assertArrayEquals() method

This lesson demonstrates how to use assertArrayEquals method in JUnit 5 to assert test conditions.

WE'LL COVER THE FOLLOWING

assertArrayEquals() method

Assertions API provide static `assertArrayEquals()` method. This method helps us in validating that `expected` and `actual` `arrays` are equal. It has many overloaded methods to assert different types of array objects.

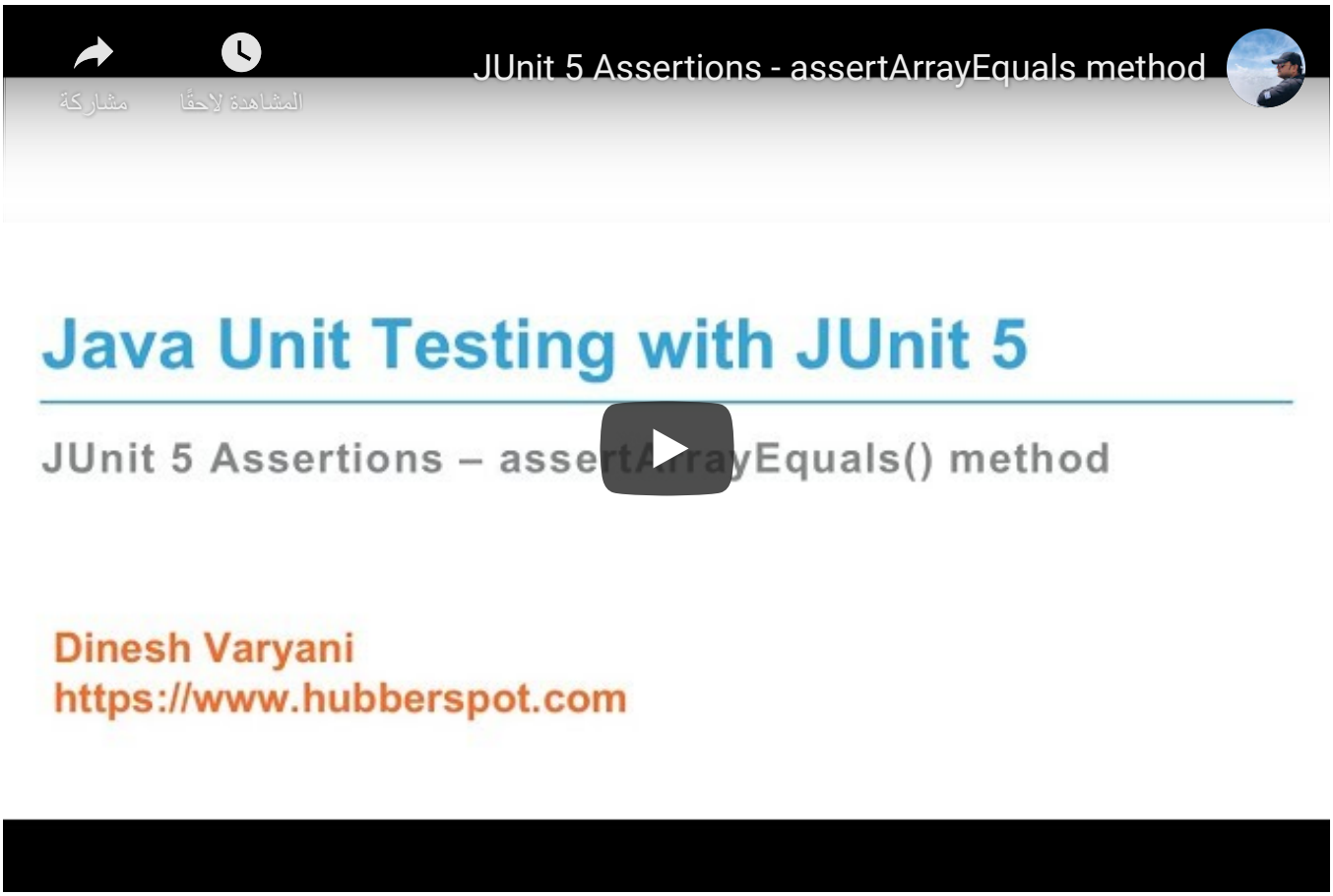
- If the actual and expected arrays are equal then the test case will pass.
- If the actual and expected arrays are not equal then the test case will fail.

There are basically three useful overloaded methods for assertArrayEquals:-

```
public static void assertArrayEquals(int[] expected, int[] actual)

public static void assertArrayEquals(int[] expected, int[] actual, String message)

public static void assertArrayEquals(int[] expected, int[] actual, Supplier<String> messageSu

// Many more same methods for different data types
```

Demo

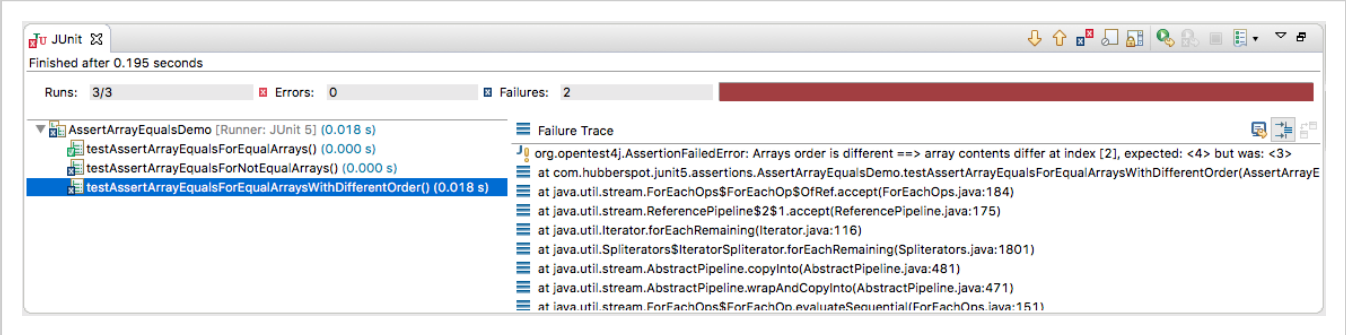
Let's look into the usage of the above methods:-



assertArrayEquals method

```
package io.educative.junit5;

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

import org.junit.jupiter.api.Test;

public class AssertArrayEqualsDemo {

	@Test
	public void testAssertArrayEqualsForEqualArrays() {
		int[] expected = {1,2,3,4};
		int[] actual = {1,2,3,4};
		assertArrayEquals(expected, actual);
	}

	@Test
	public void testAssertArrayEqualsForNotEqualArrays() {
		int[] expected = {1,2,3,4};
		int[] actual = {1,2,3};
		assertArrayEquals(expected, actual, "Arrays are not equal.");
	}

	@Test
	public void testAssertArrayEqualsForEqualArraysWithDifferentOrder() {
		int[] expected = {1,2,4,3};
		int[] actual = {1,2,3,4};
		assertArrayEquals(expected, actual, () -> "Arrays order is different");
	}
}
```

Run AssertArrayEqualsDemo class as JUnit Test.



Explanation -

In the AssertArrayEqualsDemo class, there are 3 @Test methods. These 3 methods demonstrate the working of the above 3 overloaded methods of `assertArrayEquals`:-

1. `testAssertArrayEqualsForEqualArrays()` - It asserts that actual and expected arrays are equal. Here, the expected array, {1,2,3,4} and actual array, {1,2,3,4} are passed to `assertArrayEquals()`. Thus, it passes the Junit test case because `assertArrayEquals()` finds actual and expected arrays to be equal.

2. `testAssertArrayEqualsForNotEqualArrays()` - It asserts that actual and expected arrays are equal. Here, the expected array, {1,2,3,4} and actual array, {1,2,3} are passed to `assertArrayEquals()`. Thus, it fails the Junit test case with `AssertionFailedError:` Arrays are not equal. ==> array lengths differ, expected: <4> but was: <3> because `assertArrayEquals()` finds actual and expected arrays not equal. It gives `AssertionFailedError` followed by `String message` we provide to assertArrayEquals() method.

3. `testAssertArrayEqualsForEqualArraysWithDifferentOrder()` - It asserts that actual and expected arrays are equal. Here, the expected array, {1,2,4,3} and actual array, {1,2,3,4} are passed to `assertArrayEquals()`. Thus, it fails the Junit test case with `AssertionFailedError:` Arrays order is different ==> array contents differ at index [2], expected: <4> but was: <3> because though contents of array are same they are not in same order. It gives `AssertionFailedError` followed by lazily evaluated `String message` we

provide to assertArrayEquals() method, as lambda expression.

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