

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.

SOLUTION:

Step 1: Add the Dependency in pom.xml.

CODE:

```
<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>TDDMockito</groupId>
```

```
<artifactId>mockito</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
```

```
<properties>
```

```
<maven.compiler.source>11</maven.compiler.source>
```

```
<maven.compiler.target>11</maven.compiler.target>
```

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

```
<junit.jupiter.version>5.10.0</junit.jupiter.version>
```

```
<mockito.version>5.11.0</mockito.version>
```

```
<surefire.version>3.2.3</surefire.version> <!-- Updated surefire version -->
```

```
</properties>
```

```
<dependencies>
```

```
<!-- JUnit 5 -->
```

```
<dependency>
```

```
<groupId>org.junit.jupiter</groupId>
```

```
<artifactId>junit-jupiter</artifactId>
```

```
<version>${junit.jupiter.version}</version>
```

```
<scope>test</scope>
```

```
</dependency>
```

```
<!-- Mockito Core -->
```

```
<dependency>
```

```
<groupId>org.mockito</groupId>
```

```
<artifactId>mockito-core</artifactId>
```

```
<version>${mockito.version}</version>
<scope>test</scope>
</dependency>
```

```
<!-- Mockito JUnit 5 integration -->
<dependency>
  <groupId>org.mockito</groupId>
  <artifactId>mockito-junit-jupiter</artifactId>
  <version>${mockito.version}</version>
  <scope>test</scope>
</dependency>
```

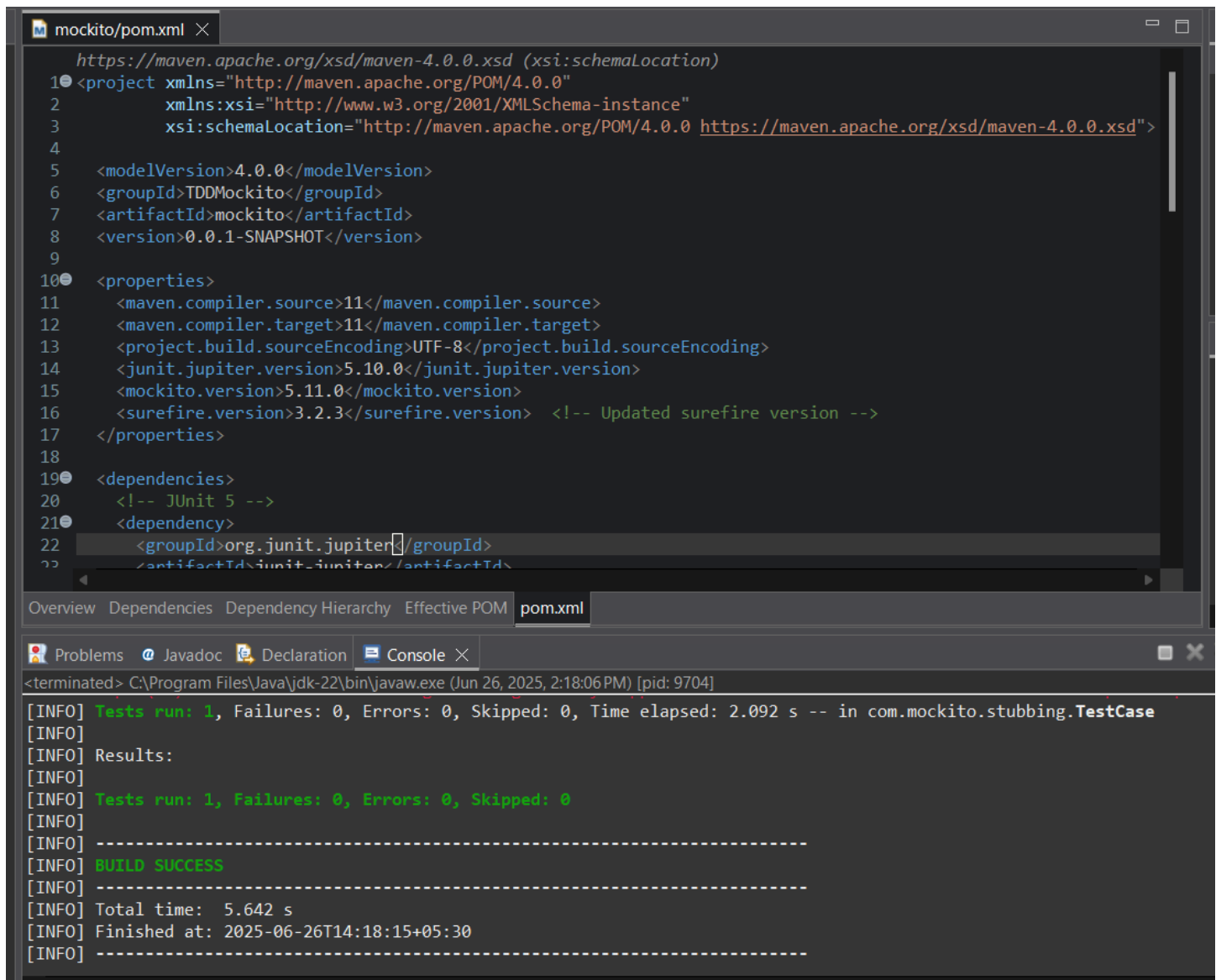
```
<!-- Add ByteBuddy dependency for Mockito's modern mocking -->
<dependency>
  <groupId>net.bytebuddy</groupId>
  <artifactId>byte-buddy</artifactId>
  <version>1.14.9</version>
  <scope>test</scope>
</dependency>
</dependencies>
```

```
<build>
  <plugins>
    <!-- Updated Maven Surefire Plugin configuration -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>${surefire.version}</version>
      <configuration>
        <argLine>
          --add-opens java.base/java.lang=ALL-UNNAMED
          --add-opens java.base/java.util=ALL-UNNAMED
        </argLine>
      </configuration>
    </plugin>
```

```
<!-- Add Maven Compiler Plugin for explicit configuration -->
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.11.0</version>
  <configuration>
    <release>11</release>
  </configuration>
</plugin>
```

```
</plugins>
</build>
</project>
```

OUTPUT:



The screenshot shows an IDE window with two tabs: 'mockito/pom.xml' and 'Console'. The 'pom.xml' tab is active, displaying the following XML content:

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4
5   <modelVersion>4.0.0</modelVersion>
6   <groupId>TDDMockito</groupId>
7   <artifactId>mockito</artifactId>
8   <version>0.0.1-SNAPSHOT</version>
9
10  <properties>
11    <maven.compiler.source>11</maven.compiler.source>
12    <maven.compiler.target>11</maven.compiler.target>
13    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
14    <junit.jupiter.version>5.10.0</junit.jupiter.version>
15    <mockito.version>5.11.0</mockito.version>
16    <surefire.version>3.2.3</surefire.version> <!-- Updated surefire version -->
17  </properties>
18
19  <dependencies>
20    <!-- JUnit 5 -->
21    <dependency>
22      <groupId>org.junit.jupiter</groupId>
23      <artifactId>junit-jupiter</artifactId>
```

The 'Console' tab is also active, showing the output of a Maven build. The output indicates that the build was successful, with 1 test run, 0 failures, 0 errors, and 0 skipped tests. The total time taken was 5.642 seconds, and the build finished at 2025-06-26T14:18:15+05:30.

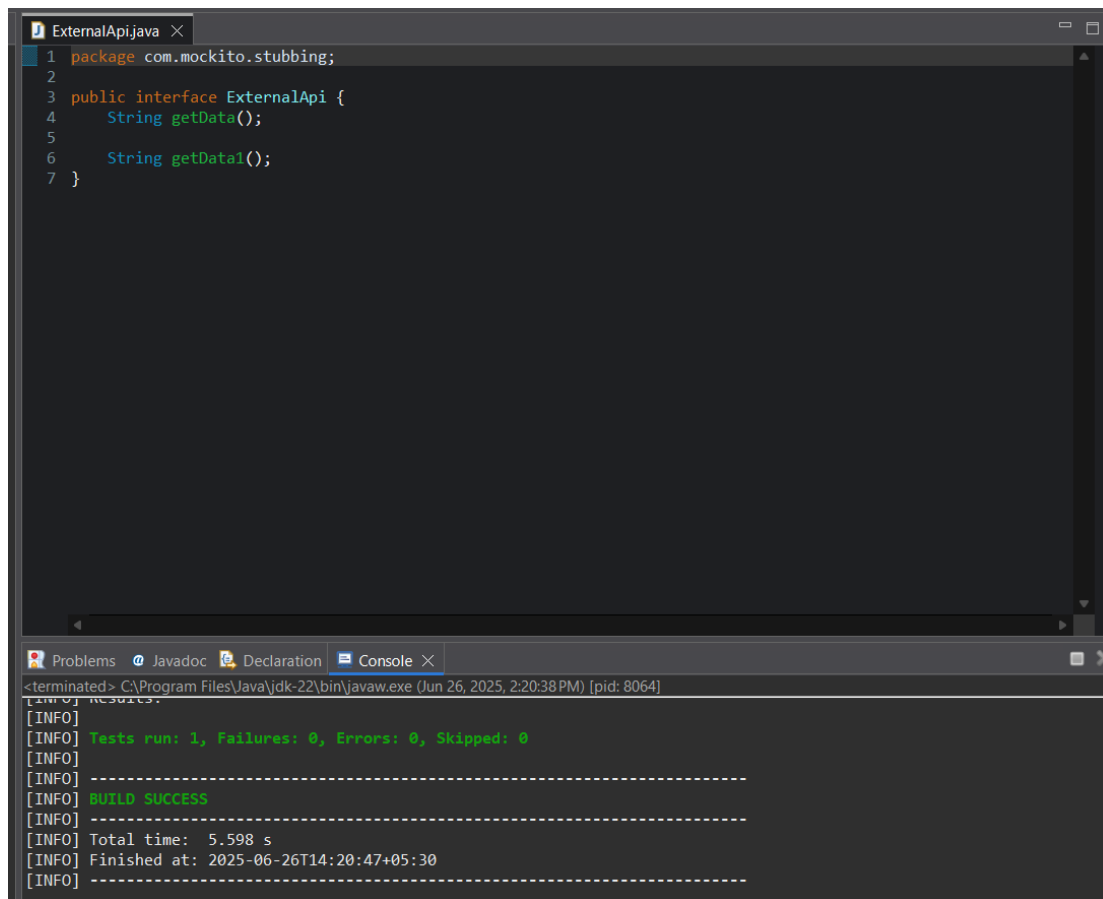
Step 2: Create a mock object for the external API.

CODE (for ExternalApi.java):

```
package com.mockito.stubbing;

public interface ExternalApi {
    String getData();
    String getData1();
}
```

OUTPUT(for ExternalApi.java):



The screenshot shows an IDE window titled 'ExternalApi.java'. The code in the editor is as follows:

```
1 package com.mockito.stubbing;
2
3 public interface ExternalApi {
4     String getData();
5
6     String getData1();
7 }
```

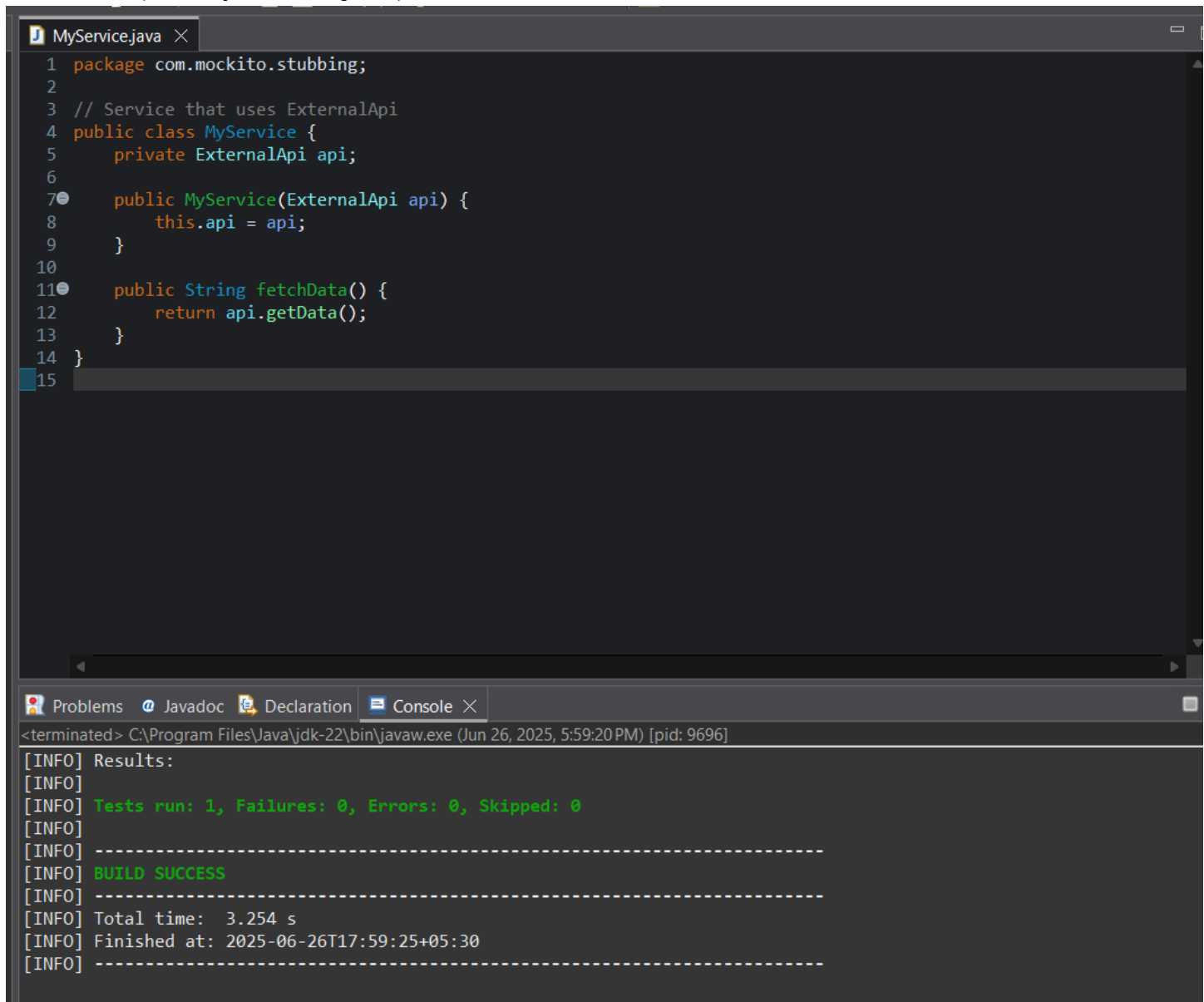
Below the editor, the 'Console' tab is active, displaying the following output:

```
<terminated> C:\Program Files\Java\jdk-22\bin\javaw.exe (Jun 26, 2025, 2:20:38 PM) [pid: 8064]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.598 s
[INFO] Finished at: 2025-06-26T14:20:47+05:30
[INFO] -----
```

CODE(for MyService.java):

```
package com.mockito.stubbing;
// Service that uses ExternalApi
public class MyService {
    private ExternalApi api;
    public MyService(ExternalApi api) {
        this.api = api;
    }
    public String fetchData() {
        return api.getData();
    }
}
```

OUTPUT(for MyService.java):



```
MyService.java X
1 package com.mockito.stubbing;
2
3 // Service that uses ExternalApi
4 public class MyService {
5     private ExternalApi api;
6
7     public MyService(ExternalApi api) {
8         this.api = api;
9     }
10
11     public String fetchData() {
12         return api.getData();
13     }
14 }
15
```

```
Problems Javadoc Declaration Console X
<terminated> C:\Program Files\Java\jdk-22\bin\javaw.exe (Jun 26, 2025, 5:59:20 PM) [pid: 9696]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.254 s
[INFO] Finished at: 2025-06-26T17:59:25+05:30
[INFO] -----
```

Step 3&4: Stub the methods to return predefined values. Write a test case that uses the mock object.

CODE(for TestCase.java):

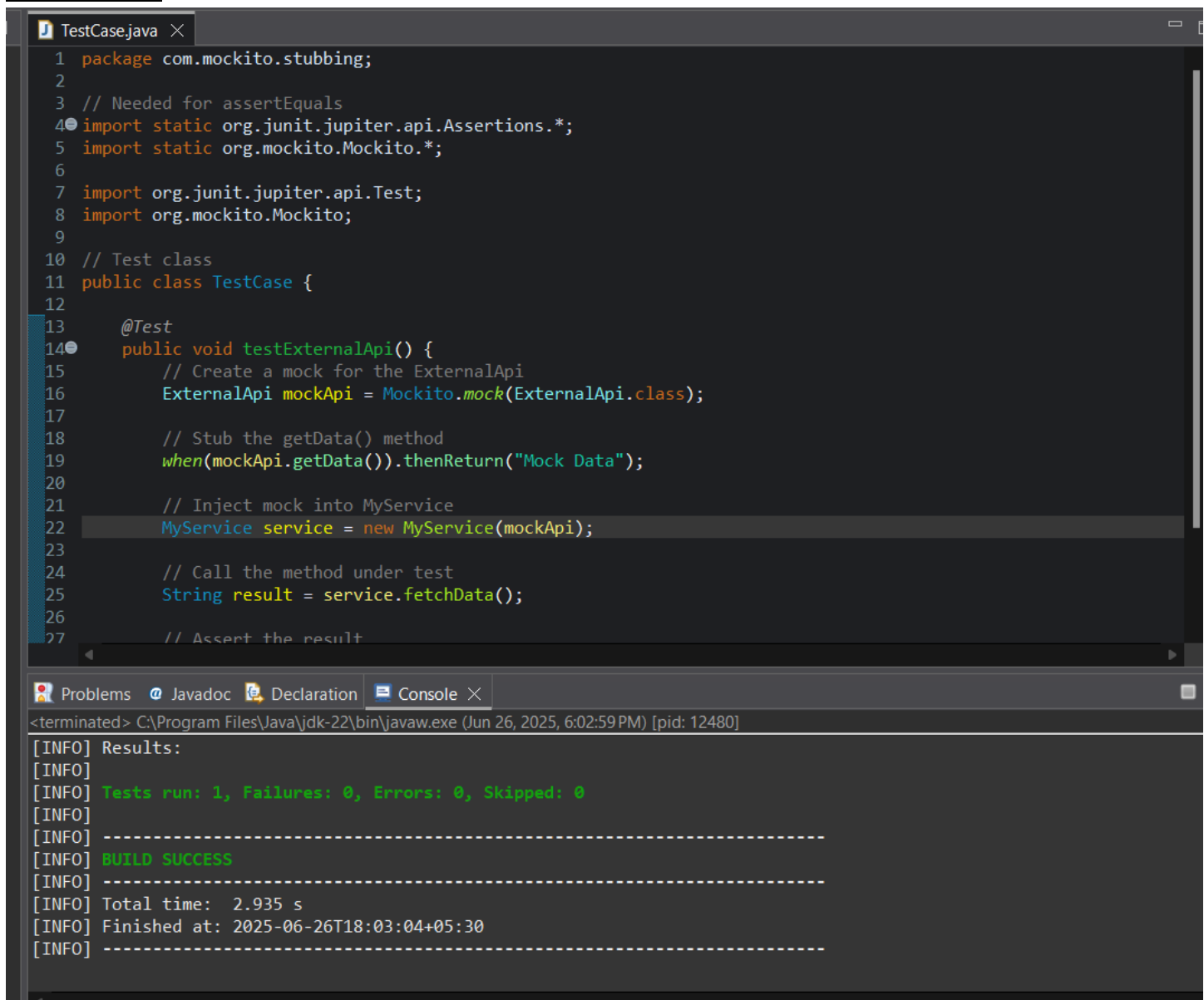
```
package com.mockito.stubbing;
// Needed for assertEquals
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
// Test class
public class TestCase {
```

@Test

```
public void testExternalApi() {  
    // Create a mock for the ExternalApi  
    ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
    // Stub the getData() method  
    when(mockApi.getData()).thenReturn("Mock Data");  
    // Inject mock into MyService  
    MyService service = new MyService(mockApi);  
    // Call the method under test  
    String result = service.fetchData();  
    // Assert the result  
    assertEquals("Mock Data", result);  
}
```

OUTPUT(for TestCase.java):

Maven test:



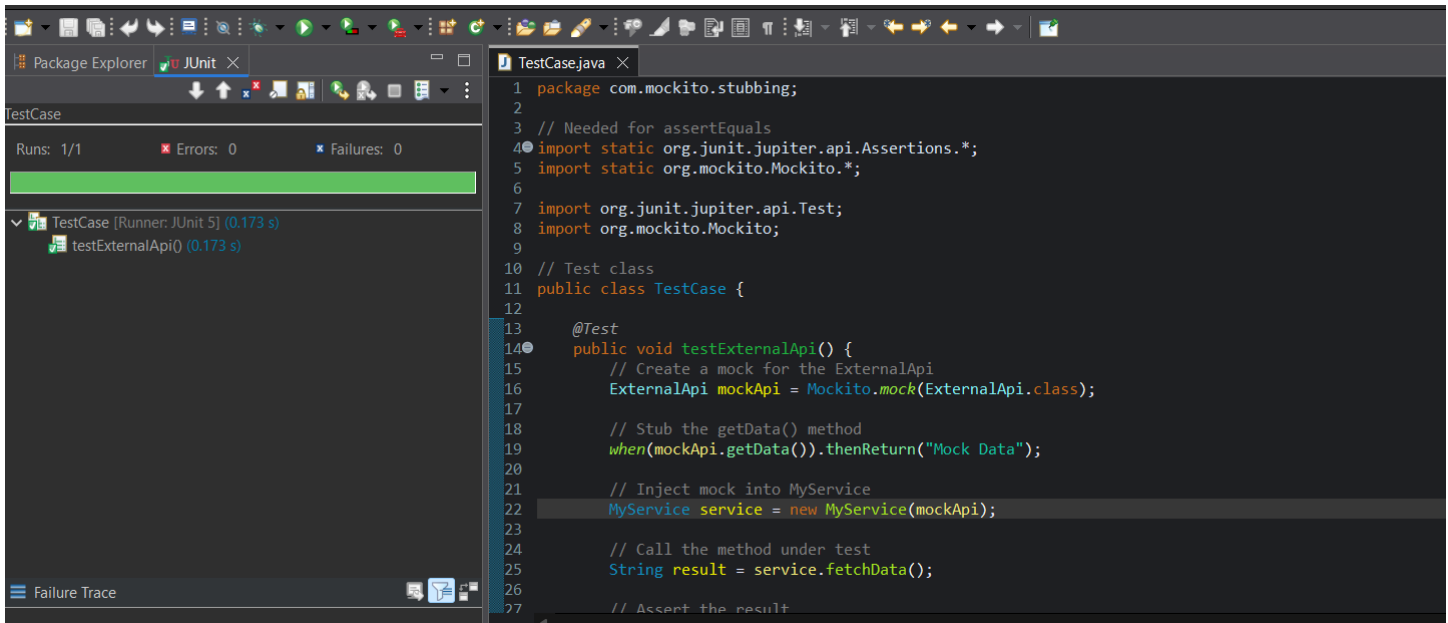
The screenshot displays an IDE window with the file `TestCase.java` open. The code is a JUnit test class that uses Mockito to mock the `ExternalApi` and verify the behavior of `MyService`. The test method `testExternalApi()` creates a mock, stubs the `getData()` method to return "Mock Data", injects the mock into a new `MyService` instance, calls `fetchData()`, and asserts that the result is "Mock Data".

Below the code editor, the Maven test output is visible in the console. It shows that the test was successful, with 1 test run, 0 failures, 0 errors, and 0 skipped tests. The build time was 2.935 seconds, and it finished at 2025-06-26T18:03:04+05:30.

```
1 package com.mockito.stubbing;  
2  
3 // Needed for assertEquals  
4 import static org.junit.jupiter.api.Assertions.*;  
5 import static org.mockito.Mockito.*;  
6  
7 import org.junit.jupiter.api.Test;  
8 import org.mockito.Mockito;  
9  
10 // Test class  
11 public class TestCase {  
12  
13     @Test  
14     public void testExternalApi() {  
15         // Create a mock for the ExternalApi  
16         ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
17  
18         // Stub the getData() method  
19         when(mockApi.getData()).thenReturn("Mock Data");  
20  
21         // Inject mock into MyService  
22         MyService service = new MyService(mockApi);  
23  
24         // Call the method under test  
25         String result = service.fetchData();  
26  
27         // Assert the result  
28     }  
29 }
```

<terminated> C:\Program Files\Java\jdk-22\bin\javaw.exe (Jun 26, 2025, 6:02:59 PM) [pid: 12480]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.935 s
[INFO] Finished at: 2025-06-26T18:03:04+05:30
[INFO] -----

JUnit test:



Exercise 6: Verifying Interaction Order

Scenario: You need to ensure that methods are called in a specific order.

SOLUTION:

Step 1: Add the Dependency in pom.xml.

CODE:

```
<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>Verify</groupId>
```

```
<artifactId>VerifyingInteractions</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
```

```
<dependencies>
```

```
<!-- JUnit 5 (Jupiter) for testing -->
```

```
<dependency>
```

```
<groupId>org.junit.jupiter</groupId>
```

```
<artifactId>junit-jupiter</artifactId>
```

```
<version>5.10.0</version>
```

```
<scope>test</scope>
```

```
</dependency>
```

```

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

<build>
  <plugins>
    <!-- Maven Surefire Plugin to run JUnit 5 tests -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.2.5</version>
      <configuration>
        <useModulePath>>false</useModulePath>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>

```

OUTPUT:

The screenshot shows an IDE window titled 'VerifyingInteractions/pom.xml'. The editor displays the XML content of the POM file, which includes dependencies for 'Verify' (0.0.1-SNAPSHOT), 'junit-jupiter' (5.10.0), and 'mockito-core' (5.11.0). The 'pom.xml' tab is selected in the bottom toolbar.

Below the editor, the 'Console' tab is active, showing the build output. The output indicates that the build was successful, with 1 test run, 0 failures, 0 errors, and 0 skipped tests. The total time taken was 2.947 seconds, and the build finished at 2025-06-26T19:13:13+05:30.

```

[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.947 s
[INFO] Finished at: 2025-06-26T19:13:13+05:30
[INFO] -----

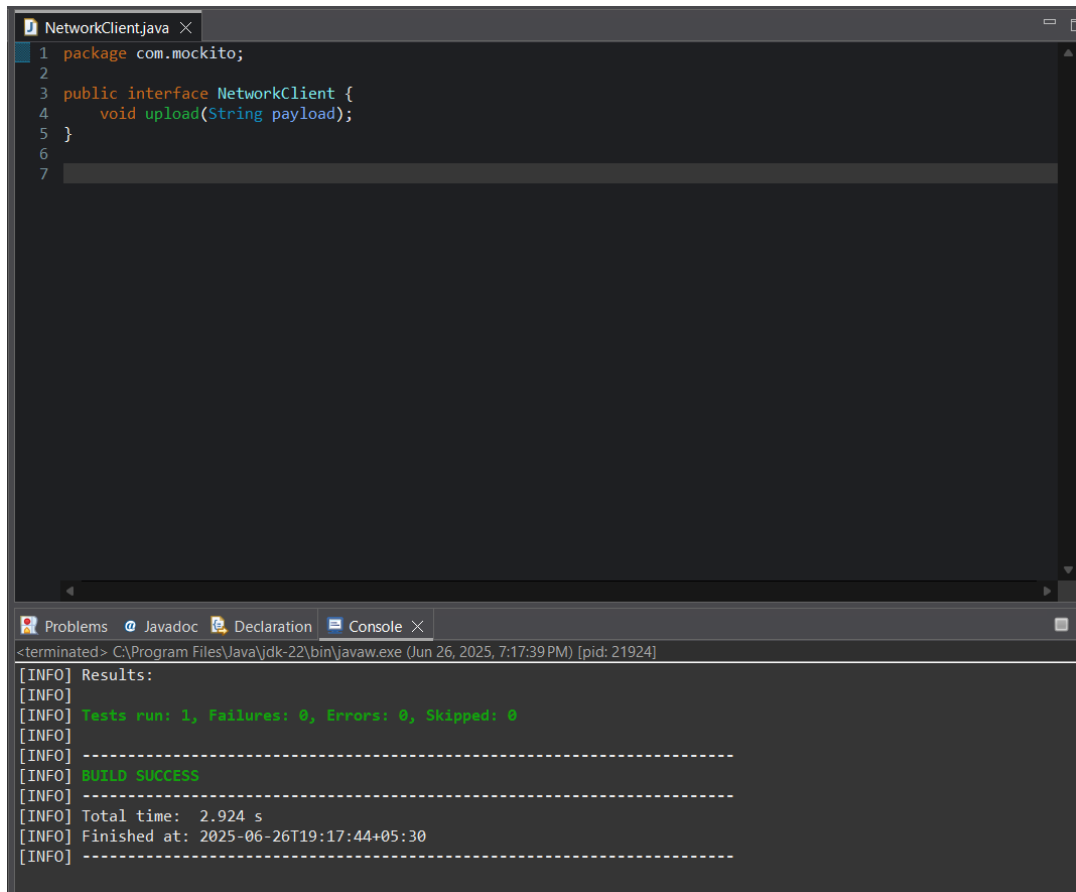
```


Step 2: Create a mock object.

CODE(for NetworkClient.java):

```
package com.mockito;  
  
public interface NetworkClient {  
    void upload(String payload);  
}
```

OUTPUT(for NetworkClient.java):



The screenshot shows an IDE window titled 'NetworkClient.java' with the following code:

```
1 package com.mockito;  
2  
3 public interface NetworkClient {  
4     void upload(String payload);  
5 }  
6  
7
```

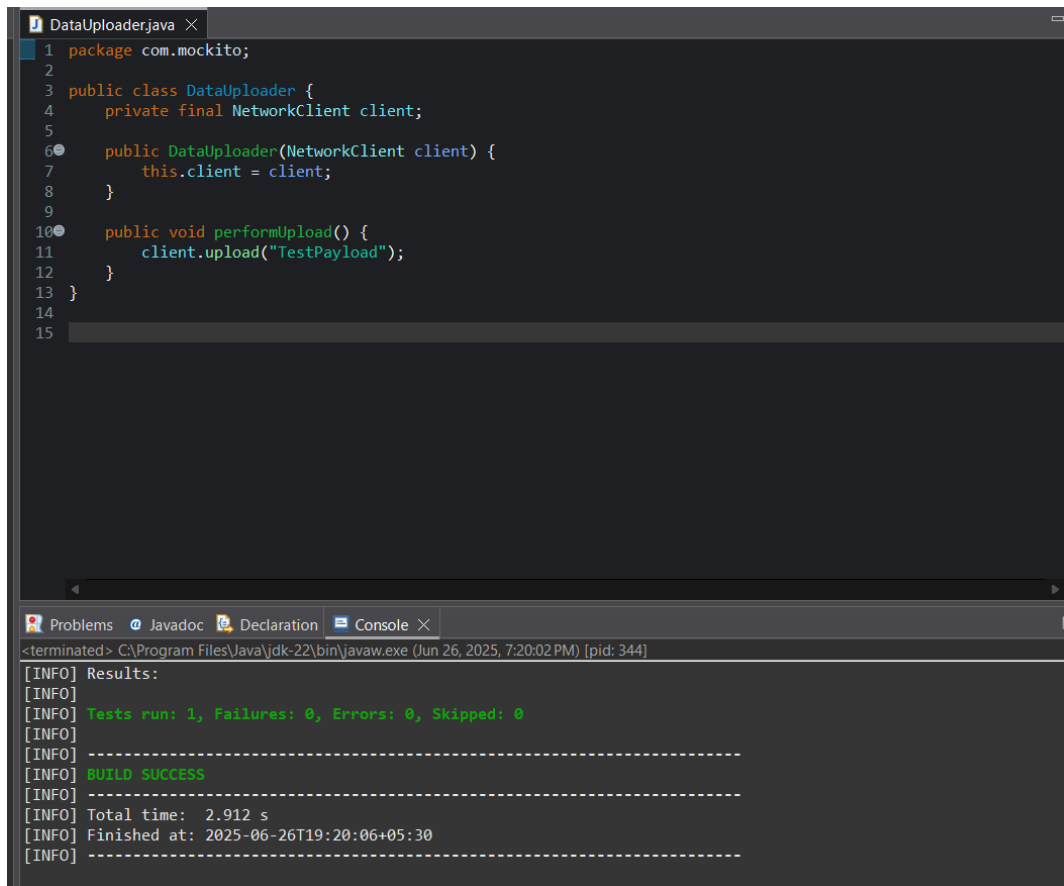
Below the code editor, the 'Console' tab is active, displaying the following output:

```
<terminated> C:\Program Files\Java\jdk-22\bin\javaw.exe (Jun 26, 2025, 7:17:39 PM) [pid: 21924]  
[INFO] Results:  
[INFO]  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
[INFO]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 2.924 s  
[INFO] Finished at: 2025-06-26T19:17:44+05:30  
[INFO] -----
```

CODE(for DataUploader.java):

```
package com.mockito;  
  
public class DataUploader {  
    private final NetworkClient client;  
    public DataUploader(NetworkClient client) {  
        this.client = client;  
    }  
    public void performUpload() {  
        client.upload("TestPayload");  
    }  
}
```

OUTPUT(for DataUploader.java):



The screenshot shows an IDE window with a file named 'DataUploader.java'. The code in the file is as follows:

```
1 package com.mockito;
2
3 public class DataUploader {
4     private final NetworkClient client;
5
6     public DataUploader(NetworkClient client) {
7         this.client = client;
8     }
9
10    public void performUpload() {
11        client.upload("TestPayload");
12    }
13 }
14
15
```

Below the code editor, the 'Console' tab is active, displaying the following output:

```
<terminated> C:\Program Files\Java\jdk-22\bin\javaw.exe (Jun 26, 2025, 7:20:02 PM) [pid: 344]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.912 s
[INFO] Finished at: 2025-06-26T19:20:06+05:30
[INFO] -----
```

Step 3&4: Call the methods in a specific order. Verify the interaction order.

CODE(for TestDataUploader.java):

```
package com.mockito;
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
public class TestDataUploader {
    @Test
    public void testUploadCalledWithCorrectArgument() {
        // Step 1: Create mock
        NetworkClient mockClient = Mockito.mock(NetworkClient.class);
        // Step 2: Call method with specific argument
        DataUploader uploader = new DataUploader(mockClient);
        uploader.performUpload();
        // Step 3: Verify interaction
        verify(mockClient).upload("TestPayload");
    }
}
```

OUTPUT:

Maven Test:

```
TestDataUploader.java ×
1 package com.mockito;
2
3 import static org.mockito.Mockito.*;
4 import org.junit.jupiter.api.Test;
5 import org.mockito.Mockito;
6
7 public class TestDataUploader {
8     @Test
9     public void testUploadCalledWithCorrectArgument() {
10         // Step 1: Create mock
11         NetworkClient mockClient = Mockito.mock(NetworkClient.class);
12
13         // Step 2: Call method with specific argument
14         DataUploader uploader = new DataUploader(mockClient);
15         uploader.performUpload();
16
17         // Step 3: Verify interaction
18         verify(mockClient).upload("TestPayload");
19     }
20 }
21
22
```

Problems Javadoc Declaration Console ×

<terminated> C:\Program Files\Java\jdk-22\bin\javaw.exe (Jun 26, 2025, 7:23:52 PM) [pid: 6172]

[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.900 s
[INFO] Finished at: 2025-06-26T19:23:56+05:30
[INFO] -----

Junit Test:

```
Package Explorer JUnit ×
Finished after 1.157 seconds
Runs: 1/1 Errors: 0 Failures: 0
> TestDataUploader [Runner: JUnit 5] (1.062 s)
Failure Trace
```

```
TestDataUploader.java ×
1 package com.mockito;
2
3 import static org.mockito.Mockito.*;
4 import org.junit.jupiter.api.Test;
5 import org.mockito.Mockito;
6
7 public class TestDataUploader {
8     @Test
9     public void testUploadCalledWithCorrectArgument() {
10         // Step 1: Create mock
11         NetworkClient mockClient = Mockito.mock(NetworkClient.class);
12
13         // Step 2: Call method with specific argument
14         DataUploader uploader = new DataUploader(mockClient);
15         uploader.performUpload();
16
17         // Step 3: Verify interaction
18         verify(mockClient).upload("TestPayload");
19     }
20 }
21
22
```

NAME-RITIKA KUMARI

SUPERSET ID-6392654