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

Code:

Calculator.java  
  
package org.example;  
  
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;  
 }  
}

CalculatorTest.java

package com.example;

import org.example.Calculator;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

private Calculator calc;

// Setup method (runs before each test)

@Before

public void setUp() {

calc = new Calculator();

System.out.println("Setup: New Calculator instance created");

}

// Teardown method (runs after each test)

@After

public void tearDown() {

calc = null;

System.out.println("Teardown: Calculator instance set to null");

}

@Test

public void testAdd() {

// Arrange

int a = 6;

int b = 4;

// Act

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

// Assert

assertEquals(10, result);

}

@Test

public void testSubtract() {

// Arrange

int a = 5;

int b = 3;

// Act

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

// Assert

assertEquals(2, result);

}

@Test

public void testMultiply() {

// Arrange

int a = 4;

int b = 5;

// Act

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

// Assert

assertEquals(20, result);

}

@Test

public void testDivide() {

// Arrange

int a = 9;

int b = 3;

// Act

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

// Assert

assertEquals(3, result);

}

@Test(expected = ArithmeticException.class)

public void testDivideByZero() {

// Arrange

int a = 10;

int b = 0;

// Act

calc.divide(a, b);

// Assert is handled by the expected exception

}

}

Output:

