**JUnit Basic Testing Exercises**

**Exercise 1: Setting Up JUnit**

**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.settingupjunit</groupId>

  <artifactId>demo</artifactId>

  <version>1.0</version>

  <name>demo</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.source>1.8</maven.compiler.source>

    <maven.compiler.target>1.8</maven.compiler.target>

  </properties>

  <!-- <dependencies>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.11</version>

      <scope>test</scope>

    </dependency>

  </dependencies> -->

<dependencies>

    <dependency>

        <groupId>junit</groupId>

        <artifactId>junit</artifactId>

        <version>4.13.2</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>3.1.0</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>3.0.2</version>

        </plugin>

        <plugin>

          <artifactId>maven-compiler-plugin</artifactId>

          <version>3.8.0</version>

        </plugin>

        <plugin>

          <artifactId>maven-surefire-plugin</artifactId>

          <version>2.22.1</version>

        </plugin>

        <plugin>

          <artifactId>maven-jar-plugin</artifactId>

          <version>3.0.2</version>

        </plugin>

        <plugin>

          <artifactId>maven-install-plugin</artifactId>

          <version>2.5.2</version>

        </plugin>

        <plugin>

          <artifactId>maven-deploy-plugin</artifactId>

          <version>2.8.2</version>

        </plugin>

        <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site\_Lifecycle -->

        <plugin>

          <artifactId>maven-site-plugin</artifactId>

          <version>3.7.1</version>

        </plugin>

        <plugin>

          <artifactId>maven-project-info-reports-plugin</artifactId>

          <version>3.0.0</version>

        </plugin>

      </plugins>

    </pluginManagement>

  </build>

</project>

**//Calculator.java**

package com.example.settingupjunit;

public class Calculator {

    public int add(int a, int b) {

        return a + b;

    }

}

**//CalculatorTest.java**

package com.example.settingupjunit;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

    @Test

    public void testAdd() {

        Calculator calc = new Calculator();

        int result = calc.add(3, 4);

        assertEquals(7, result);

    }

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 3: Assertions in JUnit**

**Code:**

**//AssertionsTest.java**

package com.example.junitassertions;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

    @Test

    public void testAssertions() {

        // Assert equals

        assertEquals(5, 2 + 3);

        // Assert true

        assertTrue(5 > 3);

        // Assert false

        assertFalse(5 < 3);

        // Assert null

        Object obj = null;

        assertNull(obj);

        // Assert not null

        assertNotNull(new Object());

    }

}

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit**

**Code:**

package com.example.aaapattern;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

class Calculator {

public int add(int a, int b) { return a + b; }

public int subtract(int a, int b) { return a - b; }

}

public class CalculatorTestWithFixture {

private Calculator calculator;

@Before

public void setUp() {

calculator = new Calculator(); // Arrange

System.out.println("Setup complete");

}

@After

public void tearDown() {

System.out.println("Teardown complete");

calculator = null; // Clean up

}

@Test

public void testAddition() {

int result = calculator.add(10, 5);

assertEquals(15, result);

}

@Test

public void testSubtraction() {

int result = calculator.subtract(10, 5);

assertEquals(5, result);

}

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**Code:**

//ExternalApi.java

package com.example.mockingandstubbing;

public interface ExternalApi {

    String getData();

}

//MyService.java

package com.example.mockingandstubbing;

public class MyService {

    private ExternalApi externalApi;

    public MyService(ExternalApi externalApi) {

        this.externalApi = externalApi;

    }

    public String fetchData() {

        return externalApi.getData();

    }

}

//MyServiceTest.java

package com.example.mockingandstubbing;

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() {

        ExternalApi mockApi = mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);

        String result = service.fetchData();

        assertEquals("Mock Data", result);

    }

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 2: Verifying Interactions**

**Code:**

//ExternalApi.java

package com.example.verifyinginteractions;

public interface ExternalApi {

    String getData();

}

//MyService.java

package com.example.verifyinginteractions;

public class MyService {

    private ExternalApi externalApi;

    public MyService(ExternalApi externalApi) {

        this.externalApi = externalApi;

    }

    public String fetchData() {

        return externalApi.getData();

    }

}

//MyServiceTest.java

package com.example.verifyinginteractions;

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() {

        ExternalApi mockApi = mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);

        String result = service.fetchData();

        assertEquals("Mock Data", result);

    }

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Logging using SLF4J**

**Exercise 1: Logging Error Messages and Warning Levels**

**Code:**

//LoggingExample.java

package com.example.loggingerrormessages;

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");

    }

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.