**SOFTWARE TESTING**

**(CSE 455)**

**LAB # 08**

|  |  |
| --- | --- |
| **NAME:** | MUAAZ BIN MUKHTAR |
| **REG NO:** | FA21-BSE-045 |
| **CLASS & SECTION:** | BSSE-7A |
| **SUBMITTED TO:** | Mam Najmun Nisa |
| **DATE SUBMITTED:** | 29-10-2024 |



**Department of Computer Science**

**Calculator class:**

package calculator;

public class Calculator {

public int add(int a, int b) {

return a+b;

}

public int sub(int a, int b) {

return a-b;

}

public int mul(int a, int b) {

return a\*b;

}

public float div(int a, int b) {

return a/b;

}

}

**Calculator Test:**

package calculator;

import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.DisplayName;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.CsvSource;

class CalculatorTest {

@ParameterizedTest

@DisplayName("Addition Tests")

@CsvSource({

"1.0, 2.0, 3.0",

"5.5, 4.5, 10.0",

"-3.0, 3.0, 0.0",

"0.0, 0.0, 0.0"

})

public void testAdd(double a, double b, double expected) {

assertEquals(expected, Calculator.add(a, b), 901, "Addition result is incorrect");

}

@ParameterizedTest

@DisplayName("Subtraction Tests")

@CsvSource({

"5.0, 3.0, 2.0",

"10.0, 5.0, 5.0",

"0.0, 3.0, -3.0",

"7.5, 2.5, 5.0"

})

public void testSubtract(double a, double b, double expected) {

assertEquals(expected, calculator.subtract(a, b), 0.001, "Subtraction result is incorrect");

}

@ParameterizedTest

@DisplayName("Multiplication Tests")

@CsvSource({

"3.0, 2.0, 6.0",

"5.0, 5.0, 25.0",

"0.0, 5.0, 0.0",

"-4.0, 3.0, -12.0"

})

public void testMultiply(double a, double b, double expected) {

assertEquals(expected, calculator.multiply(a, b), 0.001, "Multiplication result is incorrect");

}

@ParameterizedTest

@DisplayName("Division Tests")

@CsvSource({

"6.0, 2.0, 3.0",

"10.0, 5.0, 2.0",

"-9.0, 3.0, -3.0",

"5.0, 0.0, 0.0" // Division by zero case

})

public void testDivide(double a, double b, double expected) {

assertEquals(expected, calculator.divide(a, b), 0.001, "Division result is incorrect");

}

@ParameterizedTest

@DisplayName("Square Tests")

@CsvSource({

"3.0, 9.0",

"-4.0, 16.0",

"0.0, 0.0",

"5.0, 25.0"

})

public void testSquare(double a, double expected) {

assertEquals(expected, calculator.square(a), 0.001, "Square result is incorrect");

}

@ParameterizedTest

@DisplayName("Cube Tests")

@CsvSource({

"2.0, 8.0",

"-3.0, -27.0",

"0.0, 0.0",

"4.0, 64.0"

})

public void testCube(double a, double expected) {

assertEquals(expected, calculator.cube(a), 0.001, "Cube result is incorrect");

    }

}

**Calculator Class:**

package calculator;

public class Calculator {

public int add(int a, int b) {

return a+b;

}

public int sub(int a, int b) {

return a-b;

}

public int mul(int a, int b) {

return a\*b;

}

public float div(int a, int b) {

return a/b;

}

}

**Calculator Test:**

package calculator;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

class cal {

private final Calculator calculator =new Calculator();

@Test

void testAdd() {

assertEquals(2,calculator.add(1,1));

assertEquals(2,calculator.add(2,1));

}

@Test

void testSub() {

assertEquals(1,calculator.sub(2,1));

assertEquals(2,calculator.sub(1,1));

}

@Test

void testMul() {

assertEquals(2,calculator.mul(2,1));

assertEquals(2,calculator.mul(1,1));

}

@Test

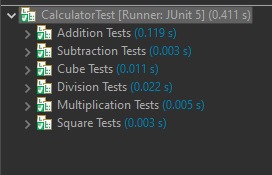
void testDiv() {

assertEquals(2,calculator.div(4,2));

assertEquals(2,calculator.div(1,1));

}

}



**Sandwich Class:**

package lab8;

public class Sandwich {

private String name;

private String size;

private int cheeseToppings;

private int pepperoniToppings;

// Constructor

public Sandwich(String name, String size, int cheeseToppings, int pepperoniToppings) {

this.name = name.toLowerCase();

this.size = size.toLowerCase();

this.cheeseToppings = cheeseToppings;

this.pepperoniToppings = pepperoniToppings;

}

public double TotalCost() {

double baseCost = 0;

double toppingCost = 0;

int totalToppings = cheeseToppings + pepperoniToppings;

if (name.equals("chicken")) {

if (size.equals("small")) {

baseCost = 10;

toppingCost = 2;

} else if (size.equals("medium")) {

baseCost = 15;

toppingCost = 3;

} else if (size.equals("large")) {

baseCost = 20;

toppingCost = 4;

}

} else if (name.equals("egg")) {

if (size.equals("small")) {

baseCost = 5;

toppingCost = 1;

} else if (size.equals("medium")) {

baseCost = 8;

toppingCost = 1;

} else if (size.equals("large")) {

baseCost = 10;

toppingCost = 1;

}

}

return baseCost + (toppingCost \* totalToppings);

}

public String getDescription() {

return String.format("Sandwich: %s, Size: %s, Cheese Toppings: %d, Pepperoni Toppings: %d, Total Cost: $%.2f",

name, size, cheeseToppings, pepperoniToppings, TotalCost());

    }

}

**Sandwich Test:**

package lab8;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.CsvSource;

import static org.junit.jupiter.api.Assertions.\*;

public class SandwichTest {

private Sandwich sandwich;

@BeforeAll

public static void setupBeforeAll() {

System.out.println("Starting Sandwich tests...");

}

@AfterAll

public static void teardownAfterAll() {

System.out.println("All tests completed.");

}

@BeforeEach

public void setup() {

// Initializing a Sandwich instance before each test

sandwich = new Sandwich("chicken", "medium", 2, 1);

}

@Test

public void testTotalCost() {

double expectedCost = 15 + (3 \* 3); // Medium chicken sandwich with 3 toppings

assertEquals(expectedCost, sandwich.TotalCost(), "Total cost calculation is incorrect.");

}

@Test

public void testGetDescription() {

String expectedDescription = "Sandwich: chicken, Size: medium, Cheese Toppings: 2, Pepperoni Toppings: 1, Total Cost: $21.00";

assertEquals(expectedDescription, sandwich.getDescription(), "Description generation is incorrect.");

}

@Test

public void testNotSameInstance() {

Sandwich anotherSandwich = new Sandwich("egg", "small", 1, 1);

assertNotSame(sandwich, anotherSandwich, "Two sandwich instances should not be the same.");

}

@Test

public void testNullSandwich() {

Sandwich nullSandwich = null;

assertNull(nullSandwich, "Sandwich should be null.");

}

// Parameterized test to check different sandwich configurations

@ParameterizedTest

@CsvSource({

"chicken, small, 2, 1, 14.00",

"chicken, medium, 0, 0, 15.00",

"chicken, large, 3, 2, 40.00",

"egg, small, 1, 1, 7.00",

"egg, medium, 2, 2, 12.00",

"egg, large, 0, 0, 10.00"

})

public void testTotalCostParameterized(String name, String size, int cheeseToppings, int pepperoniToppings, double expectedCost) {

Sandwich parameterizedSandwich = new Sandwich(name, size, cheeseToppings, pepperoniToppings);

assertEquals(expectedCost, parameterizedSandwich.TotalCost(), 0.01, "Parameterized cost calculation is incorrect.");

    }

}

