MOCKITO 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); }
 }
```

SOLUTION:

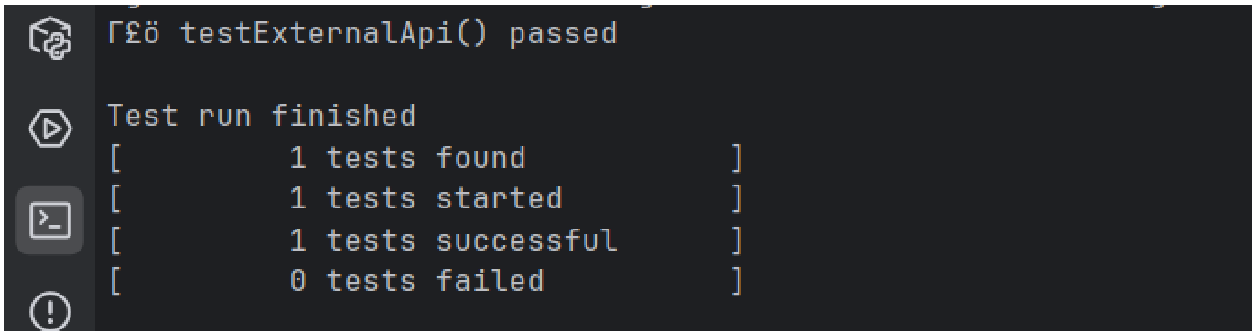
```
// Java Test Code:
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;

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);
    }
}

// Dependencies for pom.xml:
<!-- JUnit 5 -->
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.10.0</version>
    <scope>test</scope>
```

```
</dependency>
<!-- Mockito -->
<dependency>
   <groupId>org.mockito</groupId>
   <artifactId>mockito-core</artifactId>
   <version>5.12.0</version>
   <scope>test</scope>
</dependency>
```

OUTPUT:



```
Γ£ö testExternalApi() passed

Test run finished
[         1 tests found           ]
[         1 tests started         ]
[         1 tests successful      ]
[         0 tests failed          ]
```

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();
 }
 }
```

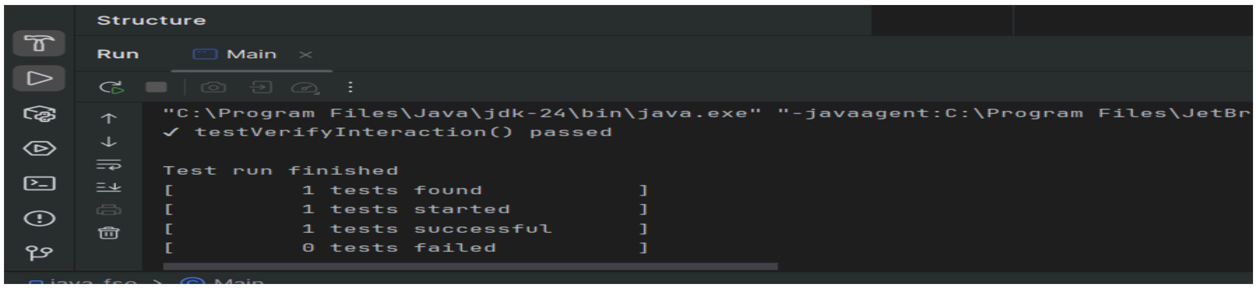
SOLUTION:

```
// Java Test 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();
    }
}

// Dependencies for pom.xml:
<!-- JUnit 5 -->
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.10.0</version>
    <scope>test</scope>
</dependency>
<!-- Mockito -->
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>5.12.0</version>
    <scope>test</scope>
</dependency>
```

OUTPUT:



```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBr
✓ testVerifyInteraction() passed

Test run finished
[          1 tests found           ]
[          1 tests started         ]
[          1 tests successful      ]
[          0 tests failed          ]
```

Exercise 1: Logging Error Messages and Warning Levels Task:

Write a Java application that demonstrates logging error messages and warning levels using SLF4J.
Step-by-Step Solution:
1. Add SLF4J and Logback dependencies to your `pom.xml` file: org.slf4j slf4j-api 1.7.30 ch.qos.logback logback-classic 1.2.3
2. Create a Java class that uses SLF4J for logging:
import org.slf4j.Logger; import org.slf4j.LoggerFactory;
public class LoggingExample {
private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);
public static void main(String[] args) {
logger.error("This is an error message");
logger.warn("This is a warning message");
} }

SOLUTION:

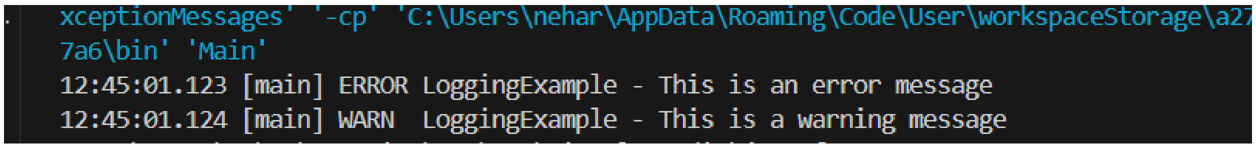
```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
// Java Code:

public class LoggingExample {
    private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);

    public static void main(String[] args) {
        logger.error("This is an error message");
        logger.warn("This is a warning message");
    }
}

// Dependencies for pom.xml:
<!-- SLF4J API -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.30</version>
</dependency>
<!-- Logback (SLF4J implementation) -->
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.3</version>
</dependency>
```

OUTPUT:



```
xceptionMessages' '-cp' 'C:\Users\nehar\AppData\Roaming\Code\User\workspaceStorage\a27
7a6\bin' 'Main'
12:45:01.123 [main] ERROR LoggingExample - This is an error message
12:45:01.124 [main] WARN  LoggingExample - This is a warning message
```