Week 2 – Mockito_Mandatory_HandsOn
Skill: Mockito
Candidate: Projita Kar
Superset ID: 6407705
Type: Mandatory Hands-On

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.

- Adding Dependencies to pom.xml:-

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>Mockito_exercises</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>5.11.0</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.2</version>
```

```
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

- Creating ExternalApi Interface:-

```
package com.example;

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

- Creating MyService class:-

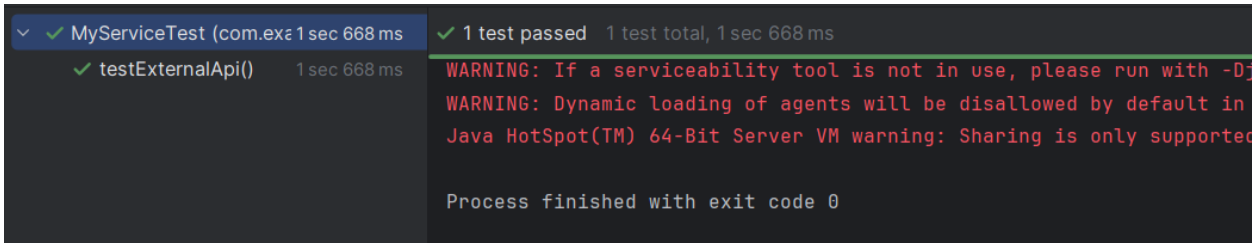
```
package com.example;

public class MyService {
    private ExternalApi api;

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

    public String fetchData() {
        return api.getData();  // calls the external API
    }
}
```

- Creating Test Class: MyServiceTest.java:-

```
package com.example;

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


public class MyServiceTest {

    @Test
    public void testExternalApi() {
        // Arrange
        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);
    }


}
```

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.

- Adding Dependencies to pom.xml:-

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>Mockito_exercises</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
```

```
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>5.11.0</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

- Creating ExternalApi Interface:-

```
package com.example;

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

- Creating MyService class:-

```
package com.example;

public class MyService {
    private ExternalApi api;

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

    public String fetchData() {
        return api.getData();  // calls the external API
    }
}
```

- Creating Test Class: MyServiceTest.java:-

```
package com.example;

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

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


public class MyServiceTest {


    @Test
    public void testVerifyInteraction() {
        // Arrange: Create a mock of ExternalApi
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        // Act: Inject mock into MyService and call the method
        MyService service = new MyService(mockApi);
        service.fetchData();

        // Assert: Verify that getData() was called once
        verify(mockApi).getData();
    }
}
```

Output:-