Using assertNull() and assertNotNull() methods together

This lesson demonstrates how to use assertNull() and assertNotNull() methods together in JUnit 5 to assert test conditions.

Demo

Step 1 - Create a Java class in Eclipse as discussed in previous lessons.

Step 2 - Give it a name as, StringUtils. `

 StringUtils.java

```java
package com.hubberspot.junit5.assertions;

public class StringUtils {

    public static String reverse(String input) {
        if(input == null) {
            return null;
        }

        if(input.length() == 0) {
            return "";
        }

        char[] charArray = input.toCharArray();
        int start = 0;
        int end = input.length() - 1;

        while(start < end) {
            char temp = charArray[start];
            charArray[start] = charArray[end];
```

```
                    charArray[end] = temp;
                    start++;
                    end--;
            }

            return new String(charArray);
        }
}
```

Class Under Test - StringUtils

StringUtils is our class under test. It has one method as, `reverse()`. This method takes in a String and returns reverse of it.

For example -

1. If we provide input String as, "ABCD", it returns back "DCBA".

2. If we provide input String as, "Student", it returns back "tnedutS".

3. If we provide input String as, **null**, it returns back **null**.

4. If we provide input String as, "", it returns back "" String.

Step 3 - Create a test class by name, "`StringUtilsTest`". This test class will demonstrate how to use `assertNull()` and `assertNotNull()` methods together to pass previous lessons failed test cases.

StringUtilsTest.java

StringUtils.java

```
package io.educative.junit5;

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

class StringUtilsTest {

        @Test
        void givenNullString_whenReverseIsCalled_thenNullIsReturned() {
                String actual = StringUtils.reverse((null));
                assertNull(actual);
        }

        @Test
        void givenEmptyString_whenReverseIsCalled_thenEmptyStringIsReturned() {
                String actual = StringUtils.reverse((""));
                assertNotNull(actual);
```

```
        }

        @Test

        void givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned() {
                String actual = StringUtils.reverse(("ABCD"));
                assertNotNull(actual);
        }

}
```
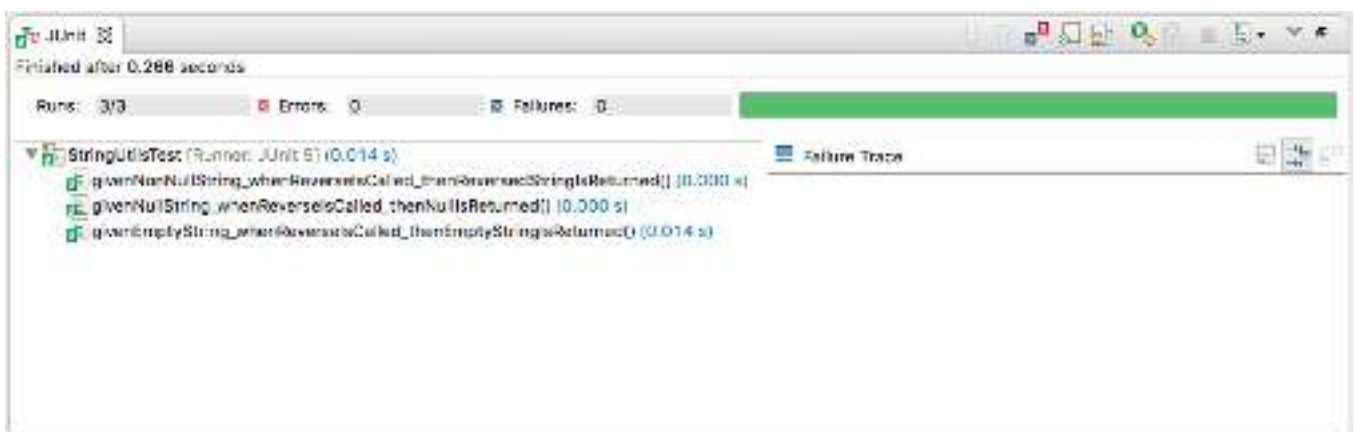
> You can perform code changes to above code widget, run and practice different outcomes.

Step 4 - Run StringUtilsTest class as Junit Test.

Output

Explanation -

The order of execution of test cases depends on Junit 5. In StringUtilsTest class there are 3 @Test methods:-

1. `givenNullString_whenReverseIsCalled_thenNullIsReturned()` - It tests the scenario that when **null** is provided to `reverse()` method of StringUtils class, then **null** is returned. So, on **line 11** providing `assertNull()` asserts that actual value returned is null. Thus, it passes the Junit test case because actual value returned is null.

2. `givenEmptyString_whenReverseIsCalled_thenEmptyStringIsReturned()` - It tests the scenario that when "" is provided to `reverse()` method of StringUtils class, then "" is returned. Here, return value is empty string

which is not null. So, on **line 17** providing `assertNotNull()` asserts that

actual value returned is **not null**. Thus, it passes the Junit test case because actual value returned is not null.

3. `givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned` - It tests the scenario that when **ABCD** is provided to `reverse()` method of StringUtils class, then **DCBA** is returned. Here, return value is not null. So, on **line 23** providing `assertNotNull()` asserts that actual value returned is **not null**. Thus, it passes the Junit test case because actual value returned is not null.

In the next lesson, we will look into `assertEquals()` assertion.