**4. Arrange-Act-Assert**

// File: src/test/java/com/example/CalculatorTestAAA.java

package com.example;

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

/\*\*

\* JUnit‑4 demo that follows the Arrange‑Act‑Assert pattern and

\* uses @Before / @After for common setup and teardown.

\*/

public class CalculatorTestAAA {

// ---- Simple system‑under‑test -----------------------------------------

/\*\*

\* Tiny Calculator so this single file compiles independently.

\* Move it to src/main/java if you prefer.

\*/

static class Calculator {

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

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

int multiply(int a, int b) { return a \* b; }

int divide(int a, int b) { return a / b; } // may throw

}

// -----------------------------------------------------------------------

private Calculator calc; // shared test fixture

/\*\* Set‑up runs before \*each\* @Test \*/

@Before

public void setUp() {

// Arrange (common part) – create fresh Calculator

calc = new Calculator();

}

/\*\* Tear‑down runs after \*each\* @Test \*/

@After

public void tearDown() {

// Clean up (not strictly needed here, but shows the pattern)

calc = null;

}

// -----------------------------------------------------------------------

// Individual tests, each expressed as Arrange‑Act‑Assert (AAA)

// -----------------------------------------------------------------------

@Test

public void add\_shouldReturnSum() {

// ---------- Arrange ----------

int a = 2;

int b = 3;

// ---------- Act --------------

int result = calc.add(a, b);

// ---------- Assert -----------

assertEquals("2 + 3 must equal 5", 5, result);

}

@Test

public void subtract\_shouldReturnDifference() {

// Arrange

int a = 10, b = 4;

// Act

int result = calc.subtract(a, b);

// Assert

assertEquals("10 − 4 must equal 6", 6, result);

}

@Test

public void multiply\_shouldReturnProduct() {

// Arrange

int a = 7, b = 6;

// Act

int result = calc.multiply(a, b);

// Assert

assertEquals("7 × 6 must equal 42", 42, result);

}

@Test

public void divide\_divideByZero\_shouldThrowException() {

// Arrange

int numerator = 8, denominator = 0;

// ---------- Act & Assert combined ----------

try {

calc.divide(numerator, denominator);

fail("Expected ArithmeticException when dividing by zero");

} catch (ArithmeticException ex) {

// Assert message contains "/ by zero"

assertTrue(ex.getMessage().contains("/ by zero"));

}

}

}