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);

}

}
```

Procedure:

Step 1: Maven Dependencies (add to pom.xml)

```
<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
        <version>5.10.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
```

```
        <version>5.10.0</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Step 2. External API Interface.

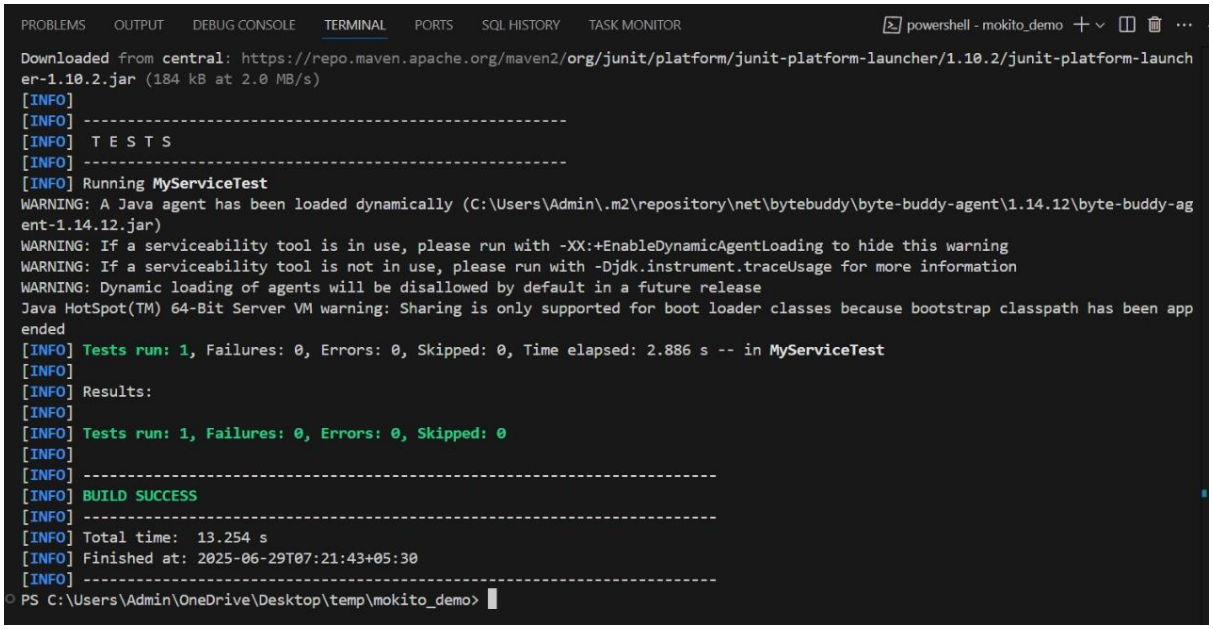
```
package com.example;
public interface ExternalApi {
    String getData();
}
```

Step 3: Unit Test Using Mockito

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

public class MyServiceTest {

    interface ExternalApi {
        String getData();
    }

    class MyService {
        private ExternalApi api;
        public MyService(ExternalApi api) {
            this.api = api;
        }
        public String fetchData() {
            return api.getData();
        }
    }

    @Test
    public void testExternalApi() {
```

```
        ExternalApi mockApi = mock(ExternalApi.class);
        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);
        String result = service.fetchData();

        assertEquals("Mock Data", result);
    }
}
```

Step 4: Out Put



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();
}
 }
```

Procedure:

Step 1: Maven Dependencies (pom.xml)

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>mockito-demo</artifactId>
  <version>1.0-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter</artifactId>
      <version>5.10.0</version>
      <scope>test</scope>
```

```
    </dependency>
    <dependency>
      <groupId>org.mockito</groupId>
      <artifactId>mockito-core</artifactId>
      <version>5.10.0</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0</version>
        <configuration>
          <includes>
            <include>**/*Test.java</include>
          </includes>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Step 2: Source Code

ExternalApi.java

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

MyService.java

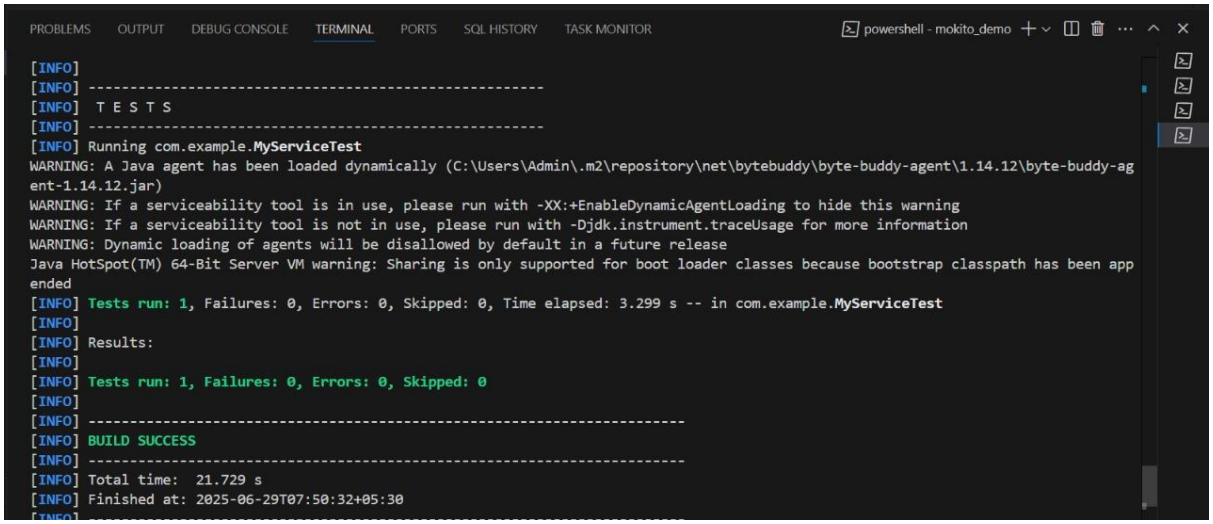
```
package com.example;
```

```
public class MyService {
    private final ExternalApi externalApi;

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

    public void fetchData() {
        externalApi.getData(); // interaction to be verified
    }
}
```

MyServiceTest.java

```
package com.example;

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

public class MyServiceTest {

    @Test
    public void testVerifyInteraction() {
        ExternalApi mockApi = mock(ExternalApi.class);
        MyService service = new MyService(mockApi);
        service.fetchData();
        verify(mockApi).getData();
    }
}
```

Step 3:Out put


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SQL HISTORY
TASK MONITOR
powershell - mokito_demo
[INFO]
[INFO] ---
[INFO] T E S T S
[INFO] ---
[INFO] Running com.example.MyServiceTest
WARNING: A Java agent has been loaded dynamically (C:\Users\Admin\.m2\repository\net\bytebuddy\byte-buddy-agent\1.14.12\byte-buddy-agent-1.14.12.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.299 s -- in com.example.MyServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 21.729 s
[INFO] Finished at: 2025-06-29T07:50:32+05:30