**JUnit**

**Exercise 1: Setting Up JUnit**

**Pom.xml**

<dependencies>

<!-- JUnit 4 dependency -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

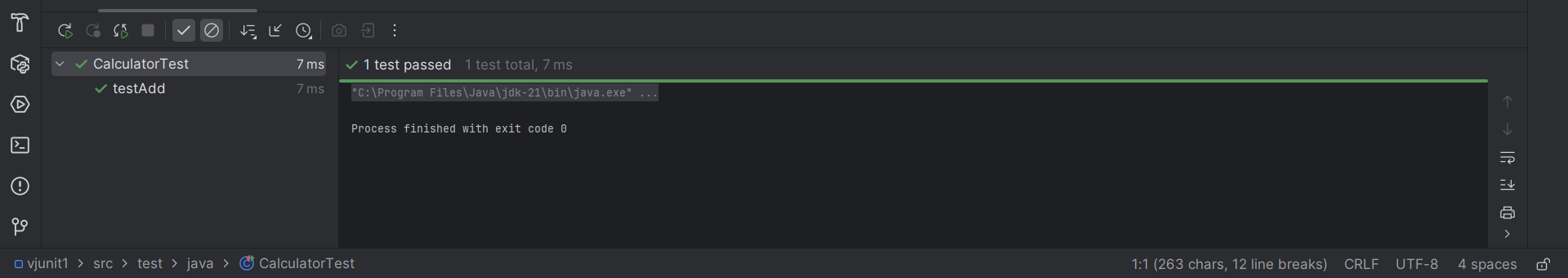
**Calculator.java**

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

**CalculatorTest.java**

import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class CalculatorTest {  
  
 @Test  
 public void testAdd() {  
 Calculator calculator = new Calculator();  
 int result = calculator.add(10, 5);  
 *assertEquals*(15, result);  
 }  
}

**Output:**

****

**Exercise 3: Assertions in JUnit**

**Pom.xml**

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

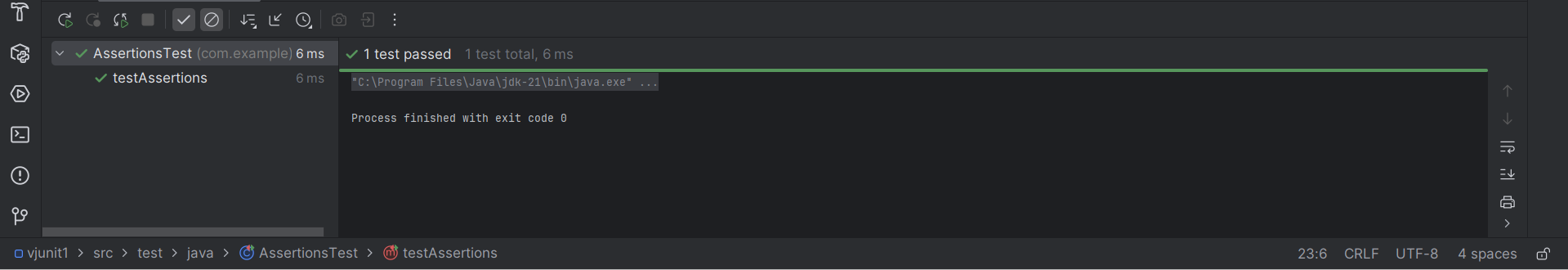
</dependency>

</dependencies>

**AssertionsTest.java**

package com.example;  
  
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  
 *assertNull*(null);  
  
 // Assert not null  
 }  
}

**Output:**

****

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

**Pom.xml**

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

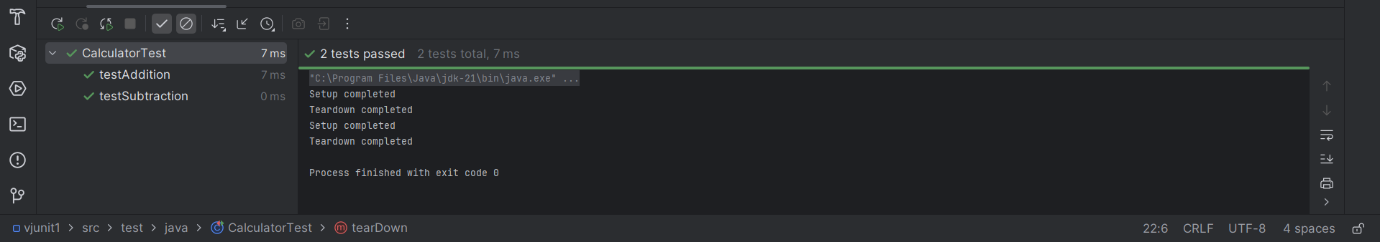
**Calculator.java**

public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
  
 public int subtract(int a, int b) {  
 return a - b;  
 }  
}

**CalculatorTest.java**

import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class CalculatorTest {  
  
 private Calculator calculator;  
  
 // Setup - runs before each test  
 @Before  
 public void setUp() {  
 calculator = new Calculator(); // Arrange  
 System.*out*.println("Setup completed");  
 }  
  
 // Teardown - runs after each test  
 @After  
 public void tearDown() {  
 calculator = null;  
 System.*out*.println("Teardown completed");  
 }  
  
 @Test  
 public void testAddition() {  
 // Arrange - already done in setUp()  
  
 // Act  
 int result = calculator.add(10, 5);  
  
 // Assert  
 *assertEquals*(15, result);  
 }  
  
 @Test  
 public void testSubtraction() {  
 // Act  
 int result = calculator.subtract(10, 5);  
  
 // Assert  
 *assertEquals*(5, result);  
 }  
}

**Output:**



Week-2:

Exercise 1: Mocking and Stubbing

Create a project name – MockitoDemo

Pom.xml

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.mockito</groupId>

<artifactId>MockitoDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Use JUnit 5 platform -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

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

<version>2.22.2</version>

<configuration>

<includes>

<include>/\*Test.java</include>

</includes>

</configuration>

</plugin>

</plugins>

</build>

</project>

in src/main/java, create com.mockito

ExternalApi.java

package com.mockito;

public interface ExternalApi {

String getData();

}

MyService.java :

package com.mockito;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

    }

}

package-info.java - package com.mockito;

in src/test/java, create com.mockito

MyServiceTest.java

package com.mockito;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

// Arrange

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

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

// Act

MyService service = new MyService(mockApi);

String result = service.fetchData();

// Assert

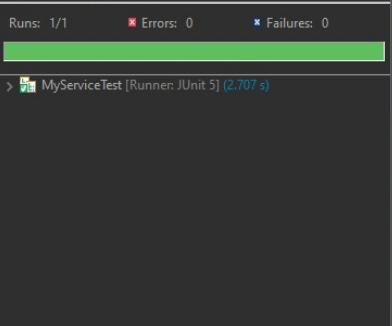
assertEquals("Mock Data", result);

    }

}

package-info.java - package com.mockito;

Output:



Exercise 2: Verifying Interactions

Create a project name – VerifyDemo

Pom.xml

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.verify</groupId>

<artifactId>VerifyDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

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

<version>2.22.2</version>

<configuration>

<includes>

<include>/\*Test.java</include>

</includes>

</configuration>

</plugin>

</plugins>

</build>

</project>

in src/main/java, create com.verify

ExternalApi.java

package com.verify;

public interface ExternalApi {

String getData();

}

MyService.java

package com.verify;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

    }

}

package-info.java - package com.verify

in src/test/java, create com.verify

MyServiceTest.java :

package com.verify;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Arrange

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

// Act

MyService service = new MyService(mockApi);

service.fetchData();

// Assert

verify(mockApi).getData(); // Verifies that getData() was called

    }

}

package-info.java - package com.verify;

Right-click MyServiceTest.java

Click Run As → JUnit Test

Output:

