Mockito Hands-On Exercises

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);
}
```

CODE:

ExternalApi.java

```
package com.example.MockDemo;
public interface ExternalApi {
String getData();
}
```

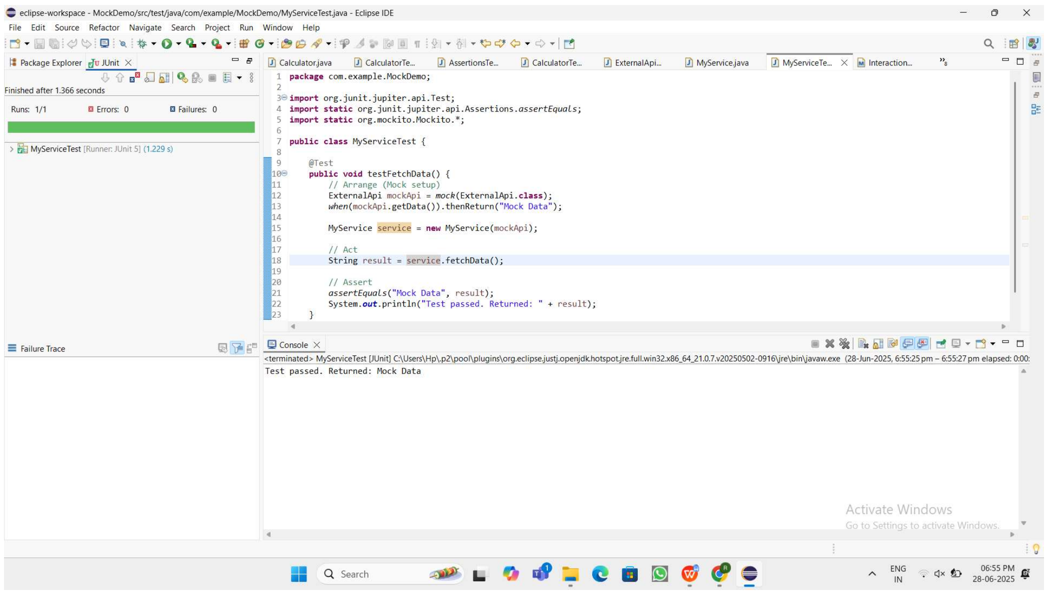
Myservice.java

```
package com.example.MockDemo;

public class MyService {
private final ExternalApi externalApi;

public MyService(ExternalApi externalApi) {
this.externalApi = externalApi;
}
public String fetchData() {
return externalApi.getData();
}
}
```

MyServiceTest.java

```
package com.example.MockDemo;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

public class MyServiceTest {

@Test
public void testFetchData() {
// Arrange (Mock setup)
ExternalApi mockApi = mock(ExternalApi.class);
when(mockApi.getData()).thenReturn("Mock Data");
MyService service = new MyService(mockApi);
String result = service.fetchData();
assertEquals("Mock Data", result);
System.out.println("Test passed. Returned: " + result);
}
}
```

OUTPUT:



eclipse-workspace - MockDemo/src/test/java/com/example/MockDemo/MyServiceTest.java - Eclipse IDE

Finished after 1.366 seconds

Runs: 1/1 Errors: 0 Failures: 0

MyServiceTest [Runner: JUnit 5] (1.229 s)

```
package com.example.MockDemo;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

public class MyServiceTest {

    @Test
    public void testFetchData() {
        // Arrange (Mock setup)
        ExternalApi mockApi = mock(ExternalApi.class);
        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);

        // Act
        String result = service.fetchData();

        // Assert
        assertEquals("Mock Data", result);
        System.out.println("Test passed. Returned: " + result);
    }
```

```
<terminated> MyServiceTest [JUnit] C:\Users\Hp\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (28-Jun-2025, 6:55:25 pm – 6:55:27 pm elapsed: 0:00:
Test passed. Returned: Mock Data
```

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();
```

CODE:

ExternalApi.java

```
package com.example.InteractionVerifier;

public interface ExternalApi {
String getData();
}
```

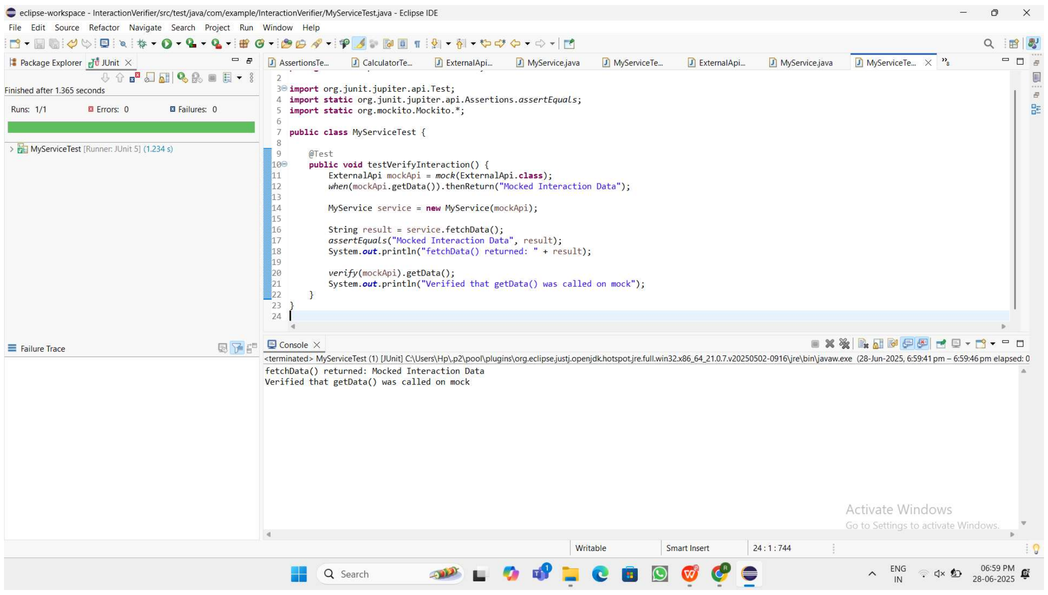
MyService.java

```
package com.example.InteractionVerifier;

public class MyService {
private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {
this.externalApi = externalApi;
}

public String fetchData() {
return externalApi.getData();
}
}
```

MyServiceTest.java

```
package com.example.InteractionVerifier;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

public class MyServiceTest {

@Test
public void testVerifyInteraction() {
ExternalApi mockApi = mock(ExternalApi.class);
when(mockApi.getData()).thenReturn("Mocked Interaction Data");

MyService service = new MyService(mockApi);

String result = service.fetchData();
assertEquals("Mocked Interaction Data", result);
System.out.println("fetchData() returned: " + result);

verify(mockApi).getData();
System.out.println("Verified that getData() was called on mock");
}
}
```

OUTPUT:


eclipse-workspace - InteractionVerifier/src/test/java/com/example/InteractionVerifier/MyServiceTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
Finished after 1.365 seconds
Runs: 1/1
Errors: 0
Failures: 0
MyServiceTest [Runner: JUnit 5] (1.234 s)
Failure Trace
AssertionsTe...
CalculatorTe...
ExternalApi...
MyService.java
MyServiceTe...
Console
Writable
Smart Insert
24 : 1 : 744

```java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

public class MyServiceTest {

    @Test
    public void testVerifyInteraction() {
        ExternalApi mockApi = mock(ExternalApi.class);
        when(mockApi.getData()).thenReturn("Mocked Interaction Data");

        MyService service = new MyService(mockApi);

        String result = service.fetchData();
        assertEquals("Mocked Interaction Data", result);
        System.out.println("fetchData() returned: " + result);

        verify(mockApi).getData();
        System.out.println("Verified that getData() was called on mock");
    }
}
```

```
<terminated> MyServiceTest (1) [JUnit] C:\Users\Hp\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (28-Jun-2025, 6:59:41 pm – 6:59:46 pm elapsed: 0
fetchData() returned: Mocked Interaction Data
Verified that getData() was called on mock
```