 ****

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA**

**Programación Móvil**

**NRC:** 6112

**Tema: Pruebas Unitarias**

**Nombres:**

* Martínez Carlos
* Monteros Paula

**SANGOLQUÍ – ECUADOR**

**MAYO – SEPTIEMBRE 2020**

**Clase CalculadoraTest**

/\*  
 \* ESPE - DCC - PROGRAMACIÓN MÓVIL  
 \* Sistema: Calculadora\_MVP  
 \* Creado 08/06/2020  
 \*  
 \* Los contenidos de este archivo son propiedad privada y están protegidos por  
 \* la licencia BSD  
 \*  
 \* Se puede utilizar, reproducir o copiar el contenido de este archivo.  
 \*/  
package com.example.calculadora\_mvp;  
  
import android.graphics.Path;  
  
import org.junit.Before;  
import org.junit.Test;  
  
import static org.junit.Assert.\*;  
  
*/\*\*  
 \* Clase para las pruebas unitarias que se realizarán en la aplicación.  
 \*  
 \** ***@author*** *Carlos Martínez  
 \** ***@author*** *Paula Monteros  
 \*/*public class CalculadoraTest {  
  
 private Operacion operacion;  
 private Numero numero;  
 private Numero numero1;  
 private Numero numero2;  
  
 */\*\*  
 \* Método set para instanciar la variables u objetos que se usarán  
 \* en la spruebas unitarias. \*  
 \*/* @Before  
 public void setUp() throws Exception {  
 operacion = new Operacion();  
 numero = new Numero(0.0);  
 numero1 = new Numero(0.0);  
 numero2 = new Numero(0.0);  
 }  
  
 */\*\*  
 \* Prueba unitaria para verificar que el objeto no sea NULL.  
 \*/* @Test  
 public void testNotNull(){  
 *assertNotNull*(operacion);  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación Suma entre 2 números.  
 \*/* @Test  
 public void testSuma() {  
 numero.setValor(3.0);numero1.setValor(1.0);numero2.setValor(2.0);  
 *assertEquals*(numero.getValor(),operacion.sumar(numero1,numero2).getValor());  
 numero.setValor(-5.0);numero1.setValor(-10.0);numero2.setValor(5.0);  
 *assertEquals*(numero.getValor(),operacion.sumar(numero1,numero2).getValor());  
 numero.setValor(4.28);numero1.setValor(2.14);numero2.setValor(2.14);  
 *assertEquals*(numero.getValor(),operacion.sumar(numero1,numero2).getValor());  
 numero.setValor(666000.0);numero1.setValor(-123456.0);numero2.setValor(789456.0);  
 *assertEquals*(numero.getValor(),operacion.sumar(numero1,numero2).getValor());  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación Resta entre 2 números.  
 \*/* @Test  
 public void testResta() {  
 numero.setValor(-1.0);numero1.setValor(1.0);numero2.setValor(2.0);  
 *assertEquals*(numero.getValor(),operacion.restar(numero1,numero2).getValor());  
 numero.setValor(-15.0);numero1.setValor(-10.0);numero2.setValor(5.0);  
 *assertEquals*(numero.getValor(),operacion.restar(numero1,numero2).getValor());  
 numero.setValor(0.0);numero1.setValor(2.14);numero2.setValor(2.14);  
 *assertEquals*(numero.getValor(),operacion.restar(numero1,numero2).getValor());  
 numero.setValor(-912912.0);numero1.setValor(-123456.0);numero2.setValor(789456.0);  
 *assertEquals*(numero.getValor(),operacion.restar(numero1,numero2).getValor());  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación Multiplicación entre 2 números.  
 \*/* @Test  
 public void testMultiplicacion() {  
 numero.setValor(2.0);numero1.setValor(1.0);numero2.setValor(2.0);  
 *assertEquals*(numero.getValor(),operacion.multiplicacion(numero1,numero2).getValor());  
 numero.setValor(-50.0);numero1.setValor(-10.0);numero2.setValor(5.0);  
 *assertEquals*(numero.getValor(),operacion.multiplicacion(numero1,numero2).getValor());  
 numero.setValor(4.5796);numero1.setValor(2.14);numero2.setValor(2.14);  
 *assertEquals*(numero.getValor(),operacion.multiplicacion(numero1,numero2).getValor());  
 numero.setValor(-9.7463079936E10);numero1.setValor(-123456.0);numero2.setValor(789456.0);  
 *assertEquals*(numero.getValor(),operacion.multiplicacion(numero1,numero2).getValor());  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación División entre 2 números.  
 \*/* @Test  
 public void testDivision() {  
 numero.setValor(0.5);numero1.setValor(1.0);numero2.setValor(2.0);  
 *assertEquals*(numero.getValor(),operacion.division(numero1,numero2).getValor());  
 numero.setValor(-2.0);numero1.setValor(-10.0);numero2.setValor(5.0);  
 *assertEquals*(numero.getValor(),operacion.division(numero1,numero2).getValor());  
 numero.setValor(1.0);numero1.setValor(2.14);numero2.setValor(2.14);  
 *assertEquals*(numero.getValor(),operacion.division(numero1,numero2).getValor());  
 numero.setValor(-0.1563811029367058);numero1.setValor(-123456.0);numero2.setValor(789456.0);  
 *assertEquals*(numero.getValor(),operacion.division(numero1,numero2).getValor());  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación exponencial.  
 \*/* @Test  
 public void testExponencial(){  
 numero.setValor(27.0);numero1.setValor(3.0);numero2.setValor(3.0);  
 *assertEquals*(numero.getValor(),operacion.exponencial(numero1,numero2).getValor());  
 numero.setValor(9.0);numero1.setValor(-3.0);numero2.setValor(2.0);  
 *assertEquals*(numero.getValor(),operacion.exponencial(numero1,numero2).getValor());  
 numero.setValor(1.0);numero1.setValor(3.0);numero2.setValor(0.0);  
 *assertEquals*(numero.getValor(),operacion.exponencial(numero1,numero2).getValor());  
 numero.setValor(-27.0);numero1.setValor(-3.0);numero2.setValor(3.0);  
 *assertEquals*(numero.getValor(),operacion.exponencial(numero1,numero2).getValor());  
 }  
  
 */\*\*  
 \* Prueba unitaria para la operación factorial de un número.  
 \*/* @Test  
 public void testFactorial(){  
 numero.setValor(6.0);numero1.setValor(3.0);  
 *assertEquals*(numero.getValor(),operacion.factorial(numero1).getValor());  
 numero.setValor(2.0);numero1.setValor(2.0);  
 *assertEquals*(numero.getValor(),operacion.factorial(numero1).getValor());  
 numero.setValor(40320.0);numero1.setValor(8.0);  
 *assertEquals*(numero.getValor(),operacion.factorial(numero1).getValor());  
 numero.setValor(1.0);numero1.setValor(1.0);  
 *assertEquals*(numero.getValor(),operacion.factorial(numero1).getValor());  
 }  
  
  
}

**Resultado de las Pruebas Unitarias Ejecutadas**

