Rafael Martín Berenguer 1º DAM

Ejercicio de JUNIT:

Programa con los métodos de calculadora:

CalculadoraMultTest.java test.java Ca

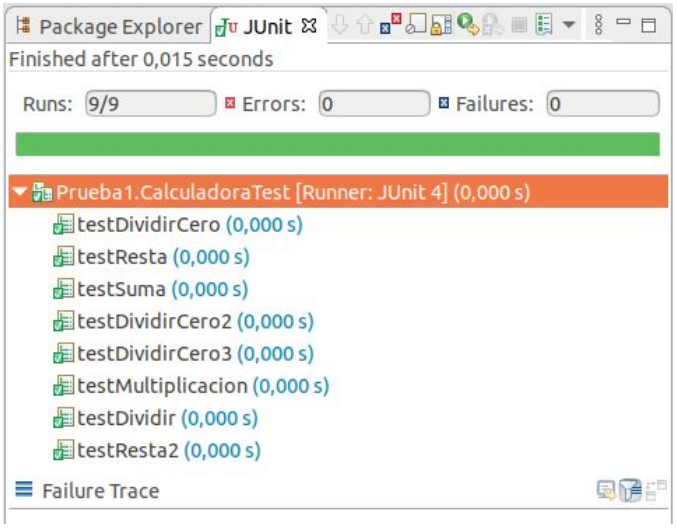
```java
package Prueba1;

public class Calculadora {

    private int num1;
    private int num2;

    public Calculadora() {}

    public Calculadora(int a, int b) {
        this.num1=a;
        this.num2=b;
    }

    public int suma() {
        int result = num1 + num2;
        return result;
    }

    public int resta() {
        int resul;

        if (resta2())
            resul = num1 - num2;
        else
            resul = num2 - num1;
        return resul;
    }

    public boolean resta2() {
        if (num1 >= num2)
            return true;
        else
            return false;
    }

    public Integer divide2() {
        if (num2 == 0)
            return null;

        int resul = num1 / num2;

        return resul;
        }

    public int multiplicar() {
        int result = num1 * num2;
        return result;
    }

    public int dividir() {
        int result = num1 / num2;
        return result;
    }
```

Casos de prueba para el programa definido anteriormente:

CalculadoraMultTest.java | test.java | Calculadora.java | Calculado

```java
package Prueba1;

import static org.junit.Assert.*;

public class CalculadoraTest {

    //Si la suma devuele el resultado esperado está correcto.
    @Test
    public void testSuma() {
        Calculadora calc = new Calculadora (20,10);
        int resultado = calc.suma();
        assertEquals("Fallo en la suma" ,30, resultado);
    }

    //Si la resta devuele el resultado esperado está correcto.
    @Test
    public void testResta() {
        Calculadora calc = new Calculadora (20,10);
        int resultado = calc.resta();
        assertEquals("Fallo en la Resta", 10, resultado);
    }

    //Si en la resta el num1 es mayor al num2 devuelve True.
    @Test
    public void testResta2() {
        Calculadora calc = new Calculadora (20,10);
        boolean resultado = calc.resta2();
        assertTrue("Fallo en la Resta", resultado);
    }

    //Si en la resta el num1 es menor al num2 devuelve False.
    @Test
    public void testResta3() {
        Calculadora calc = new Calculadora (10,20);
        assertFalse("Resta devuelve el resultado", calc.resta2());
    }

    //Prueba para cuando el num2 sea igual a 0
    @Test
    public void testDividirCero() {
        try {
            Calculadora calc = new Calculadora (20,0);
            int resultado = calc.dividir();
            fail("FALLO");
        } catch (ArithmeticException e) {
            // TODO: handle exception
        }

    }

    //Devuelve null si el num2 es igual a 0
    @Test
    public void testDividirCero2() {
            Calculadora calc = new Calculadora (20,0);
            Integer resultado = calc.divide2();
            assertNull("FALLO", resultado);
    }

    //Devuelve NotNull si la division es correcta.
    @Test
    public void testDividirCero3() {
           Calculadora calc = new Calculadora (20,1);
           int resultado = calc.divide2();
           assertNotNull("FALLO", resultado);
    }

    //Si la multiplicación devuele el resultado esperado está correcto.
    @Test
    public void testMultiplicacion() {
        Calculadora calc = new Calculadora (20,10);
        int resultado = calc.multiplicar();
        assertEquals("Fallo en la multiplicación", 200, resultado);
    }

    //Si la división devuele el resultado esperado está correcto.
    @Test
    public void testDividir() {
        Calculadora calc = new Calculadora (20,10);
        int resultado = calc.dividir();
        assertEquals("Fallo al dividir", 2, resultado);
    }


}
```

Resultados de los casos de prueba anteriores:



Caso de prueba para la multiplicación con parámetros:

CalculadoraMultTest.java | test.java | Calculadora.java

```java
package Prueba1;

import static org.junit.Assert.assertEquals;

import java.util.Arrays;
import java.util.Collection;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.class)
class CalculadoraMultTest {

    private int num1;
    private int num2;
    private int resul;

    public CalculadoraMultTest(int num1, int num2, int resul) {

        this.num1 = num1;
        this.num2 = num2;
        this.resul = resul;

    @Parameters
    public static Collection<Object[]> numeros() {
        return Arrays.asList(new Object[][] {
            {1, 10, 10},{2, 10, 20},{20, 20, 400}
        });
    }

    @Test
    public void testMultiplica() {
        Calculadora calc = new Calculadora (num1, num2);
        int resultado = calc.multiplicar();
        assertEquals(resul, resultado);
    }

}
```

Caso de prueba para la suma con parámetros:

CalculadoraMultTest.java | CalculadoraSumaTest.java | Calculadora.ja

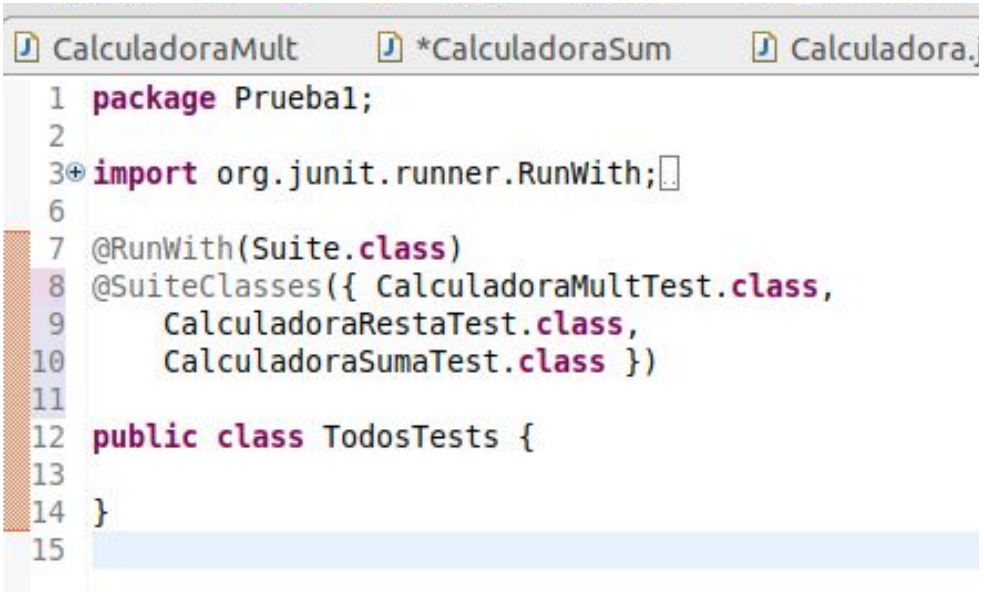
```java
package Prueba1;

import static org.junit.Assert.assertEquals;

@RunWith(Parameterized.class)
public class CalculadoraSumaTest {

    private int num1;
    private int num2;
    private int resul;

    public CalculadoraSumaTest(int num1, int num2, int resul) {

        this.num1 = num1;
        this.num2 = num2;
        this.resul = resul;
    }


    @Parameters
    public static Collection<Object[]> numeros() {
        return Arrays.asList(new Object[][] {
            {20, 10, 30},{50, 10, 60},{100, 20, 120}
        });
    }

    @Test
    public void testSuma() {
        Calculadora calc = new Calculadora (num1, num2);
        int resultado = calc.suma();
        assertEquals(resul, resultado);
    }

}
```

Caso de prueba para la resta con parámetros:

CalculadoraMultTe | CalculadoraSumaTe | Calculadora.java | Calcu

```java
package Prueba1;

import static org.junit.Assert.assertEquals;

import java.util.Arrays;
import java.util.Collection;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.class)
public class CalculadoraRestaTest {

    private int num1;
    private int num2;
    private int resul;

    public CalculadoraRestaTest(int num1, int num2, int resul) {

        this.num1 = num1;
        this.num2 = num2;
        this.resul = resul;
    }


    @Parameters
    public static Collection<Object[]> numeros() {
        return Arrays.asList(new Object[][] {
            {20, 10, 10},{50, 10, 40},{100, 20, 80}
        });
    }

    @Test
    public void testResta() {
        Calculadora calc = new Calculadora (num1, num2);
        int resultado = calc.resta();
        assertEquals(resul, resultado);
    }

}
```

Clase que ejecuta las diferentes pruebas anteriores que están en clases diferentes:



Resultado de las pruebas anteriores: