**Setting Up JUnit**

**JUnit\_Basic Testing**

Vaishnavi

28/6/2025

JUnit is a Java testing framework used to write unit tests — small pieces of code that test individual methods in your program.

This Java project demonstrates how to set up and use JUnit for unit testing a simple Calculator class. It verifies arithmetic operations like addition, subtraction, multiplication, division, and other logic using automated test cases.

**Objective:**

* **Ensure Code Accuracy:** Validate that each method in the Calculator class performs the expected operation correctly.
* **Handle Edge Cases:** Test for exceptional conditions like division by zero to ensure application robustness.
* **Automate Testing:** Use JUnit to automate the testing process, enabling faster development and reliable code changes.

**Implementation:**

### Create a New Java Project

**IntelliJ IDEA**: File → New → Project → Java

### Add JUnit to Your Project

#### By using ****Maven****: add the dependencies in pom.xml

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**Create a Sample Java Class and Test Class**

**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;

}

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

if (b == 0) throw new ArithmeticException("Cannot divide by zero");

return a / b;

}

public int square(int a) {

return a \* a;

}

public int max(int a, int b) {

return a > b ? a : b;

}

public boolean isEven(int a) {

return a % 2 == 0;

}

}

**CalculatorTest.java:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

Calculator calc = new Calculator();

@Test

public void testAddPositive() {

assertEquals(5, calc.add(2, 3));

}

@Test

public void testAddNegative() {

assertEquals(-5, calc.add(-2, -3));

}

@Test

public void testSubtract() {

assertEquals(1, calc.subtract(4, 3));

}

@Test

public void testMultiply() {

assertEquals(12, calc.multiply(3, 4));

}

@Test

public void testMultiplyByZero() {

assertEquals(0, calc.multiply(0, 10));

}

@Test

public void testDivide() {

assertEquals(2, calc.divide(10, 5));

}

@Test(expected = ArithmeticException.class)

public void testDivideByZero() {

calc.divide(10, 0);

}

@Test

public void testSquare() {

assertEquals(16, calc.square(4));

}

@Test

public void testMax() {

assertEquals(9, calc.max(7, 9));

}

@Test

public void testIsEvenTrue() {

assertTrue(calc.isEven(6));

}

@Test

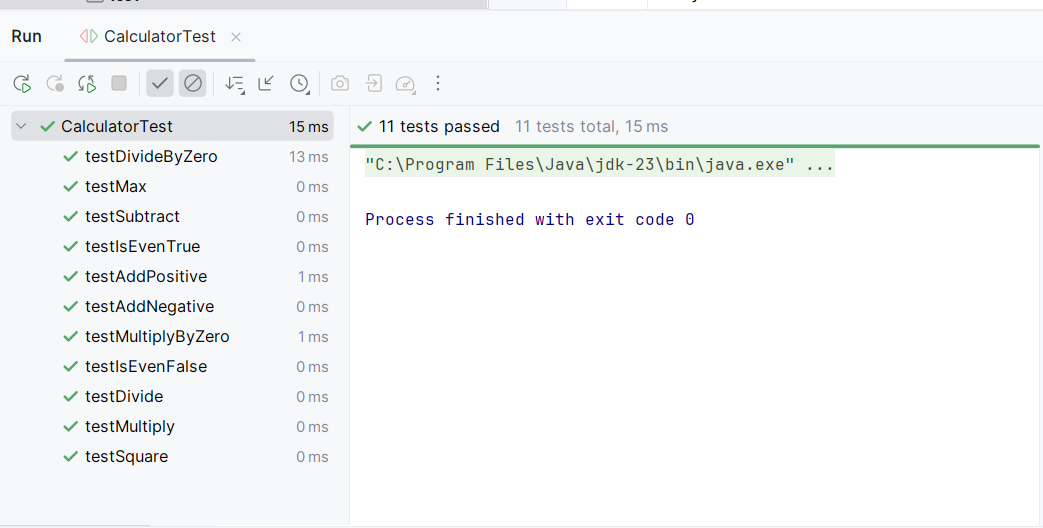
public void testIsEvenFalse() {

assertFalse(calc.isEven(5));

}

}

**Output:**

****