

Práctica 18: JUnit.

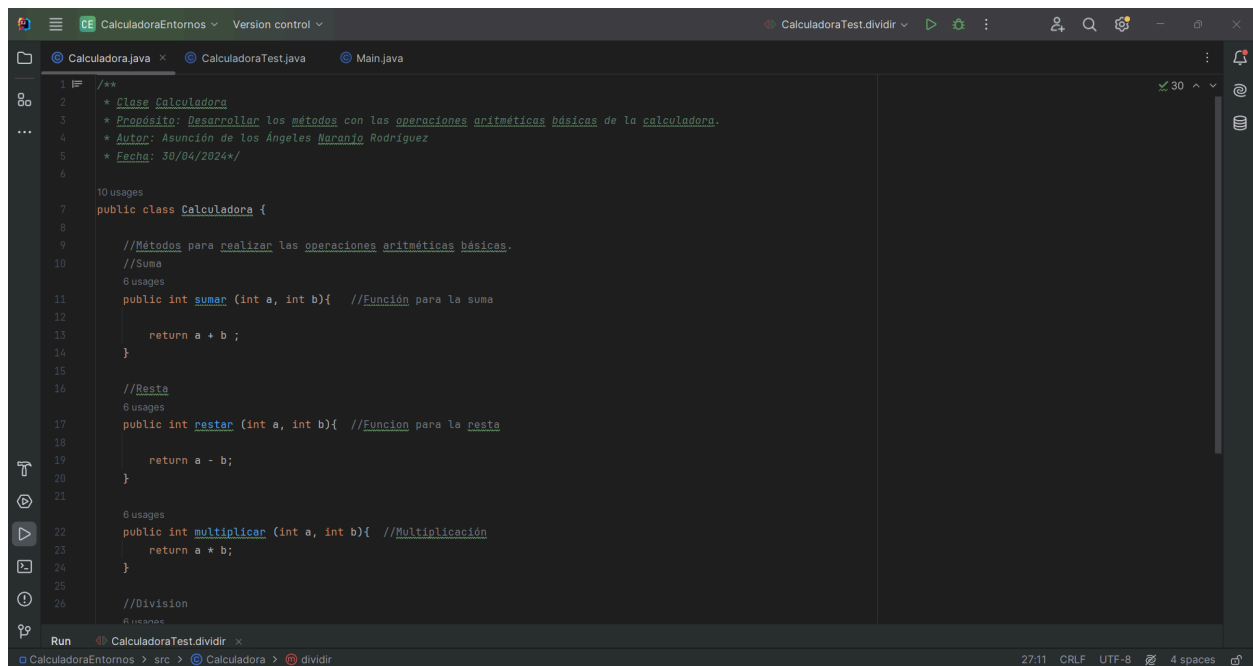
Entornos de desarrollo.

Asunción de los Ángeles Naranjo Rodríguez
1ºDAM

ACTIVIDADES

1. Se solicita programar la clase Calculadora vista en teoría de forma que, además de incluir el método suma (visto en transparencias), incluya un método para restar 2 números, otro para multiplicar 2 números y otro para dividir entre 2 números enteros.

CLASE CALCULADORA



```
1  /**
2   * Clase Calculadora
3   * Propósito: Desarrollar los métodos con las operaciones aritméticas básicas de la calculadora.
4   * Autor: Asunción de los Ángeles Naranjo Rodríguez
5   * Fecha: 30/04/2024*/
6
7  10 usages
8  public class Calculadora {
9
10     //Métodos para realizar las operaciones aritméticas básicas.
11     //Suma
12     6 usages
13     public int sumar (int a, int b){ //Función para la suma
14
15         return a + b ;
16     }
17
18     //Resta
19     6 usages
20     public int restar (int a, int b){ //Función para la resta
21
22         return a - b;
23     }
24
25     6 usages
26     public int multiplicar (int a, int b){ //Multiplicación
27         return a * b;
28     }
29
30     //Division
31     6 usages
32     public int dividir (int a, int b){ //Función para la división
33         return a / b;
34     }
35 }
```

```
25 //Division
26 6 usages
27 public int dividir (int a, int b){
28     if(b != 0){
29         return a/b;
30     } else {
31         throw new ArithmeticException("No se puede dividir entre cero");
32     }
33 }
34 }
35
```

Run CalculadoraTest.dividir x

CalculadoraEntornos > src > Calculadora > dividir 27:11 CRLF UTF-8 4 spaces

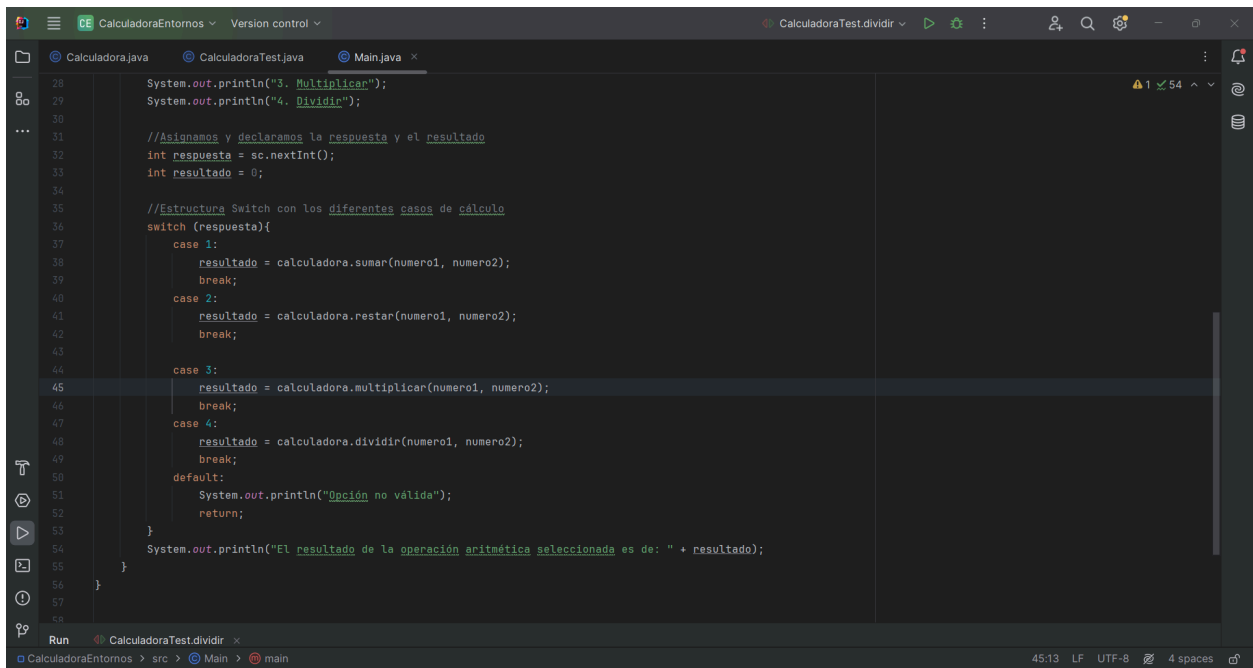
CLASE MAIN

```
CalculadoraEntornos Version control CalculadoraTest.dividir
Calculadora.java CalculadoraTest.java Main.java x
1 import java.util.Scanner;
2
3 /**
4  * Clase Main
5  * Propósito: main en el que se probarán los métodos de la calculadora.
6  * Autor: Asunción de los Angeles Maranto Rodriguez
7  * Fecha: 30/04/2024*/
8
9 public class Main {
10     public static void main(String[] args) {
11
12         Scanner sc = new Scanner(System.in); //Instanciamos Scanner.
13
14         //Solicitamos al usuario que introduzca los números con los que vamos a operar.
15         System.out.println("Introduce un número para realizar cálculos.");
16         int numero1 = sc.nextInt();
17
18         System.out.println("Introduce el otro número con que se va a operar.");
19         int numero2 = sc.nextInt();
20
21         //Creamos la calculadora
22         Calculadora calculadora = new Calculadora();
23
24         //Solicitamos al usuario que marque la operación que desea realizar
25         System.out.println("Selecciona la operación aritmética que quiera realizar");
26         System.out.println("1. Sumar");
27         System.out.println("2. Restar");
28         System.out.println("3. Multiplicar");
29         System.out.println("4. Dividir");
30
31         //Asignamos y declaramos la respuesta y el resultado

```

Run CalculadoraTest.dividir x

CalculadoraEntornos > src > Main > main 45:13 LF UTF-8 4 spaces



```
28 System.out.println("3. Multiplicar");
29 System.out.println("4. Dividir");
30
31 //Asignamos y declaramos la respuesta y el resultado
32 int respuesta = sc.nextInt();
33 int resultado = 0;
34
35 //Estructura Switch con los diferentes casos de cálculo
36 switch (respuesta){
37     case 1:
38         resultado = calculadora.sumar(numero1, numero2);
39         break;
40     case 2:
41         resultado = calculadora.restar(numero1, numero2);
42         break;
43     case 3:
44         resultado = calculadora.multiplicar(numero1, numero2);
45         break;
46     case 4:
47         resultado = calculadora.dividir(numero1, numero2);
48         break;
49     default:
50         System.out.println("Opción no válida");
51         return;
52 }
53 System.out.println("El resultado de la operación aritmética seleccionada es de: " + resultado);
54 }
55 }
56 }
57 }
58 }
```

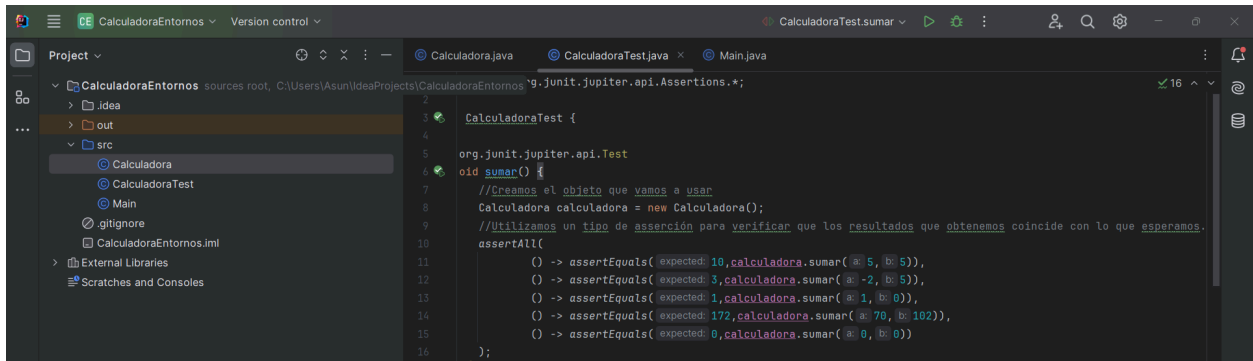
2. Deberás realizar los test con JUnit a todos los métodos indicados anteriormente.

Para realizar test unitarios, se va utilizar JUnit, este Framework, permite la automatización de pruebas en proyectos de Software. Básicamente, se realizará un test para cada tipo de operación que realiza la calculadora que hemos diseñado en el ejercicio anterior.

Para la realización de los test utilizamos un tipo de aserción, de los distintos tipos que existen, llamada assertEquals. Esta aserción se aplica a cada una de las operaciones que tenemos que probar de nuestra calculadora. Primero, se le pasa el resultado esperado, y a continuación, al método que estemos probando se le pasa por parámetros dos números que coincidirá o no con el resultado esperado.

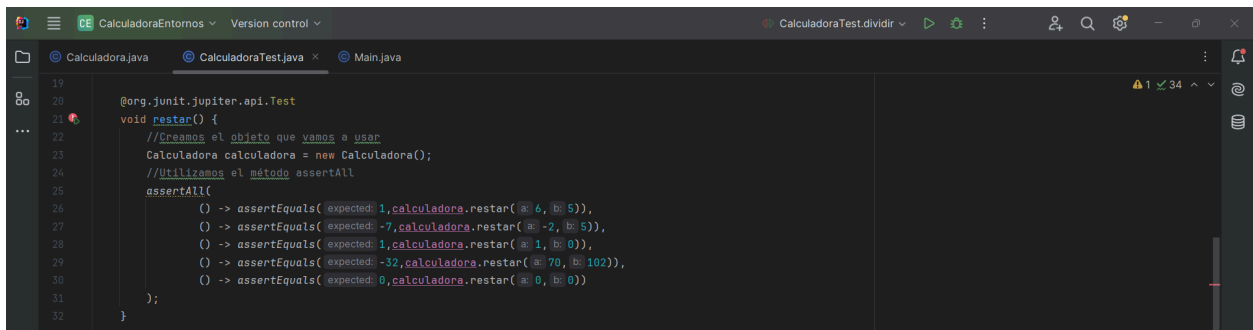
Esta inserción te permite comparar si el resultado esperado coincide con el obtenido, en el caso de que así sea, el test pasa la prueba, y, por tanto, nuestro código está bien diseñado. Nuestra calculadora funciona y cumple con las condiciones de diseño previamente definidas. Si el resultado no coincide, nuestro programa no pasa los test y nuestra calculadora no funciona correctamente.

SUMA



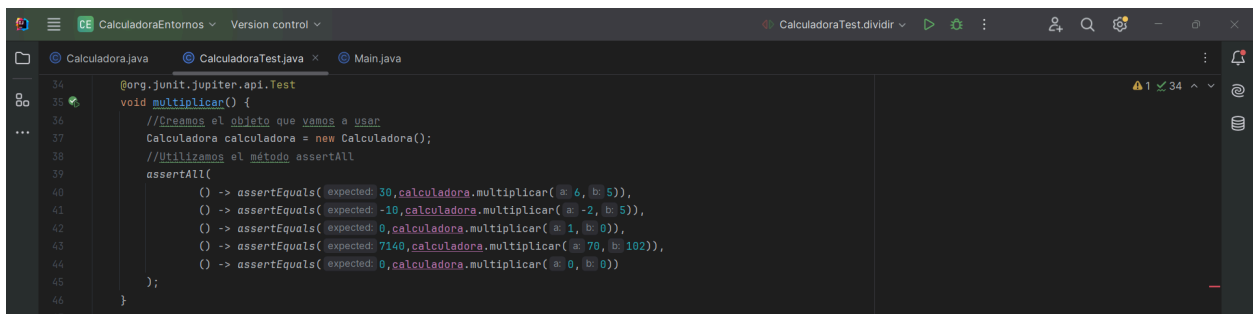
```
1 package org.entornos.calculadora;
2
3 import org.junit.jupiter.api.Test;
4
5 import org.junit.jupiter.api.Assertions.*;
6
7 public class CalculadoraTest {
8     @org.junit.jupiter.api.Test
9     void sumar() {
10         //Creamos el objeto que vamos a usar
11         Calculadora calculadora = new Calculadora();
12         //Utilizamos un tipo de aserción para verificar que los resultados que obtenemos coincide con lo que esperamos.
13         assertEquals(10, calculadora.sumar(5, 5));
14         assertEquals(3, calculadora.sumar(-2, 5));
15         assertEquals(1, calculadora.sumar(1, 0));
16         assertEquals(172, calculadora.sumar(70, 102));
17         assertEquals(0, calculadora.sumar(0, 0));
18     }
19 }
```

RESTA



```
19
20 @org.junit.jupiter.api.Test
21 void restar() {
22     //Creamos el objeto que vamos a usar
23     Calculadora calculadora = new Calculadora();
24     //Utilizamos el método assertEquals
25     assertEquals(1, calculadora.restar(6, 5));
26     assertEquals(-7, calculadora.restar(-2, 5));
27     assertEquals(1, calculadora.restar(1, 0));
28     assertEquals(-32, calculadora.restar(70, 102));
29     assertEquals(0, calculadora.restar(0, 0));
30 }
31
32 }
```

MULTIPLICACIÓN



```
34
35 @org.junit.jupiter.api.Test
36 void multiplicar() {
37     //Creamos el objeto que vamos a usar
38     Calculadora calculadora = new Calculadora();
39     //Utilizamos el método assertEquals
40     assertEquals(30, calculadora.multiplicar(6, 5));
41     assertEquals(-10, calculadora.multiplicar(-2, 5));
42     assertEquals(0, calculadora.multiplicar(1, 0));
43     assertEquals(7140, calculadora.multiplicar(70, 102));
44     assertEquals(0, calculadora.multiplicar(0, 0));
45 }
46
47 }
```

DIVISIÓN

```
48 @org.junit.jupiter.api.Test
49 void dividir() {
50     //Creamos el objeto que vamos a usar
51     Calculadora calculadora = new Calculadora();
52     //Utilizamos el método assertEquals
53     assertEquals(
54         () -> assertEquals(expected: 6, calculadora.dividir(a: 6, b: 1)),
55         () -> assertEquals(expected: 8, calculadora.dividir(a: 8, b: 20)),
56         () -> assertEquals(expected: 5, calculadora.dividir(a: 20, b: 4)),
57         () -> assertEquals(expected: -2, calculadora.dividir(a: -4, b: 2)),
58         () -> assertEquals(expected: 48, calculadora.dividir(a: 144, b: 3))
59     );
60 }
61
62 }
```

Run CalculadoraTest.dividir x

CalculadoraEntornos > src > CalculadoraTest > dividir > Lambda 58:38 CRLF UTF-8 4 spaces

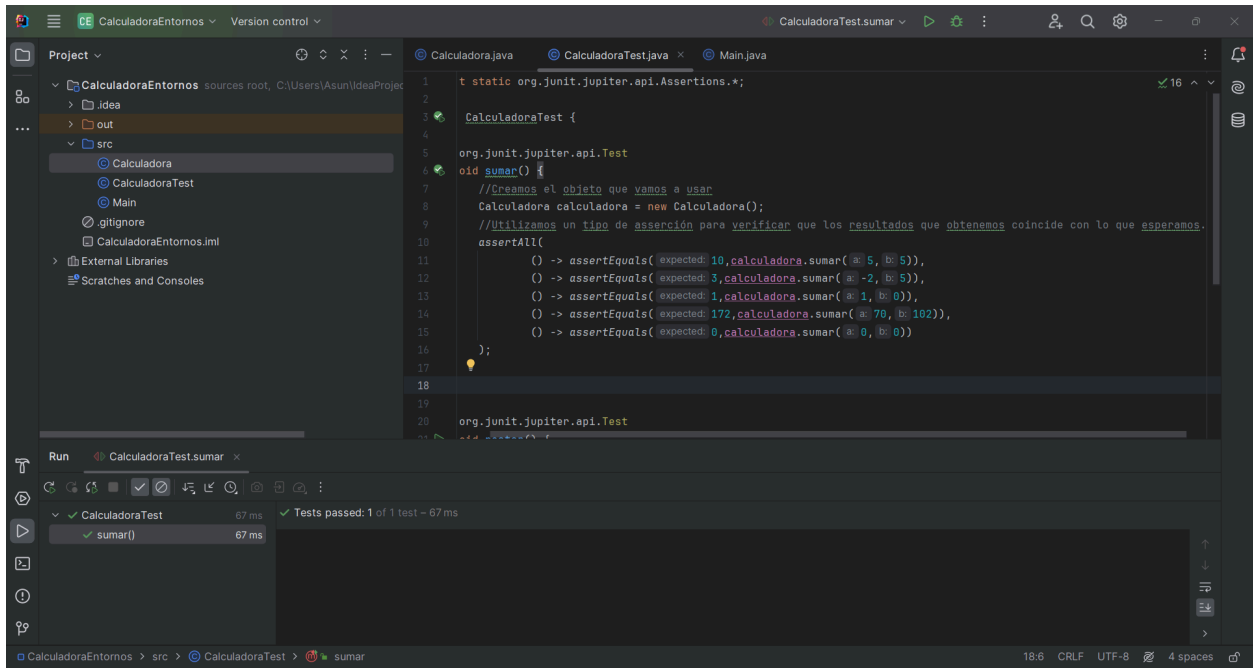
3. Comprueba que los test pasan satisfactoriamente con los resultados esperados fuerza en todos los casos con valores que den como resultado test fallido. Prueba también con dividir entre 0.

En las pruebas con resultados esperados, todos los valores que se les pasa a los métodos por parámetros coinciden con el valor esperado del test, por tanto, las distintas pruebas que se realizan darán como resultado checks verdes que señala que el método ha pasado la prueba y funciona como debería.

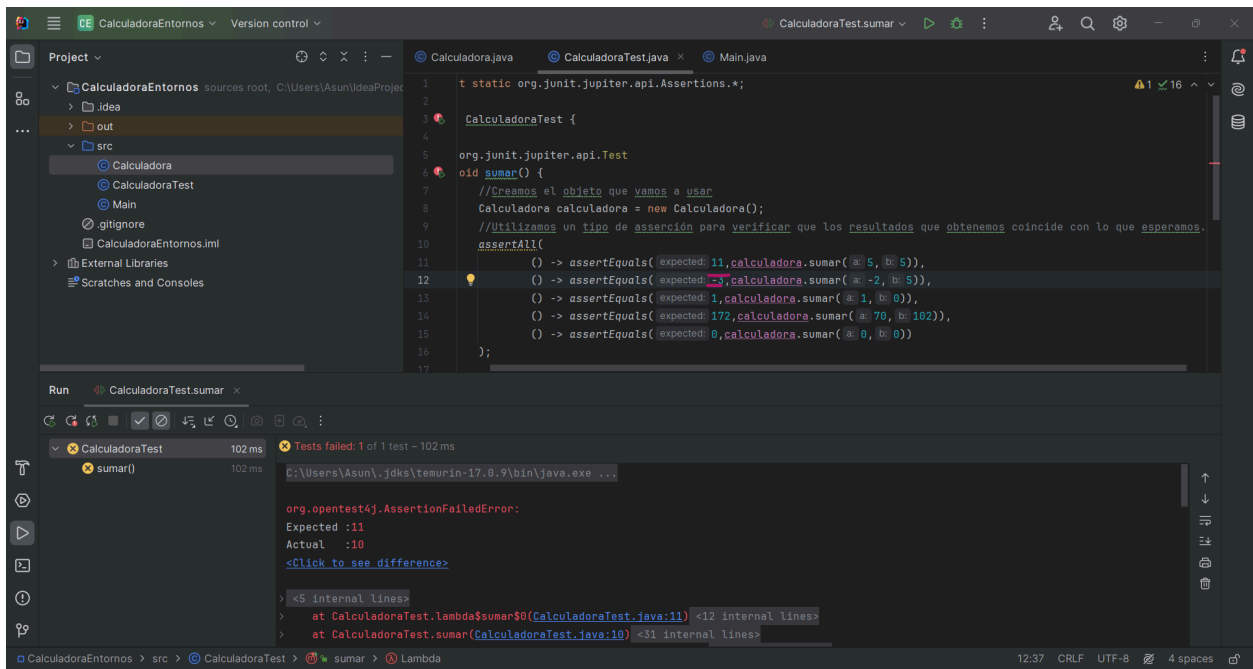
Este proceso se repite con cada una de las operaciones aritméticas que tenemos que probar de nuestra calculadora.

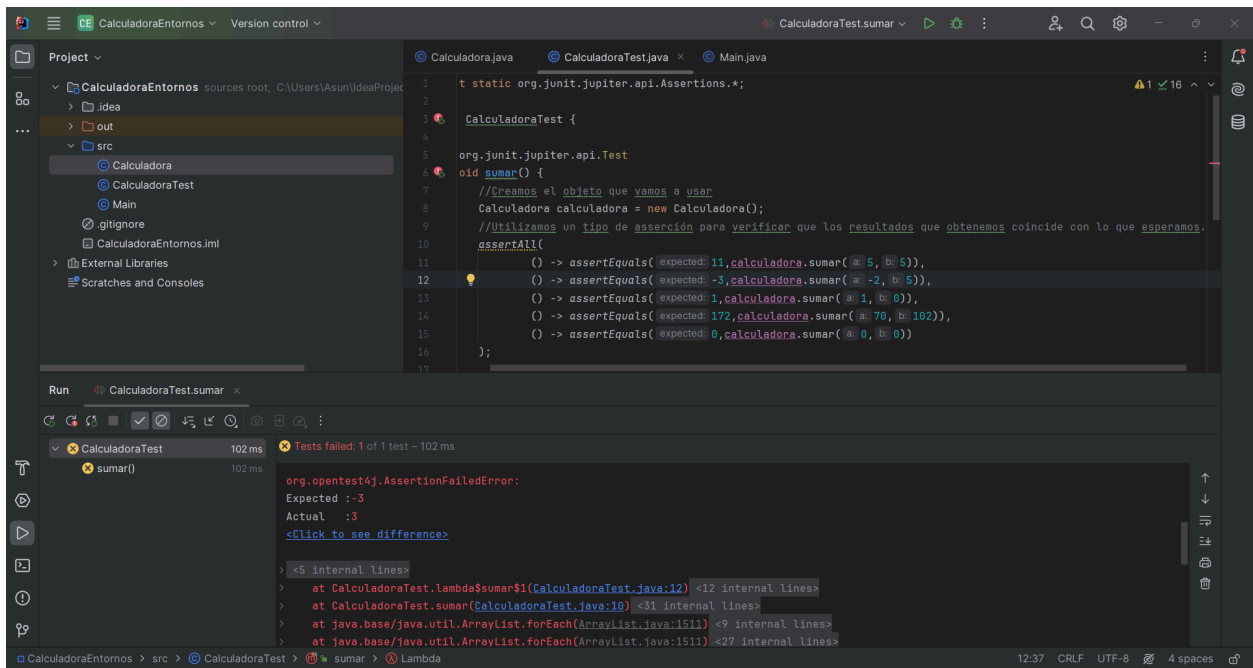
Si las pruebas se aplican a operaciones con resultados no esperados, se espera que los distintos métodos desarrollados no pasen los test a los que se les somete, y por tanto, nos devuelve como resultado errores, en los que te especifican el resultado que se debería de obtener. Por tanto, aquí los resultados esperados están cambiados y no da lo que debería al pasar esos número por parámetros, permitiendo verificar cómo funcionan los test unitarios cuando nuestro código presenta errores.

SUMA CON LOS RESULTADOS ESPERADOS

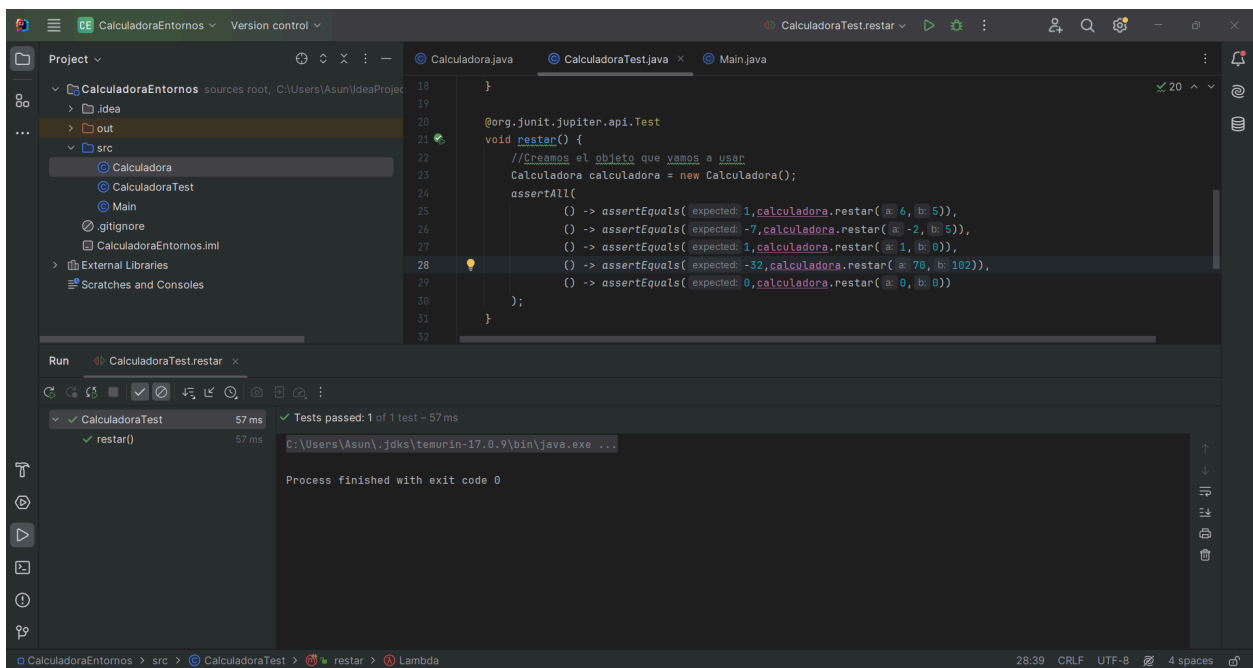


SUMA CON RESULTADOS ERRÓNEOS

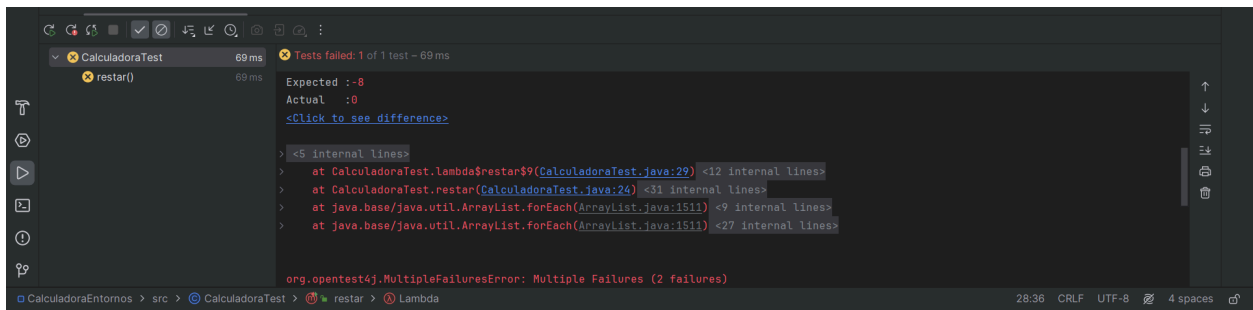
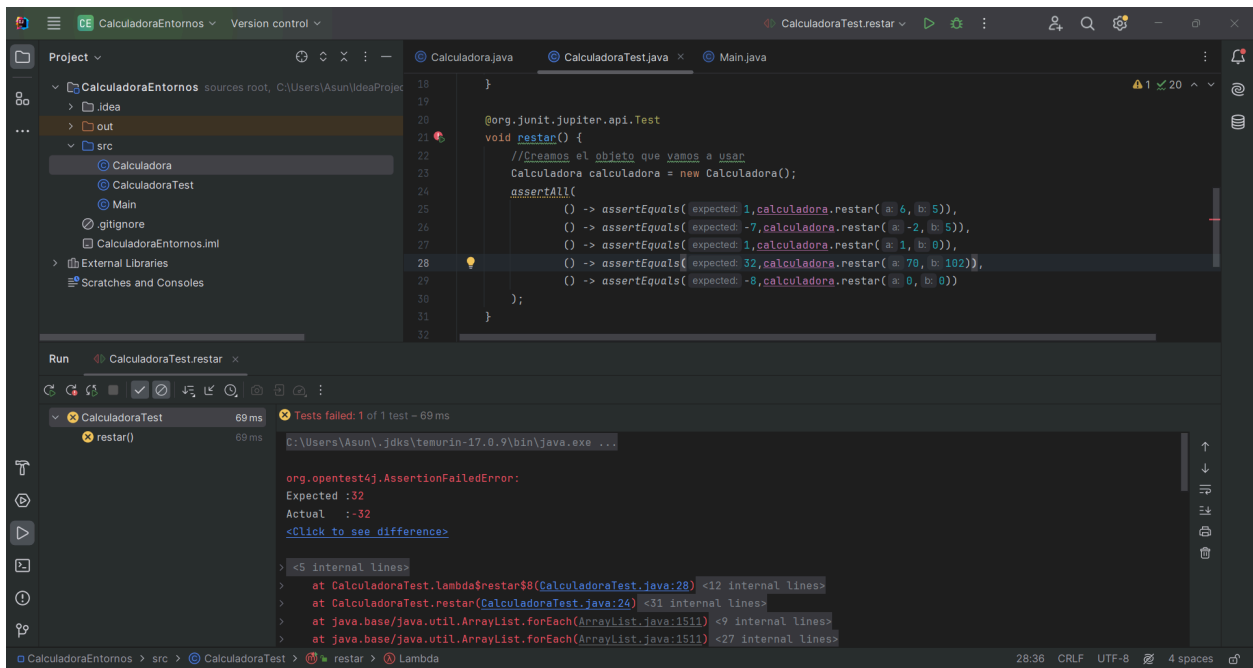




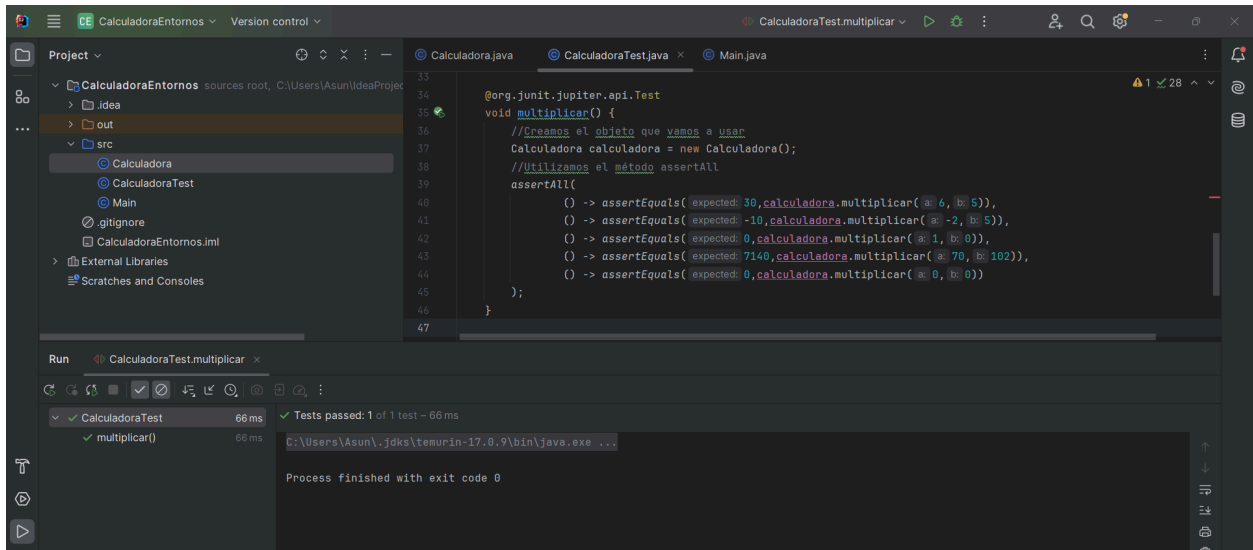
RESTA CON LOS RESULTADOS ESPERADOS



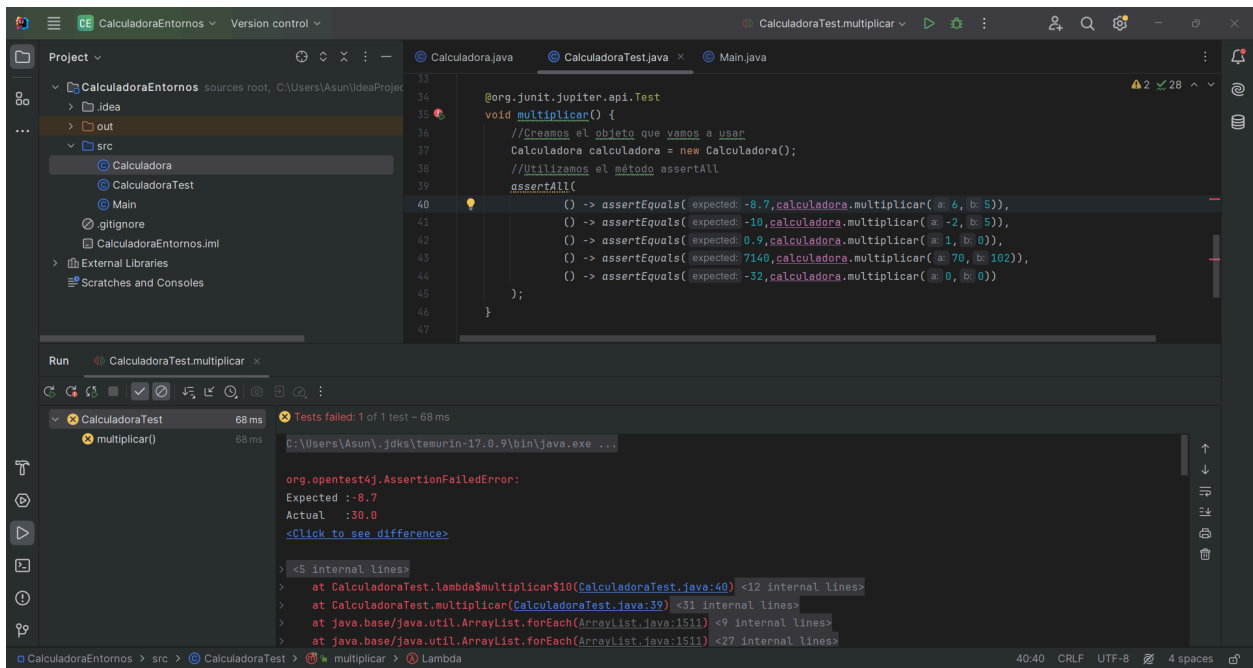
RESTA CON LOS RESULTADOS ERRÓNEOS

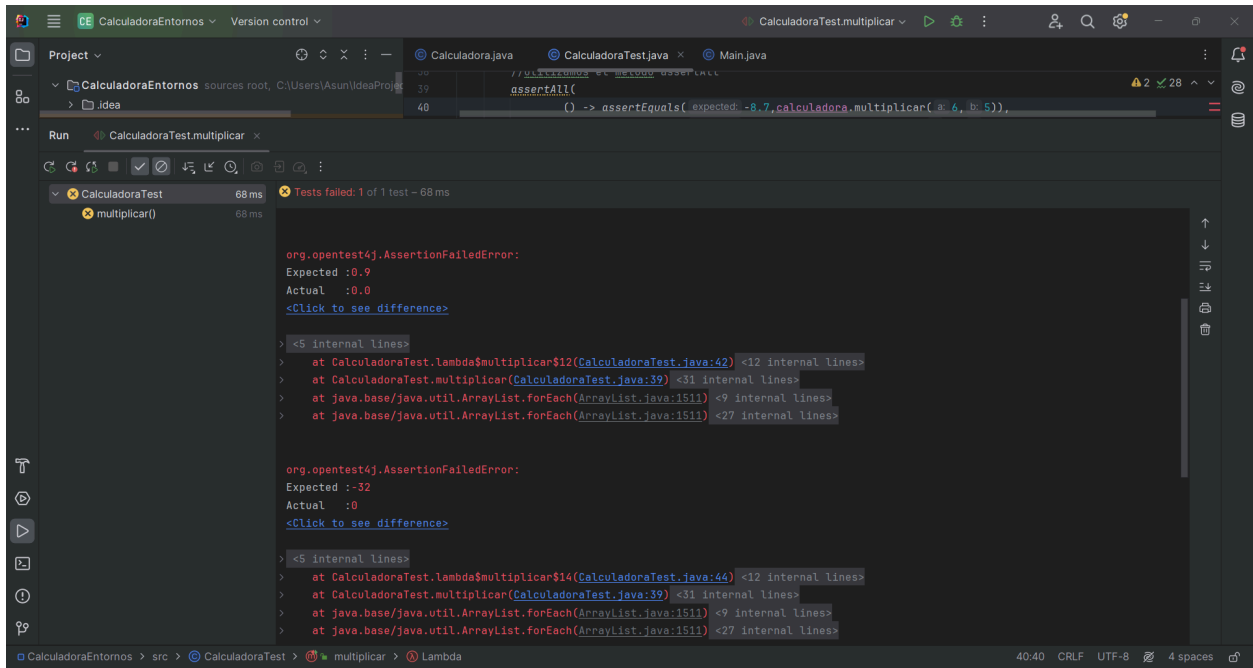


MULTIPLICACIÓN CON LOS RESULTADOS ESPERADOS

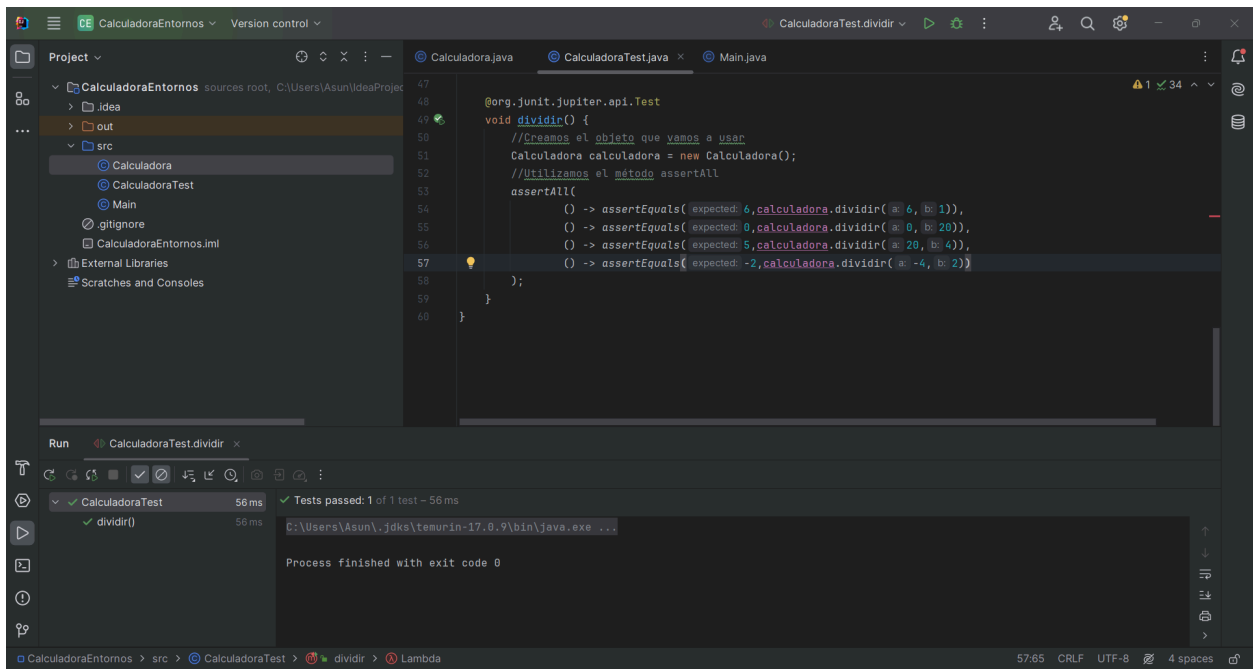


MULTIPLICACIÓN CON LOS RESULTADOS ERRÓNEOS





DIVISIÓN CON LOS RESULTADOS ESPERADOS



DIVISIÓN CON LOS RESULTADOS ERRÓNEOS

