 ****

**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 29/05/2020  
 \*  
 \* 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 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 Modelo;  
  
 */\*\*  
 \* Método set para instanciar un objeto de tipo  
 \* Modelo.  
 \*  
 \*/* @Before  
 public void setModelo(){  
 modelo = new Modelo();  
 }  
  
 */\*\*  
 \* Prueba unitaria para verificar que el objeto no sea NULL.  
 \*/* @Test  
 public void testNotNull(){  
 *assertNotNull*(modelo);  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación Suma entre 2 números.  
 \*/* @Test  
 public void testSuma() {  
 *assertEquals*("3.0", modelo.calcularM("1","2","Sumar"));  
 *assertEquals*("-5.0", modelo.calcularM("-10","5","Sumar"));  
 *assertEquals*("4.28", modelo.calcularM("2.14","2.14","Sumar"));  
 *assertEquals*("666000.0", modelo.calcularM("-123456","789456","Sumar"));  
 *assertEquals*("3.0", modelo.calcularM("1","2","Sumar"));  
 *assertEquals*("mal", modelo.calcularM("","","Sumar"));  
 *assertEquals*("mal", modelo.calcularM("","2","Sumar"));  
 *assertEquals*("mal", modelo.calcularM("2","","Sumar"));  
 *assertEquals*("mal", modelo.calcularM("asd","qwe","Sumar"));  
 *assertEquals*("mal", modelo.calcularM("...","\*\*\*","Sumar"));  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación Resta entre 2 números.  
 \*/* @Test  
 public void testResta(){  
 *assertEquals*("-1.0", modelo.calcularM("1","2","Restar"));  
 *assertEquals*("-15.0", modelo.calcularM("-10","5","Restar"));  
 *assertEquals*("0.0", modelo.calcularM("2.14","2.14","Restar"));  
 *assertEquals*("-912912.0", modelo.calcularM("-123456","789456","Restar"));  
 *assertEquals*("1.0", modelo.calcularM("2","1","Restar"));  
 *assertEquals*("mal", modelo.calcularM("","","Restar"));  
 *assertEquals*("mal", modelo.calcularM("","2","Restar"));  
 *assertEquals*("mal", modelo.calcularM("2","","Restar"));  
 *assertEquals*("mal", modelo.calcularM("asd","qwe","Restar"));  
 *assertEquals*("mal", modelo.calcularM("...","\*\*\*","Restar"));  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación Multiplicación entre 2 números.  
 \*/* @Test  
 public void testMultiplicacion() {  
 *assertEquals*("2.0", modelo.calcularM("1","2","Multiplicar"));  
 *assertEquals*("-50.0", modelo.calcularM("-10","5","Multiplicar"));  
 *assertEquals*("4.5796", modelo.calcularM("2.14","2.14","Multiplicar"));  
 *assertEquals*("-9.7463079936E10", modelo.calcularM("-123456","789456","Multiplicar"));  
 *assertEquals*("2.0", modelo.calcularM("1","2","Multiplicar"));  
 *assertEquals*("mal", modelo.calcularM("","","Multiplicar"));  
 *assertEquals*("mal", modelo.calcularM("","2","Multiplicar"));  
 *assertEquals*("mal", modelo.calcularM("2","","Multiplicar"));  
 *assertEquals*("mal", modelo.calcularM("asd","qwe","Multiplicar"));  
 *assertEquals*("mal", modelo.calcularM("...","\*\*\*","Multiplicar"));  
 }  
  
 */\*\*  
 \* Prueba Unitaria para la operación División entre 2 números.  
 \*/* @Test  
 public void testDivision(){  
 *assertEquals*("0.5", modelo.calcularM("1","2","Dividir"));  
 *assertEquals*("-2.0", modelo.calcularM("-10","5","Dividir"));  
 *assertEquals*("1.0", modelo.calcularM("2.14","2.14","Dividir"));  
 *assertEquals*("-0.1563811029367058", modelo.calcularM("-123456","789456","Dividir"));  
 *assertEquals*("-5.0", modelo.calcularM("-25","5","Dividir"));  
 *assertEquals*("mal", modelo.calcularM("0","0","Dividir"));  
 *assertEquals*("mal", modelo.calcularM("2","0","Dividir"));  
 *assertEquals*("mal", modelo.calcularM("2","","Dividir"));  
 *assertEquals*("mal", modelo.calcularM("asd","qwe","Dividir"));  
 *assertEquals*("mal", modelo.calcularM("...","\*\*\*","Dividir"));  
 }  
  
 */\*\*  
 \* Prueba unitaria para la operación MSuma  
 \*/* @Test  
 public void testMplus(){  
 *assertEquals*("5.0", modelo.calcularM("","5","mPLus"));  
 *assertEquals*("8.0", modelo.calcularM("3","","mPLus"));  
 }  
  
 */\*\*  
 \* Prueba unitaria para la operación MResta  
 \*/* @Test  
 public void testMrest(){  
 *assertEquals*("-5.0", modelo.calcularM("","5","mRest"));  
 *assertEquals*("-8.0", modelo.calcularM("3","","mRest"));  
 }  
}

**Resultado de las Pruebas Unitarias Ejecutadas**

