|  |  |
| --- | --- |
| **WEEK 2 – JUnit Testing** | **Superset Id : 6429486**  **Name : Akshaya V** |

**JUnit Testing Exercises:**

**Exercise 1 : Setting up JUnit**

**Scenario:** You need to set up JUnit in your Java project to start writing unit tests.

**Steps:**

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

3. Create a new test class in your project.

**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>org.example</groupId>  
 <artifactId>JUnitProject</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <properties>  
 <maven.compiler.source>24</maven.compiler.source>  
 <maven.compiler.target>24</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

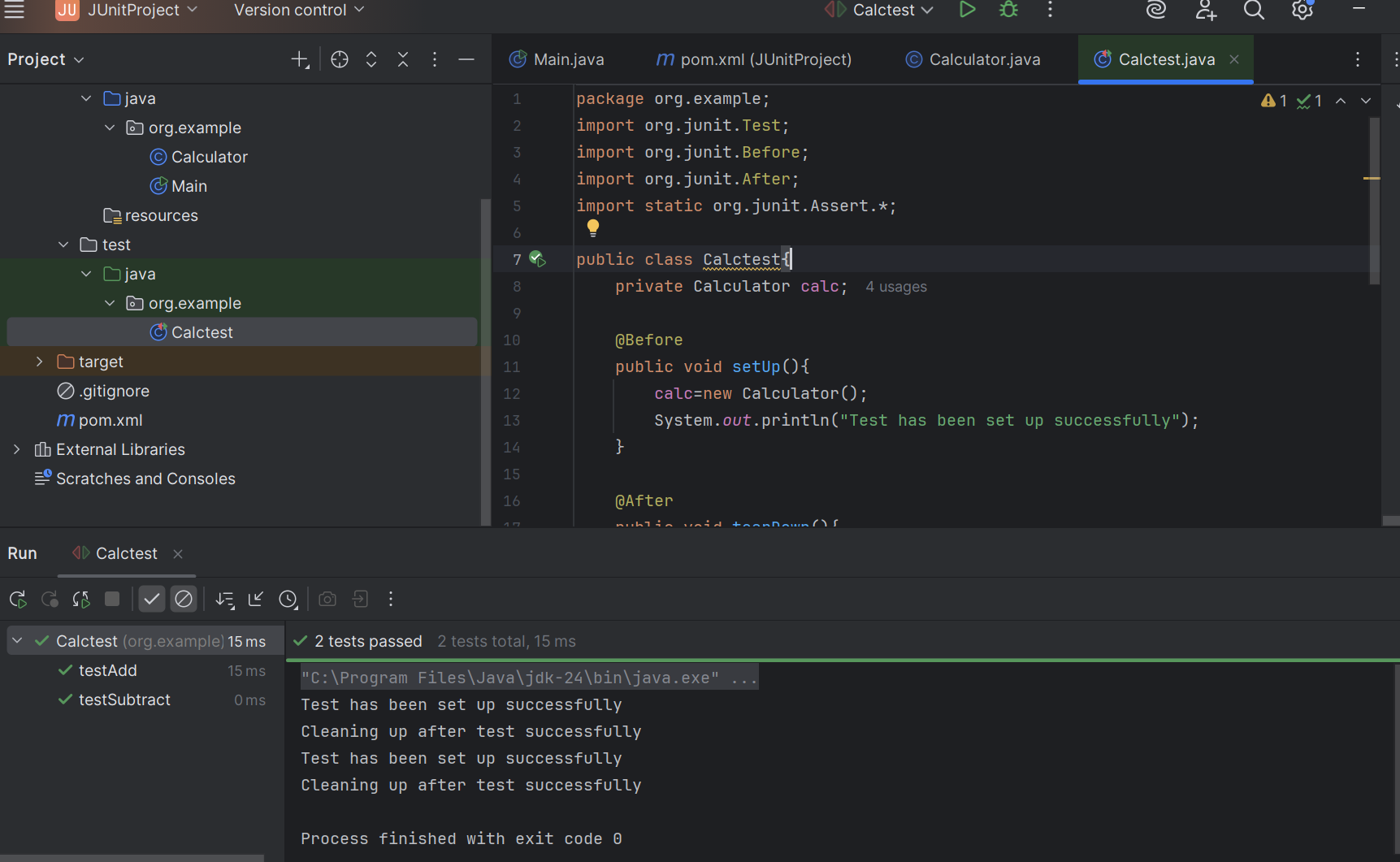
**Calculator.java:**

package org.example;  
public class Calculator{  
 public int addition(int a,int b){  
 return a+b;  
 }  
 public int subtraction(int a,int b){  
 return a-b;}}

**Calctest.java:**

package org.example;  
import org.junit.Test;  
import org.junit.Before;  
import org.junit.After;  
import static org.junit.Assert.\*;  
public class Calctest{  
 private Calculator calc;  
 @Before  
 public void setUp(){  
 calc=new Calculator();  
 System.*out*.println("Test has been set up successfully");  
 }  
 @After  
 public void tearDown(){  
 calc=null;  
 System.*out*.println("Cleaning up after test successfully");  
 }  
 @Test  
 public void testAdd(){  
 int result=calc.addition(5, 3);  
 *assertEquals*(8, result);  
 }  
 @Test  
 public void testSubtract(){  
 int result=calc.subtraction(10, 4);  
 *assertEquals*(6, result);  
 }  
}

**Output:**



**Exercise 3: Assertions in JUnit**

**Scenario: You need to use different assertions in JUnit to validate your test results.**

**Steps:**

1. Write tests using various JUnit assertions.

**Solution Code:**

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

assertNotNull(new Object());

}

}

**Code:**

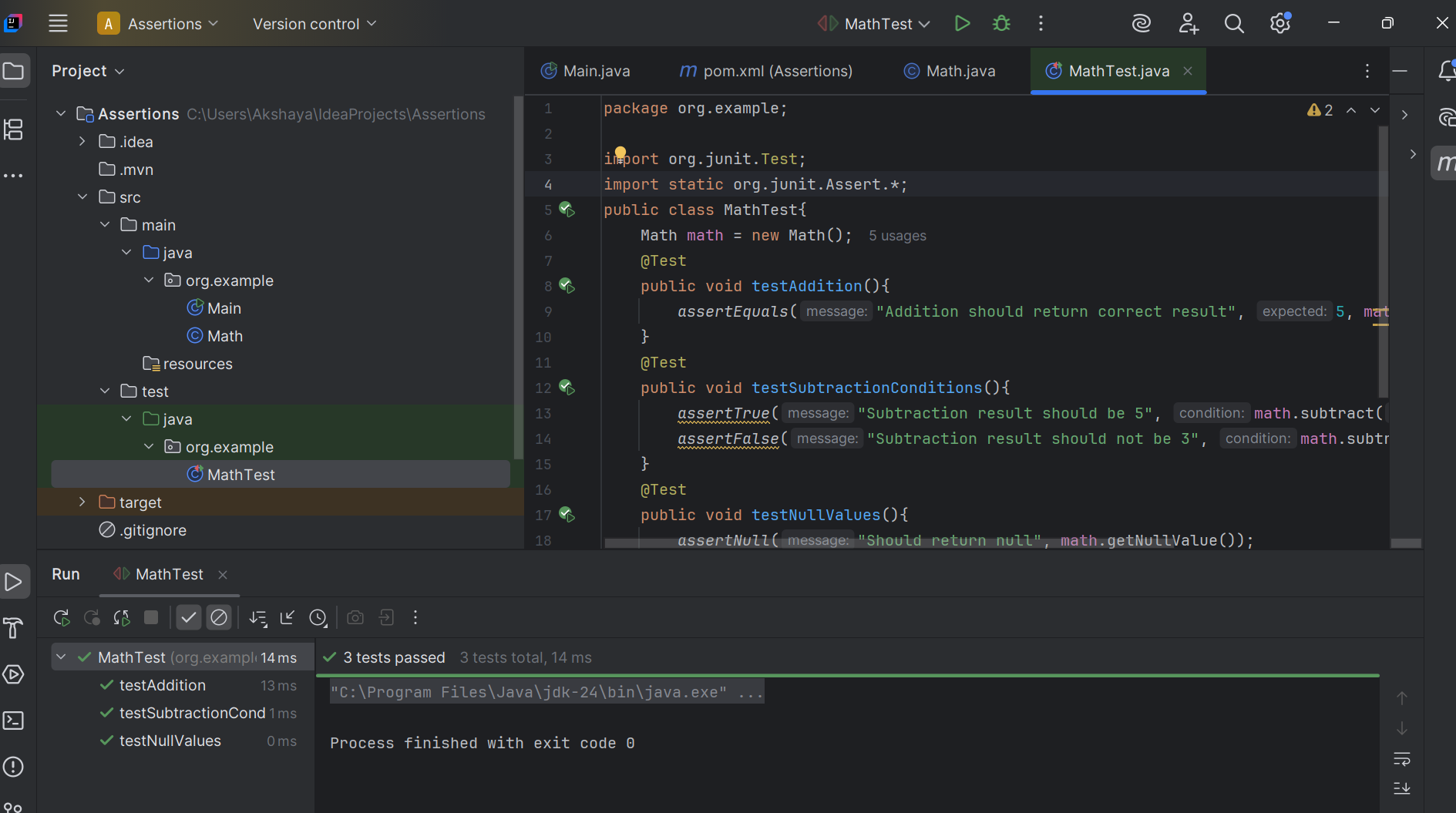
**Math.java:**

package org.example;  
public class Math {  
 public int add(int a, int b) {  
 return a+b;  
 }  
 public int subtract(int a, int b) {  
 return a-b;  
 }  
 public String getNullValue() {  
 return null;  
 }  
 public String getName() {  
 return "Akshaya";  
 }  
}

**MathTest.java:**

package org.example;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
public class MathTest{  
 Math math = new Math();  
 @Test  
 public void testAddition(){  
 *assertEquals*("Addition should return correct result", 5, math.add(2, 3));  
 }  
 @Test  
 public void testSubtractionConditions(){  
 *assertTrue*("Subtraction result should be 5", math.subtract(10, 5) == 5);  
 *assertFalse*("Subtraction result should not be 3", math.subtract(10, 5) == 3);  
 }  
 @Test  
 public void testNullValues(){  
 *assertNull*("Should return null", math.getNullValue());  
 *assertNotNull*("Name should not be null", math.getName());  
 }  
}

**Output:**



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

**Scenario:** You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

**Steps:**

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods.

**Code:**

**SmartCalculator.java:**

package org.example;  
public class SmartCalculator{  
 public int add(int a,int b){  
 return a+b;  
 }  
 public int subtract(int a,int b){  
 return a-b;  
 }  
 public int multiply(int a,int b){  
 return a\*b;  
 }  
 public int divide(int a,int b){  
 if(b==0){  
 throw new IllegalArgumentException("Division by zero is not allowed.");  
 }  
 return a/b;  
 }  
 public boolean isEven(int number){  
 return number%2==0;  
 }  
 public String getNullValue(){  
 return null;  
 }  
 public String getName(){  
 return "Akshaya";  
 }  
}

**SmartCalculatorTest.java:**

package org.example;  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
public class SmartCalculatorTest{  
 private SmartCalculator calculator;  
 @Before  
 public void setUp(){  
 calculator=new SmartCalculator();  
 System.*out*.println("SmartCalculator instance created.");  
 }  
 @After  
 public void tearDown(){  
 calculator=null;  
 System.*out*.println("SmartCalculator instance cleared.\n");  
 }  
 @Test  
 public void testAddition(){  
 int a=8,b=5;  
 int result=calculator.add(a,b);  
 *assertEquals*("Addition of 8+5 should be 13",13,result);  
 }  
 @Test  
 public void testSubtraction(){  
 int x=20,y=7;  
 int result=calculator.subtract(x,y);  
 *assertEquals*("Subtraction of 20-7 should be 13",13,result);  
 }  
 @Test  
 public void testMultiplication(){  
 int p=4,q=5;  
 int product=calculator.multiply(p,q);  
 *assertEquals*("Multiplication of 4\*5 should be 20",20,product);  
 }  
 @Test  
 public void testDivision() {  
 int dividend=16,divisor=4;  
 int quotient=calculator.divide(dividend,divisor);  
 *assertEquals*("Division of 16/4 should be 4",4,quotient);  
 }  
 @Test(expected=IllegalArgumentException.class)  
 public void testException(){  
 calculator.divide(10,0);  
 }  
 @Test  
 public void testIsEvenTrue(){  
 boolean result=calculator.isEven(12);  
 *assertTrue*("12 is an even number",result);  
 }  
 @Test  
 public void testIsEvenFalse(){  
 boolean result = calculator.isEven(7);  
 *assertFalse*("7 is an odd number",result);  
 }  
 @Test  
 public void testNull(){  
 *assertNull*("getNullValue() should return null",calculator.getNullValue());  
 }  
 @Test  
 public void testNotNull(){  
 *assertNotNull*("getName() should not return null",calculator.getName());  
 }  
}

**Output:**