutput: false

# TestCase3:

* + Input:"hello", null
  + ExpectedOutput: false

# TestCase4:

* + Input:null,null
  + ExpectedOutput: false

# TestCase5:

* + Input:"",""(Emptystrings)
  + ExpectedOutput:true Sample Output:

# TestCase1:

* + Input:"hello","hello"
  + Output:true(Thestrings are equal)

# TestCase2:

* + Input:"hello","world"
  + Output:false(Thestrings are different)

# TestCase3:

* + Input:"hello", null
  + Output:false(Onestringis null)

# TestCase4:

* + Input:null,null
  + Output:false(Bothstringsare null)

# TestCase5:

* + Input:"",""(Emptystrings)
  + Output:true(Bothareemptystringsand consideredequal)

Results:

Thefollowingoutcomesareexpectedfrom runningthe test cases:

* **TestCase1:**"hello"is equalto "hello",sothetest passes.
* **TestCase2:**"hello"isnot equalto"world",so thetest passes.
* **TestCase3:**A non-nullstringis not equalto null,sothe test passes.
* **TestCase4:**Twonullvaluesarenotconsideredequalbythemethod,sothetest passes.
* **TestCase5:**Two emptystrings""areconsideredequal,sothetest passes.

Inconclusion,theJUnittestsconfirmthatthecompareStringsmethodworkscorrectlyfor variousinputscenarios,includingequalstrings,differentstrings,nullvalues,andempty.



