assertNotNull() method

This lesson demonstrates how to use assertNotNull() method in JUnit 5 to assert test conditions.

WE'LL COVER THE FOLLOWING

assertNotNull() method

Assertions API provide static `assertNotNull()` method. This method helps us in validating that particular object is **not** null. This method takes the actual value and checks whether it is **null** or not.

- If the actual value is **not null** then test case will pass.
- If the actual value is **null** then test case will fail.

There are basically three overloaded methods for assertNotNull:-

```
public static void assertNotNull(Object actual)

public static void assertNotNull(Object actual, String message)

public static void assertNotNull(Object actual, Supplier<String> messageSupplier)
```

1. `assertNotNull(Object actual)` - It assert whether actual value is **not null**.
2. `assertNotNull(Object actual, String message)` - It assert whether actual value is **not null**. In case, if the actual value is **null** then test case will fail with the provided message.

3. `assertNotNull(Object actual, Supplier<String> messageSupplier)` - It assert whether actual value is **not null**. In case, if the actual value is **null** then test case will fail with the provided message through Supplier function. The main advantage of using Supplier function is that it lazily evaluates to String only when the test case fails.

Demo

In our previous lesson, we created a class by name, StringUtils. It has the reverse() method, which reverses the String passed to it.

assertNotNull method

StringUtils.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.

In our previous lesson, we created a test class by name, “`StringUtilsTest`”. This test class will demonstrate all overloaded `assertNotNull()` methods.

StringUtilsTest2.java

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

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

import java.util.function.Supplier;
import org.junit.jupiter.api.Test;

class StringUtilsTest2 {

        // ******** assertNotNull Example - Start **********
        @Test
```

```
        void givenNullString_whenReverseIsCalled_thenNullIsReturned() {
                String actual = StringUtils.reverse((null));
                String message = "Actual String should not be null !!! ";

                // assertNotNull with message
                assertNotNull(actual, message);
        }

        @Test
        void givenNullString_whenReverseIsCalled_thenNullIsReturned2() {
                String actual = StringUtils.reverse((null));
                Supplier<String> messageSupplier = () -> "Actual String should not be null !!

                // assertNotNull with Java 8 MessageSupplier
                assertNotNull(actual, messageSupplier);
        }

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

                // assertNotNull without message
                assertNotNull(actual);
        }

        @Test
        void givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned() {
                String actual = StringUtils.reverse(("ABCD"));

                // assertNotNull without message
                assertNotNull(actual);
        }

        // ******** assertNotNull Example - End **********
}
```

StringUtilsTest2.java

StringUtils.java

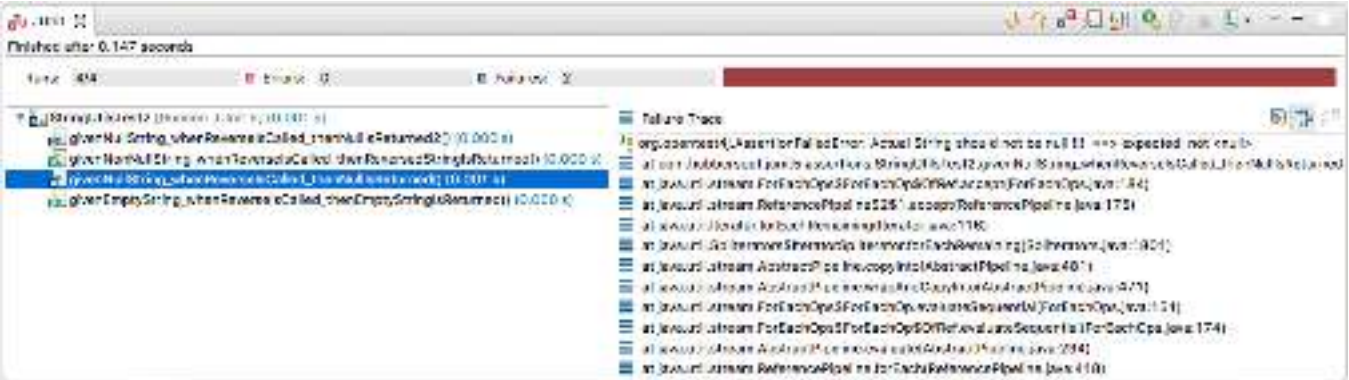
```
package io.educative.junit5;

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

import java.util.function.Supplier;
import org.junit.jupiter.api.Test;

class StringUtilsTest2 {

        // ******** assertNotNull Example - Start **********
        @Test
        void givenNullString_whenReverseIsCalled_thenNullIsReturned() {
                String actual = StringUtils.reverse((null));
                String message = "Actual String should not be null !!! ";

                // assertNotNull with message
                assertNotNull(actual, message);
        }
```

```
        @Test
        void givenNullString_whenReverseIsCalled_thenNullIsReturned2() {

                String actual = StringUtils.reverse((null));
                Supplier<String> messageSupplier = () -> "Actual String should not be null

                // assertNotNull with Java 8 MessageSupplier
                assertNotNull(actual, messageSupplier);
        }

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

                // assertNotNull without message
                assertNotNull(actual);
        }

        @Test
        void givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned() {
                String actual = StringUtils.reverse(("ABCD"));

                // assertNotNull without message
                assertNotNull(actual);
        }

        // ******** assertNotNull Example - End **********
}
```

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

Run StringUtilsTest class as JUnit Test.

Output



Explanation -

The order of execution of test cases depends on Junit 5. In StringUtilsTest2 class there are 4 @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 17** providing `assertNotNull()` asserts that actual value returned is **not null**. Thus, it fails the Junit test case because actual value returned is **null**. In this test case, we are using overloaded assertNotNull() method, which takes **String message** as second argument. As, this test case doesn't satisfy assertion condition, it fails and give "`AssertionFailedError:` Actual String should not be null !!! ==> expected: not ". It gives `AssertionFailedError` followed by **String message** we provide to assertNotNull() method.

2. `givenNullString_whenReverseIsCalled_thenNullIsReturned2()` - It tests the scenario that when **null** is provided to `reverse()` method of StringUtils class, then **null** is returned. So, on **line 26** providing `assertNotNull()` asserts that actual value returned is **not null**. Thus, it fails the Junit test case because actual value returned is **null**. In this test case, we are using overloaded assertNotNull() method, which takes Supplier<String> messageSupplier as second argument. As, this test case doesn't satisfy assertion condition, it fails and give "`AssertionFailedError:` Actual String should not be null !!! ==> expected: not ". It gives `AssertionFailedError` followed by lazily evaluates **String message** we provide to assertNotNull() method, as lambda expression.

3. `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 34** providing `assertNotNull()` asserts that actual value returned is **not null**. Thus, it pass the Junit test case because actual value returned is "" which is not null.

4. `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 42** providing `assertNotNull()` asserts that actual value returned is **not null**. Thus, it pass the Junit test case because actual value returned is **"DCBA"** which is not null.

In our upcoming lesson, we will look how we can pass all above test cases using **assertNull()** and **assertNotNull()** methods together.