JUnit Testing

JUnit is a popular testing framework for Java used to write and run unit tests. versions: JUnit 4 and JUnit 5 (JUnit Jupiter)

JUnit Architecture

Test Class: Contains test methods Test Method: Annotated with @Test

Assertions: Methods to check expected results (assertEquals, assertTrue, etc.)

Annotations:

@Test - marks a test method

@BeforeEach / @Before - runs before each test

@AfterEach / @After - runs after each test

@BeforeAll / @AfterAll - runs once before/after all tests

```
package org.example;

∨ □ JavaUnitTestExample ~/IdeaProjects/JavaUnitT

  > 🗀 .idea
                                                 import org.junit.jupiter.api.Test;
    mvn.

∨ □ src

                                                 import static org.junit.jupiter.api.Assertions.*;
    main
      java
                                                class CalculatorTest {

√ org.example

                                           8
             Calculator
                                          9
                                                     private final Calculator myCalc = new Calculator(); 2 usages
             (C) Main
                                          10
             © Service
                                          11 🗅
                                                     void add() {
                                                         assertEquals( expected: 5, myCalc.add( a: 4, b: 1));
                                          12
        resources

∨ □ test

                                          13
                                          14
      🗸 🗀 java

✓ org.example

                                          16 🗅
                                                     void subtract() {
             CalculatorTest
                                          17
                                                        assertEquals( expected: 4, myCalc.subtract( a: 12, b: 8));
  > Target
                                          18
    .gitignore
                                          19
    JavaUnitTestExample.iml
    m pom.xml
```

Example of a Test Class Structure:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
class CalculatorTest {
    @Test
    void testAddition() {
        Calculator calc = new Calculator();
        assertEquals(5, calc.add(2, 3));
    }
}
```

Common Assertions in JUnit

Assertion	Description	Example
<pre>assertEquals(expected , actual)</pre>	Checks if two values are equal	<pre>assertEquals(5, calc.add(2,3));</pre>
assertTrue(condition)	Checks if condition is true	<pre>assertTrue(calc.isPosi tive(3));</pre>
<pre>assertFalse(condition)</pre>	Checks if condition is false	<pre>assertFalse(calc.isPos itive(-1));</pre>
assertThrows(Exception.class, executable)	Checks if α method throws an exception	<pre>assertThrows(Arithmeti cException.class, () - > calc.divide(5,0));</pre>
assertNull(object)	Checks if object is null	<pre>assertNull(calc.getVal ue());</pre>
assertNotNull(object)	Checks if object is not null	<pre>assertNotNull(calc.get Value());</pre>

Writing Your First JUnit Test

Step-by-step process:

Create a class to test (Calculator.java)

Create a JUnit test class (CalculatorTest.java)

Write test methods using @Test

Use assertions to validate expected behavior

Run tests using IDE or Maven/Gradle

```
class Calculator {
    int add(int a, int b) { return a + b; }
    int subtract(int a, int b) { return a - b; }
}
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class CalculatorTest {
    Calculator calc = new Calculator();
    @Test
    void testAdd() {
        assertEquals(5, calc.add(2, 3));
    }
    @Test
    void testSubtract() {
        assertEquals(2, calc.subtract(5, 3));
    }
}
```