

## Mockito Hands-On Exercises:

### Exercise 1: Mocking and Stubbing

```
//Externalapi.java

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

```
MyService.java

public class MyService {

    private ExternalApi externalApi;

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

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

```
//Tests:

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 testExternalApi() {
        // Step 1: Create mock object
        ExternalApi mockApi = mock(ExternalApi.class);

        // Step 2: Stub the method
        when(mockApi.getdata()).thenReturn("Mock Data");

        // Step 3: Inject mock into service
        MyService service = new MyService(mockApi);

        // Call method and assert result
        String result = service.fetchData();
        assertEquals("Mock Data", result);
    }
}
```

```
//Pom.xml

<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>mockito</groupId>
    <artifactId>Mockito</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
        <!-- JUnit 5 API and Engine -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.2</version>
            <scope>test</scope>
        </dependency>

        <!-- Mockito Core -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>5.11.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.2.5</version>
                <configuration>
                    <includes>
                        <include>/**/*.Test.java</include>
                    </includes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

## Output:

The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure with 'Mockito' and 'MyServiceTest2.java'. The Console window at the bottom displays the test results, indicating that the test 'testExternalApi()' passed successfully after 2.669 seconds. The main editor area shows the source code of 'MyServiceTest2.java', which includes imports for JUnit 5 and Mockito, and a test method 'testExternalApi()' that uses Mockito to mock 'ExternalApi' and verify the behavior of 'MyService'.

```
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.assertEquals;
3 import static org.mockito.Mockito.*;
4
5 public class MyServiceTest {
6
7     @Test
8     public void testExternalApi() {
9         // Step 1: Create mock object
10        ExternalApi mockApi = mock(ExternalApi.class);
11
12        // Step 2: Stub the method
13        when(mockApi.getData()).thenReturn("Mock Data");
14
15        // Step 3: Inject mock into service
16        MyService service = new MyService(mockApi);
17
18        // Call method and assert result
19        String result = service.fetchData();
20        assertEquals("Mock Data", result);
21    }
22 }
```

## Exercise 2: Verifying Interactions:

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

public class MyServiceTest2 {

    @Test
    public void testVerifyInteraction() {
        // Step 1: Create mock
        ExternalApi mockApi = mock(ExternalApi.class);

        // Step 2: Inject into service
        MyService service = new MyService(mockApi);

        // Step 3: Call method
        service.fetchData();

        // Step 4: Verify interaction
        verify(mockApi).getdata(); // This checks that getData() was
indeed called
    }
}
```

## Output:

