**SOFTWARE TESTING**

**(CSE 455)**

**LAB # 07**

|  |  |
| --- | --- |
| **NAME:** | MUAAZ BIN MUKHTAR |
| **REG NO:** | FA21-BSE-045 |
| **CLASS & SECTION:** | BSSE-7A |
| **SUBMITTED TO:** | Mam Najmun Nisa |
| **DATE SUBMITTED:** | 22-10-2024 |



**Department of Computer Science**

**Triangle class:**

package triangle;

public class Triangle {

private int A;

private int B;

private int C;

public Triangle(int A, int B, int C) {

this.A = A;

this.B = B;

this.C = C;

}

public boolean isValidTriangle() {

return (A + B > C) &&

(A + C > B) &&

(B + C > A);

}

public String getTriangleType() {

if (!isValidTriangle()) {

return "Not a triangle";

} else if (A == B && B == C) {

return "Equilateral triangle";

} else if (A == B || A == C || B == C) {

return "Isosceles triangle";

} else {

return "Scalene triangle";

    }

}

}

**Triangle Test:**

package triangle;

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

import org.junit.jupiter.api.Test;

class triTest {

@Test

public void testEquilateralTriangle() {

Triangle triangle = new Triangle(3, 3, 3);

assertEquals("Equilateral triangle", triangle.getTriangleType());

}

@Test

public void testIsoscelesTriangle() {

Triangle triangle = new Triangle(5, 5, 3);

assertEquals("Isosceles triangle", triangle.getTriangleType());

Triangle triangle2 = new Triangle(3, 5, 5);

assertEquals("Isosceles triangle", triangle2.getTriangleType());

}

@Test

public void testScaleneTriangle() {

Triangle triangle = new Triangle(3, 4, 5);

assertEquals("Scalene triangle", triangle.getTriangleType());

}

@Test

public void testNotATriangle() {

Triangle triangle = new Triangle(1, 2, 3);

assertEquals("Not atriangle", triangle.getTriangleType());

}

@Test

public void testInvalidTriangle() {

Triangle triangle = new Triangle(1, 10, 12);

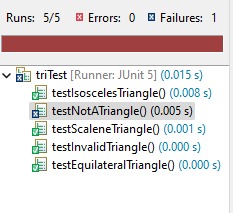
assertEquals("Not a triangle", triangle.getTriangleType());

Triangle triangle2 = new Triangle(0, 0, 0); // Invalid sides

assertEquals("Not a triangle", triangle2.getTriangleType());

    }

}



**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

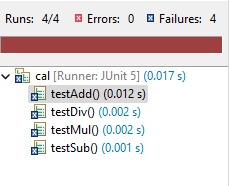
void testDiv() {

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

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

}

}



**Number Class:**

package number;

public class Number {

private int number;

public Number(int number) {

this.number = number;

}

public boolean isPalindrome() {

int original = number;

int reversed = 0;

int temp = original;

while (temp != 0) {

int digit = temp % 10;

reversed = reversed \* 10 + digit;

temp /= 10;

}

return original == reversed;

}

public long factorial() {

if (number < 0) {

throw new IllegalArgumentException("Factorial is not defined for negative numbers.");

}

long fact = 1;

for (int i = 2; i <= number; i++) {

fact \*= i;

}

return fact;

}

public int sumOfDigits() {

int sum = 0;

int temp = number;

while (temp != 0) {

sum += temp % 10;

temp /= 10;

}

return sum;

    }

}

**Number Test:**

package number;

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

import org.junit.jupiter.api.Test;

class NumTest {

@Test

public void testIsPalindrome() {

Number number1 = new Number(121);

assertTrue(number1.isPalindrome());

Number number2 = new Number(123);

assertFalse(number2.isPalindrome());

Number number3 = new Number(0);

assertTrue(number3.isPalindrome());

Number number4 = new Number(-121);

assertFalse(number4.isPalindrome());

}

@Test

public void testFactorial() {

Number number1 = new Number(5);

assertEquals(120, number1.factorial());

Number number2 = new Number(0);

assertEquals(1, number2.factorial());

Number number3 = new Number(1);

assertEquals(1, number3.factorial());

Number number4 = new Number(10);

assertEquals(3628800, number4.factorial());

// Test for negative number

Number number5 = new Number(-5);

assertThrows(IllegalArgumentException.class, number5::factorial);

}

@Test

public void testSumOfDigits() {

Number number1 = new Number(123);

assertEquals(6, number1.sumOfDigits());

Number number2 = new Number(0);

assertEquals(0, number2.sumOfDigits());

Number number3 = new Number(9876);

assertEquals(30, number3.sumOfDigits());

Number number4 = new Number(555);

assertEquals(15, number4.sumOfDigits());

Number number5 = new Number(-123);

assertEquals(-6, number5.sumOfDigits());

    }

}

