assertSame() method

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

We'll cover the following

assertSame() method

Assertions API provide static `assertSame()` method. This method helps us in validating that `expected` and `actual` refer to the exact same object. JUnit uses `==` operator to perform this assert.

- If the actual and expected value refers to the same object then the test case will pass.
- If the actual and expected value does not refer to the same object then the test case will fail.

There are basically three useful overloaded methods for assertSame:-

```
1  public static void assertSame(
2
3  public static void assertSame(
4
5  public static void assertSame(
```

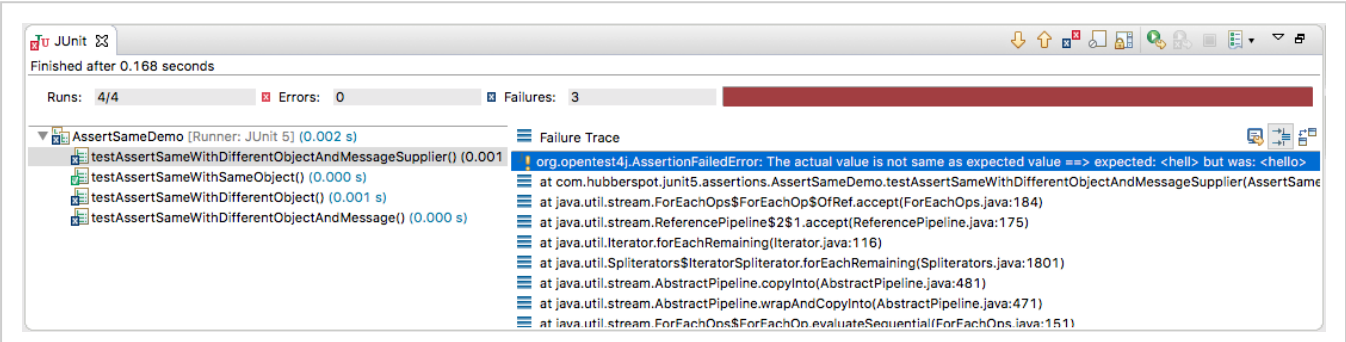
Demo

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

```
package io.educative.junit5;

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

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

public class AssertSameDemo {

	@Test
	public void testAssertSameWithSameObject() {
		String actual = "hello";
		String expected = "hello";
		assertSame(expected, actual);
	}

	@Test
	public void testAssertSameWithDifferentObject() {
		String actual = "hello";
		String expected = "hell";
		assertSame(expected, actual);
	}

	@Test
	public void testAssertSameWithDifferentObjectAndMessage() {
		String actual = "hello";
		String expected = "hell";
		assertSame(expected, actual, "The actual value is not same as expected value"
	}

	@Test
	public void testAssertSameWithDifferentObjectAndMessageSupplier() {
		String actual = "hello";
		String expected = "hell";
		assertSame(expected, actual, () -> "The actual value is not same as expected
	}
}
```

Run AssertSameDemo class as JUnit Test.



Explanation -

In the AssertSameDemo class, there are 4 @Test methods. These 4 methods demonstrate the working of the above 3 overloaded methods of `assertSame`:-

1. `testAssertSameWithSameObject()` - It asserts that actual value refers to

same expected object. Here, the expected value and actual value passed

to `assertSame()` is `hello`. Thus, it passes the Junit test case because `assertSame` finds actual and expected objects as same.

2. `testAssertSameWithDifferentObject()` - It asserts that actual value refers to same expected object. Here, the expected value and actual value passed to `assertSame()` are `hell` and `hello`. Thus, it fails the Junit test case with `AssertionFailedError:` expected: <hell> but was: <hello> because hell and hello are not same String objects.

3. `testAssertSameWithDifferentObjectAndMessage` - It asserts that actual value refers to same expected object. Here, the expected value and actual value passed to `assertSame()` are `hell` and `hello`. Thus, it fails the Junit test case with `AssertionFailedError:` The actual value is not same as expected value ==> expected: <hell> but was: <hello> because hell and hello are not same String objects. It gives `AssertionFailedError` followed by `String message` we provide to assertSame() method.

4. `testAssertTrueWithFalseConditionAndMessageSupplier` - It asserts that actual value refers to same expected object. Here, the expected value and actual value passed to `assertSame()` are `hell` and `hello`. Thus, it fails the Junit test case with `AssertionFailedError:` The actual value is not same as expected value ==> expected: <hell> but was: <hello> because hell and hello are not same String objects. It gives `AssertionFailedError` followed by lazily evaluated `String message` we provide to assertSame() method, as lambda expression.

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