# **WEEK 2 MOCKITO EXERCISES HANDS ON**

**SUBMITTED BY :-**

NAME – AYUSH SAHOO

SUPERSET ID – 6372243

COURSE – DIGITAL NURTURE 4.0 JAVA FSE

EMAIL – [ayush.sahoo003@gmail.com](mailto:ayush.sahoo003@gmail.com)

COLLEGE – C.V. RAMAN GLOBAL UNIVERSITY

**Exercise 1: Mocking and Stubbing**

**Scenario:**

**You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.**

**Steps:**

**1. Create a mock object for the external API.**

**2. Stub the methods to return predefined values.**

**3. Write a test case that uses the mock object.**

**Solution Code:**

**import static org.mockito.Mockito.\*;**

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

**import org.mockito.Mockito;**

**public class MyServiceTest {**

**@Test**

**public void testExternalApi() {**

**ExternalApi mockApi = Mockito.mock(ExternalApi.class);**

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

**MyService service = new MyService(mockApi);**

**String result = service.fetchData();**

**assertEquals("Mock Data", result);**

**}**

**}**

**SOLUTION:-**

Note:- Add .jar files like

byte-buddy-1.12.22.jar

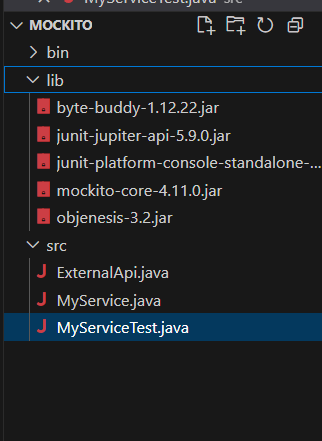
junit-jupiter-api-5.9.0.jar

junit-platform-console-standalone-1.10.0.jar

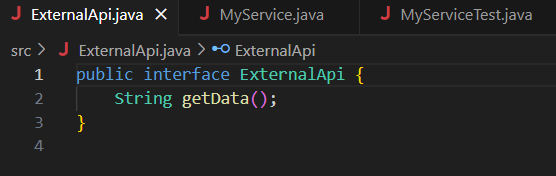
mockito-core-4.11.0.jar

objenesis-3.2.jar

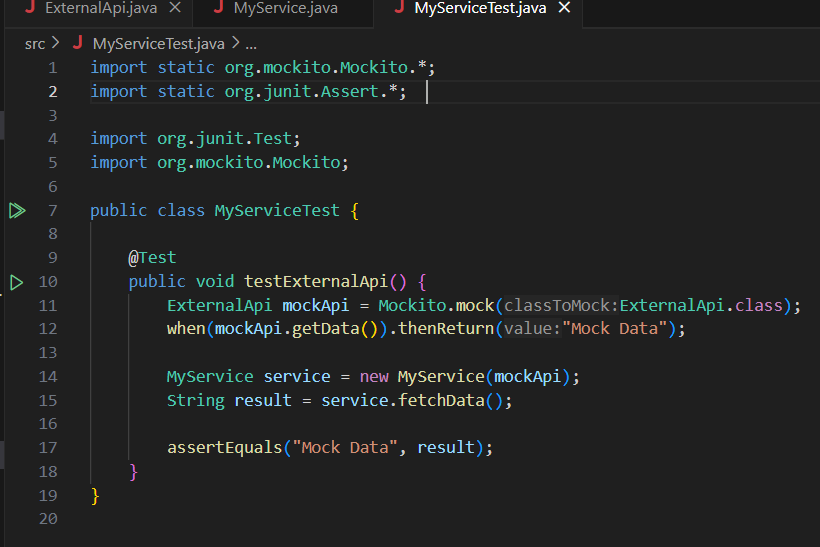
in lib folder

****

This is how my folder is looking.

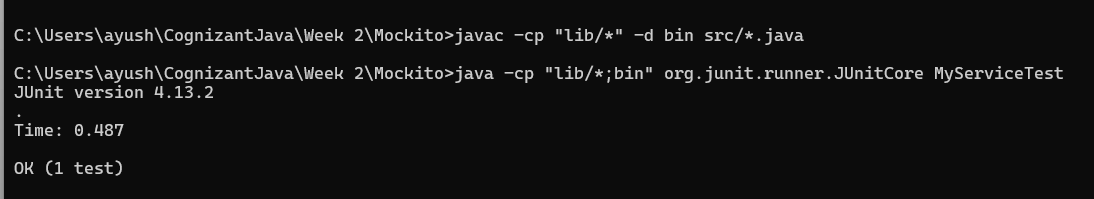






(In above picture we have mock the external API, Stub the methods to return pre-defined methods and also written a test case)

**OUTPUT:-**

****

**Exercise 2: Verifying Interactions Scenario:**

**You need to ensure that a method is called with specific arguments.**

**Steps:**

**1. Create a mock object.**

**2. Call the method with specific arguments.**

**3. Verify the interaction.**

**Solution Code:**

**import static org.mockito.Mockito.\*;**

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

**import org.mockito.Mockito;**

**public class MyServiceTest {**

**@Test**

**public void testVerifyInteraction() {**

**ExternalApi mockApi = Mockito.mock(ExternalApi.class);**

**MyService service = new MyService(mockApi);**

**service.fetchData();**

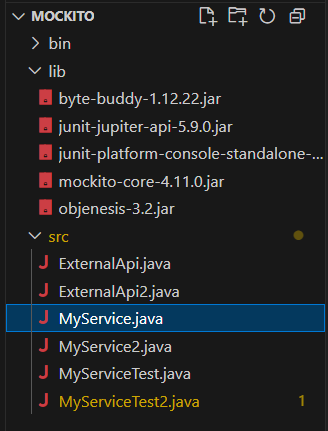
**verify(mockApi).getData();**

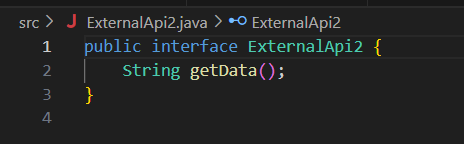
**}**

**}**

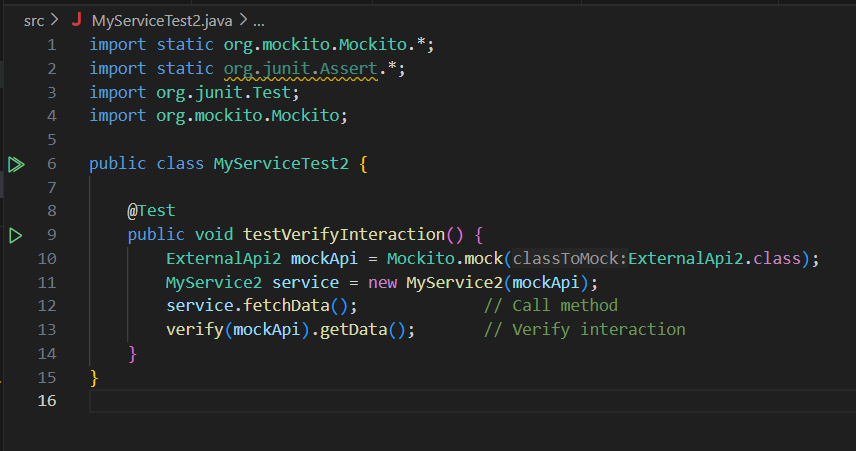
**SOLUTION:-**

**NOTE: HERE WE HAVE NAMED THE FILE AS “MyService2.java”, “MyServiceTest2.java” and “ExternalApi2.java” INSTEAD “MyService.java”, “MyServiceTest.java” and “ExternalApi.java” AS NAMES FROM EXERCISE 1 WERE COLLUDING.**

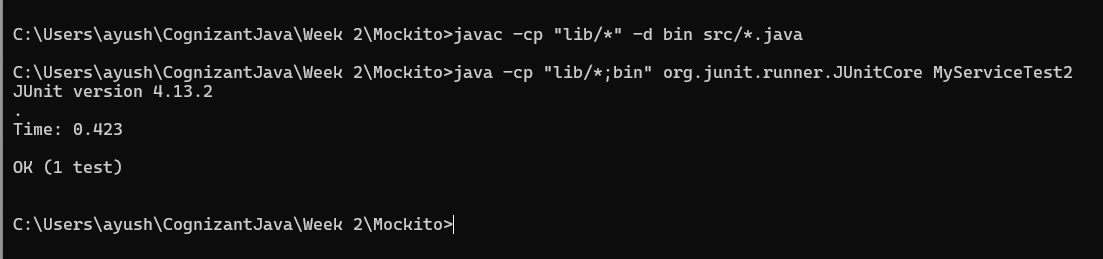
****

****

****

****

**OUTPUT:-**

****

| **Step** | **Purpose** | **Code Used** | **Where** |
| --- | --- | --- | --- |
| **1. Create mock** | **Simulate ExternalApi2** | **Mockito.mock(ExternalApi2.class)** | **MyServiceTest2.java** |
| **2. Call method** | **Execute logic to trigger interaction** | **service.fetchData();** | **MyServiceTest2.java** |
| **3. Verify** | **Confirm method was called** | **verify(mockApi).getData();** | **MyServiceTest2.java** |

**x----x----x**