

Mockito Hands On Exercises

Exercise 1: Mocking and Stubbing with Mockito

Scenario:

You want to test a service (MyService) that depends on an external API (ExternalApi) by using **Mockito** to:

- Mock the API,
- Stub the method getData(),
- Verify the result in the service.

Step 1: Add Mockito + JUnit 5 to pom.xml

```
<!-- JUnit 5 -->
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter</artifactId>
  <version>5.9.3</version>
  <scope>test</scope>
</dependency>

<!-- Mockito -->
<dependency>
  <groupId>org.mockito</groupId>
  <artifactId>mockito-core</artifactId>
  <version>5.7.0</version>
  <scope>test</scope>
</dependency>
</dependencies>
```

Step 2: Create the Interface (External API)

src/main/java/com/example/ExternalApi.java

```
1 package com.example;
2
3 public interface ExternalApi {
4     String getData();
5 }
6 |
```

Step 3: Create the Service That Uses the API

src/main/java/com/example/MyService.java

```
1 package com.example;
2
3 public class MyService {
4     private ExternalApi externalApi;
5
6     public MyService(ExternalApi externalApi) {
7         this.externalApi = externalApi;
8     }
9
10    public String fetchData() {
11        return externalApi.getData();
12    }
13 }
14
```

Step 4: Create the Test Using Mockito

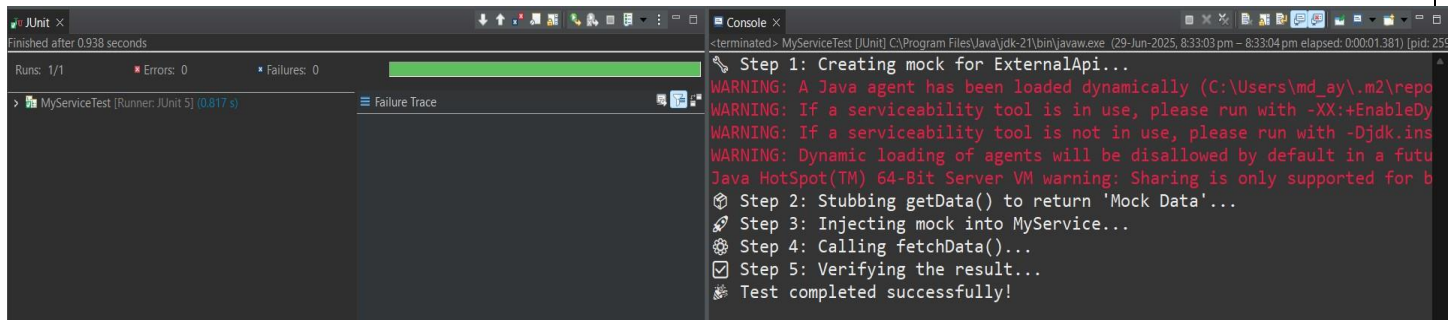
src/test/java/com/example/MyServiceTest.java

```

1 package com.example;
2
3 import org.junit.jupiter.api.Test;
4 import static org.junit.jupiter.api.Assertions.*;
5 import static org.mockito.Mockito.*;
6
7 public class MyServiceTest {
8
9     @Test
10    public void testExternalApi() {
11        System.out.println("🔧 Step 1: Creating mock for ExternalApi...");
12        ExternalApi mockApi = mock(ExternalApi.class);
13
14        System.out.println("🔗 Step 2: Stubbing getData() to return 'Mock Data'...");
15        when(mockApi.getData()).thenReturn("Mock Data");
16
17        System.out.println("💡 Step 3: Injecting mock into MyService...");
18        MyService service = new MyService(mockApi);
19
20        System.out.println("⚙️ Step 4: Calling fetchData()...");
21        String result = service.fetchData();
22
23        System.out.println("✅ Step 5: Verifying the result...");
24        assertEquals("Mock Data", result);
25
26        System.out.println("🎉 Test completed successfully!");
27    }
28 }

```

Expected Output



Exercise 2: Verifying Interactions

Scenario:

ExternalApi1.java

```
1 package com.example;
2
3 public interface ExternalApi1 {
4     String getData();
5 }
6 |
```

MyService.java

```
1 package com.example;
2
3 public class MyService1 {
4     private ExternalApi externalApi;
5
6     public MyService1(ExternalApi externalApi) {
7         this.externalApi = externalApi;
8     }
9     public String fetchData() {
10         return externalApi.getData();
11     }
12
13 }
14
```

Updated Test Code with Verification + SOPs

MyServiceTest.java

```

1 package com.example;
2
3 import org.junit.jupiter.api.Test;
4 import static org.mockito.Mockito.*;
5 import static org.junit.jupiter.api.Assertions.*;
6
7 public class MyServiceTest1 {
8
9     @Test
10    public void testVerifyInteraction() {
11        System.out.println("🔗 Step 1: Creating mock for ExternalApi...");
12        ExternalApi mockApi = mock(ExternalApi.class);
13
14        System.out.println("🔗 Step 2: Injecting mock into MyService...");
15        MyService service = new MyService(mockApi);
16
17        System.out.println("⚙️ Step 3: Calling fetchData()...");
18        service.fetchData();
19
20        System.out.println("🔍 Step 4: Verifying that getData() was called...");
21        verify(mockApi).getData();
22
23        System.out.println("✅ Test passed: getData() was called as expected.");
24    }
25 }
26

```

Expected Output

```

<terminated> MyServiceTest1 [JUnit] C:\Program Files\Java\jdk-21\bin\javaw.exe (29-Jun-2025, 8:40:00 pm - 8:40:04 pm elapsed: 0:00:03.658) [pid:
Step 1: Creating mock for ExternalApi...
WARNING: A Java agent has been loaded dynamically (C:\Users\md_ay\.m2\rep
WARNING: If a serviceability tool is in use, please run with -XX:+EnableD
WARNING: If a serviceability tool is not in use, please run with -Djdk.in
WARNING: Dynamic loading of agents will be disallowed by default in a fut
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for
Step 2: Injecting mock into MyService...
Step 3: Calling fetchData()...
Step 4: Verifying that getData() was called...
Test passed: getData() was called as expected.

```

KANMANI MURUGHAIYAN
Superset id: 6407636