Exercise 1: Mocking and Stubbing

Code:

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-demoo</artifactId>

 <version>0.0.1-SNAPSHOT</version>

 <name>mockito-demoo</name>

 <!-- FIXME change it to the project's website -->

 <url>http://www.example.com</url>

 <properties>

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

  <maven.compiler.release>17</maven.compiler.release>

 </properties>

 <dependencyManagement>

  <dependencies>

   <dependency>

    <groupId>org.junit</groupId>

    <artifactId>junit-bom</artifactId>

    <version>5.11.0</version>

    <type>pom</type>

    <scope>import</scope>

   </dependency>

  </dependencies>

 </dependencyManagement>

 <dependencies>

 <!-- JUnit Jupiter API -->

 <dependency>

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

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

```
  <scope>test</scope>
 </dependency>
 <!-- JUnit Jupiter Params (optional) -->
 <dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-params</artifactId>
  <scope>test</scope>
 </dependency>

 <!-- JUnit Engine to run tests -->
 <dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-engine</artifactId>
  <scope>test</scope>
 </dependency>

 <!-- ✅ Mockito -->
 <dependency>
  <groupId>org.mockito</groupId>
  <artifactId>mockito-core</artifactId>
  <version>4.11.0</version>
  <scope>test</scope>
 </dependency>
 </dependencies>

 <build>
  <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
   <plugins>
    <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Lifecycle -->
    <plugin>
     <artifactId>maven-clean-plugin</artifactId>
     <version>3.4.0</version>
    </plugin>
```

```
    <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_jar_packaging -->
    <plugin>
      <artifactId>maven-resources-plugin</artifactId>
      <version>3.3.1</version>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.13.0</version>
    </plugin>
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.3.0</version>
    </plugin>
    <plugin>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.4.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-install-plugin</artifactId>
      <version>3.1.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-deploy-plugin</artifactId>
      <version>3.1.2</version>
    </plugin>
    <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle -->
    <plugin>
      <artifactId>maven-site-plugin</artifactId>
      <version>3.12.1</version>
    </plugin>
    <plugin>
      <artifactId>maven-project-info-reports-plugin</artifactId>
      <version>3.6.1</version>
```

```
      </plugin>
    </plugins>
  </pluginManagement>
 </build>
</project>
```

ExternalApi.java:

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

MyService.java:

```
package com.example.mockito_demoo;
public class MyService {
  private ExternalApi api;
  public MyService(ExternalApi api) {
    this.api = api;
  }
  public String fetchData() {
    return api.getData(); // Relies on external API
  }
}
```

MyServiceTest.java:

```
package com.example.mockito_demoo;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.when;

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);
    }
}
```

Output: