assertNotEquals() method

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

We'll cover the following

assertNotEquals() method

Assertions API provide static `assertNotEquals()` method. This method helps us in validating that actual and expected values are `not` equal. This method uses `equals()` to assert the in-equality of actual and expected value.

- If the actual value is `not equal` to expected value then the test case will pass.
- If the actual value is `equal` to expected value then the test case will fail.

There are basically three useful overloaded methods for assertNotEquals:-

```
1  public static void assertNotEquals(Object expected, Object actual)
2
3  public static void assertNotEquals(Object expected, Object actual, String message)
4
5  public static void assertNotEquals(Object expected, Object actual, Supplier<String> mess
6
```

1. `assertNotEquals(Object expected, Object actual)` - It assert whether expected and actual value are `not` equal.

2. `assertNotEquals(Object expected, Object actual, String message)` - It

asserts whether expected and actual value are `not` equal. In case, if the expected value is `equal` to actual value then the test case will fail with the provided message.

3. `assertNotEquals(Object expected, Object actual, Supplier<String> messageSupplier)` - It assert whether expected and actual value are `not` equal. In case, if the expected value is `equal` to actual value 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.

assertNotEquals method

Demo

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

Step 2 - Give it a name as, StringUtils.

StringUtils.java

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

public class StringUtils {
```

```
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.

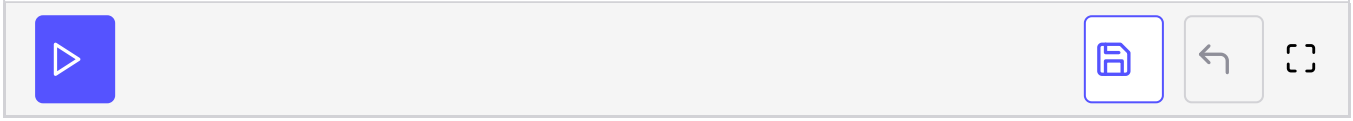
Step 3 - Create a test class by name, “StringUtilsTest2”. This test class will demonstrate all overloaded `assertNotEquals()` methods.

StringUtilsTest2.java

StringUtils.java

```
package io.educative.junit5;
```

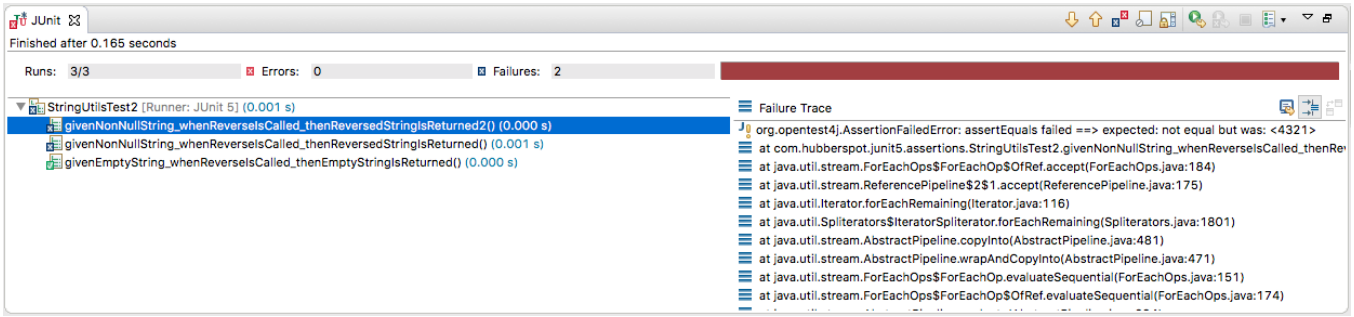
```
import static org.junit.jupiter.api.Assertions.*;

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

class StringUtilsTest2 {

	// ******** assertNotEquals Example - Start **********

	@Test
	void givenEmptyString_whenReverseIsCalled_thenEmptyStringIsReturned() {
		String actual = StringUtils.reverse((""));
		String expected = "1234";

		// assertNotEquals without message
		assertNotEquals(expected, actual);
	}

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

		String message = "assertNotEquals failed";
		// assertNotEquals with message
		assertNotEquals(expected, actual, message);
	}

	@Test
	void givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned2() {
		String actual = StringUtils.reverse(("1234"));
		String expected = "4321";

		Supplier<String> messageSupplier = () -> "assertNotEquals failed";
		// assertNotEquals with Java 8 Supplier<String>
		assertNotEquals(expected, actual, messageSupplier);
	}

	// ******** assertNotEquals Example - End **********
}
```



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

Step 4 - Run StringUtilsTest2 class as Junit Test.

Output



Explanation -

The order of execution of test cases depends on Junit 5. In StringUtilsTest1 class, there are 3 @Test methods:-

1. `givenEmptyString_whenReverseIsCalled_thenEmptyStringIsReturned()` - It tests the scenario that when “” is provided to `reverse()` method of StringUtils class, then “” is returned. Here, return value is “”. So, on **line 18** providing `assertNotEquals()` asserts that expected value and actual value returned not are equal. Thus, it passes the Junit test case because our expected value which is **“1234”** and actual value returned are not equal.

2. `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 **DCBA**. So, on **line 28** providing `assertNotEquals()` asserts that expected value and actual value returned are not equal. Thus, it fails the Junit test case because expected value is **DBCA** and actual value returned is **DCBA**.

   In this test case, we are using overloaded assertNotEquals() method, which takes `String message` as second argument. As, this test case doesn’t satisfy assertion condition, it fails and give "`AssertionFailedError:` assertNotEquals failed ==> expected: not equal but was: <DCBA>.

3. `givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned2` - It tests the scenario that when **1234** is provided to `reverse()` method of StringUtils class, then **4321** is returned. Here, return value is **4321**. So, on **line 38** providing `assertNotEquals()` asserts that expected value and actual value returned are not equal. Thus, it fails the Junit test case because expected value is **4321** and actual value returned is **4321**.

In this test case, we are using overloaded assertNotEquals() method, which takes Supplier<String> messageSupplier as second argument. As, this test case doesn't satisfy assertion condition, it fails and give "`AssertionFailedError:` assertEquals failed ==> expected: not equal but was: <4321>. It gives `AssertionFailedError` followed by lazily evaluates `String message` we provide to assertNotEquals() method, as lambda expression.

Though, actual value above returned from `reverse()` method is correct, but even if we provide the wrong expected value test case will fail.

Below code will pass all the above test cases.

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 {

	// ******** assertNotEquals Example - Start **********

	@Test
	void givenEmptyString_whenReverseIsCalled_thenEmptyStringIsReturned() {
		String actual = StringUtils.reverse((""));
		String expected = "1234";

		// assertNotEquals without message
		assertNotEquals(expected, actual);
	}

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

		String message = "assertNotEquals failed";
		// assertNotEquals with message
		assertNotEquals(expected, actual, message);
	}

	@Test
```

```
        void givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned2() {
                String actual = StringUtils.reverse(("1234"));
                String expected = "4321";

                Supplier<String> messageSupplier = () -> "assertNotEquals failed";
                // assertNotEquals with Java 8 Supplier<String>
                assertNotEquals(expected, actual, messageSupplier);
        }

        // ******** assertNotEquals Example - End **********
}
```




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