**Cognizant Digital Nurture 4.0-Week(2)**

**Name: Gandu Lasya Sri**

**Email: gandulasyasri@gmail.com**

**Superset ID:6428164**

**TDD using JUnit5 and Mockito**

**Exercise 1: Mocking and Stubbing**

**Scenario:**

I had to test a service class that depends on an external API. But instead of calling the real API, I used Mockito to create a fake one (called a *mock*) and told it what output to return (called *stubbing*). This way, I could test my logic without depending on actual API results. This is especially useful when the real API is slow, unavailable, or still under development.

**Steps I Followed:**

1. **Created a Mock Object:** I used Mockito.mock() to create a fake version of the ExternalApi interface. This fake version won’t call any real API but behaves like it.
2. **Stubbed the Method:** I used when(...).thenReturn(...) to tell the mock what value it should return when its method is called. In this case, I told it to return "Mock Data" when getData() is called.
3. **Passed the Mock to the Service Class:** Instead of creating the real API object, I passed the mock to the MyService constructor.
4. **Wrote a JUnit Test:** I used assertEquals() to check that fetchData() returned "Mock Data" correctly, proving that the mock worked as expected.

**Code:**

**ExternalApi.java**

package com.lasya;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.lasya;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.lasya;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testExternalApi() {

// Creating the mock

ExternalApi mockApi = mock(ExternalApi.class);

// Stubbing the method

when(mockApi.getData()).thenReturn("Mock Data");

// Using the mock in the service

MyService service = new MyService(mockApi);

// Testing the output

String result = service.fetchData();

// Verifying the result

assertEquals("Mock Data", result);

}

}

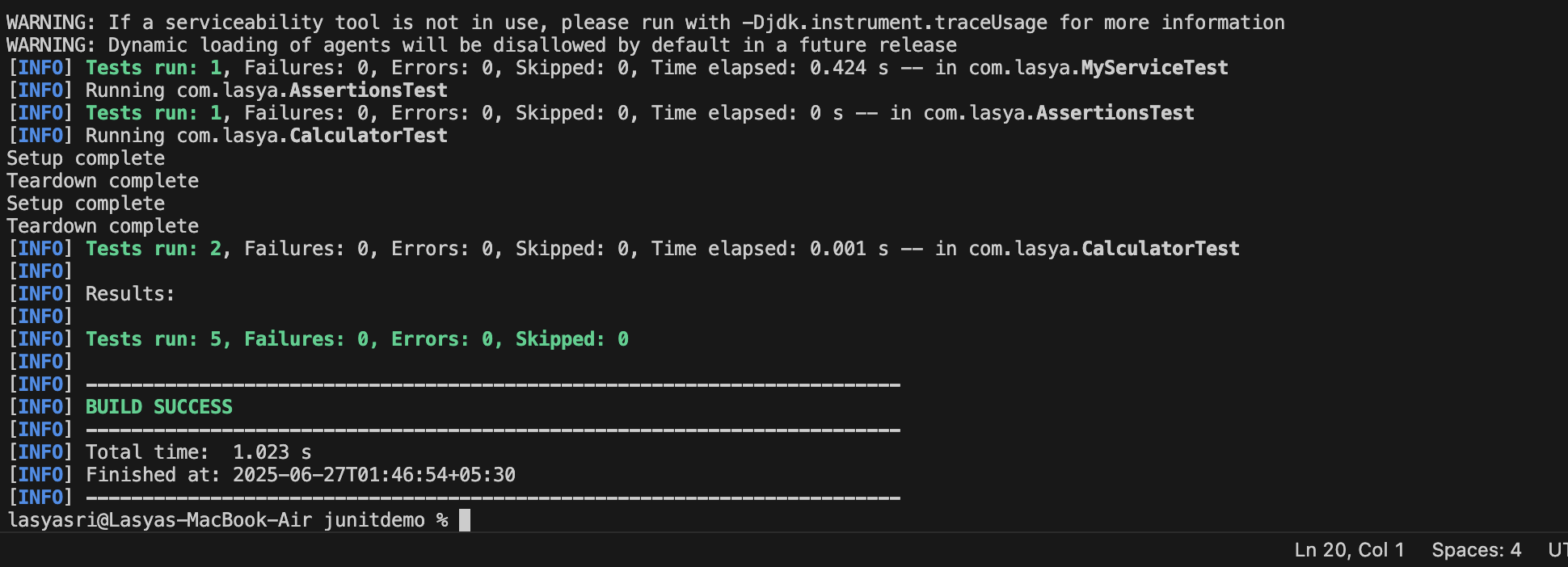
**Output:**

From the terminal using mvn test:

Running com.lasya.MyServiceTest

Tests run: 1, Failures: 0

[INFO] BUILD SUCCESS

✅ This means the mock worked and the service returned the expected result. The test passed successfully without using a real API.

**Exercise 2: Verifying Interactions**

**Scenario:**

In this exercise, I had to make sure that a specific method was actually called during execution. I used Mockito’s verify() method to check if my service really interacted with the mock object as expected. This kind of testing helps ensure that the internal behavior of a class is correct and not just the final output.

**Steps I Followed:**

1. **Created a Mock Object:** I created a fake version of the ExternalApi interface using Mockito.mock().
2. **Called the Method:** I passed the mock into MyService, and called the fetchData() method.
3. **Verified the Interaction:** I used verify(mock).getData() to check that the getData() method was actually called during the test. This confirmed that my service used the API as expected.

**Code:**

**ExternalApi.java**

package com.lasya;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.lasya;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java** (updated test for verification)

package com.lasya;

import org.junit.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Create mock object

ExternalApi mockApi = mock(ExternalApi.class);

// Inject into service

MyService service = new MyService(mockApi);

// Call the method

service.fetchData();

// Verify interaction

verify(mockApi).getData();

}

}

**Output:**

Running mvn test shows:

Running com.lasya.MyServiceTest

Tests run: 1, Failures: 0

[INFO] BUILD SUCCESS

✅ This confirms that the test passed and getData() was called exactly once as expected.

