

## Tarea para DI08.

Detalles de la tarea de esta unidad.

### Enunciado.

En BK han recibido algunas quejas de clientes sobre defectos en su software.

Ada está muy enfadada porque no se han seguido los protocolos de pruebas que la empresa tiene estandarizados. Por eso, en el nuevo proyecto que se va a desarrollar, tendrás que plantear la estrategia que asegure que los errores van a ser los mínimos posibles. Sabiendo que:

- Se trata de una aplicación desarrollada en Java
- Se van a realizar todas las pruebas vistas en la unidad.
- En principio, sólo se hará una versión por cada prueba.

Para desarrollar esta actividad necesitarás tener instalado NetBeans y JUnit. Durante el desarrollo del módulo, hemos estado trabajando con la versión 8.2 de NetBeans, el cuál ya trae incorporado JUnit. En el caso de utilizar otra versión, asegúrate de tener instalado JUnit en NetBeans.

1. Más abajo puedes descargar el [Proyecto Java](#), ábrelo con NetBeans. Observa los métodos definidos en la clase Calculando.java. Vamos a probar cada método de la clase con JUnit. Para ello, deberás de seleccionar la clase y en el menú Herramientas deberás de seleccionar la opción Create /update Tests. Nos aparecerá una ventana donde consta la clase a la que se le van a realizar las pruebas y la ubicación de las mismas. Seleccionaremos como Framework JUnit y veremos que el código de la aplicación importa automáticamente el framework JUnit. Como solución a este apartado deberás de aportar el código de la clase generado. (Calificación 1 punto)

2. Selecciona la nueva clase de pruebas que has generado. Ejecútala. Realiza una captura de la ventana Test results como solución a este apartado. (Calificación 1 punto)

3. Accede al código de la clase de pruebas y elimina las líneas:

```
// TODO review the generated test code and remove the default call to fail.
```

```
fail("The test case is a prototype.");
```

que aparece al final de cada método. Como solución a este apartado deberás de entregar el código de la clase generado. (Calificación 1 punto)

4-. Selecciona la clase de prueba y ejecútala de nuevo. Debes de corregir todos los errores asignándole valores a las variables. Al final, debes de conseguir que la ejecución de la prueba sea satisfactoria. Como solución a este apartado deberás de aportar el código de la clase de prueba una vez que ha sido modificado para conseguir que las pruebas fueran satisfactorias. (Calificación 1 punto)

5-. Implementa la planificación de las pruebas de integración, sistema y regresión.(Calificación 2 puntos)

6-. Planifica las restantes pruebas, estableciendo qué parámetros se van a analizar. (Calificación 2 puntos)

7-. Supuestas exitosas las pruebas, documenta el resultado (Calificación 2 puntos).

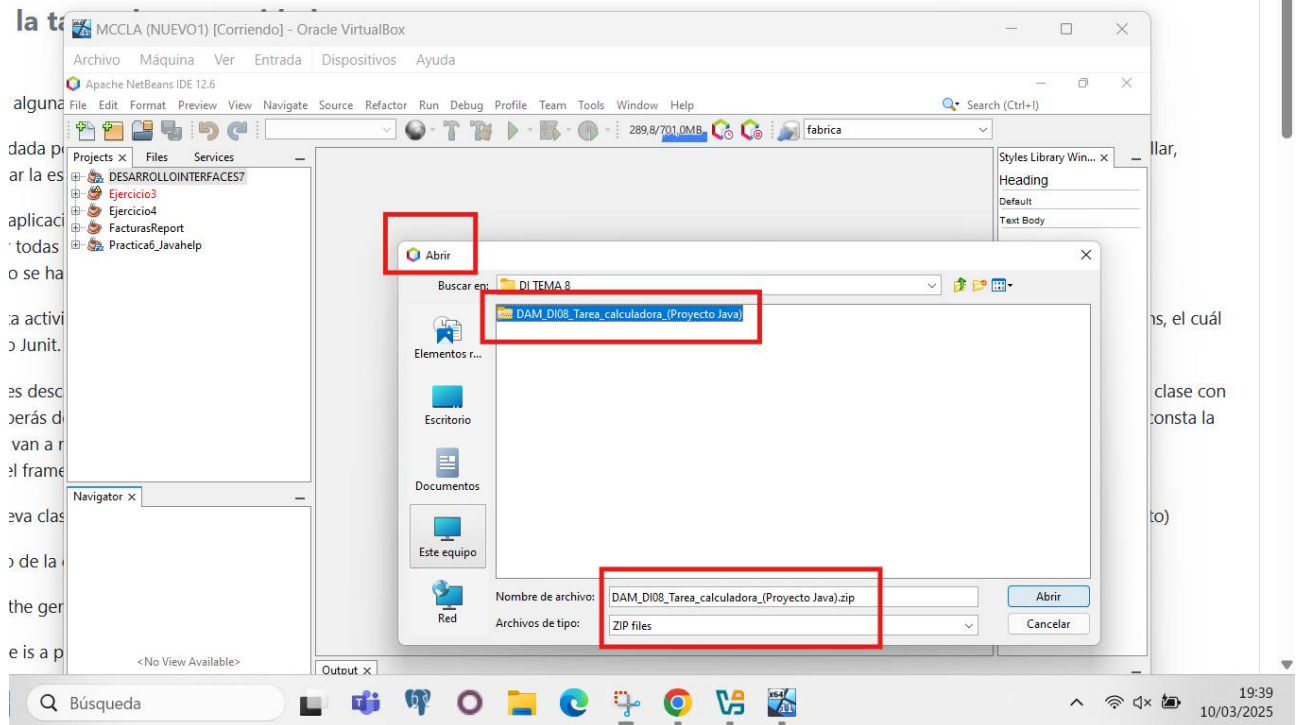
### A TENER EN CUENTA:

Una practica mas guiada que las anteriores , imprescindible hacer pruebas en cualquier aplicación.

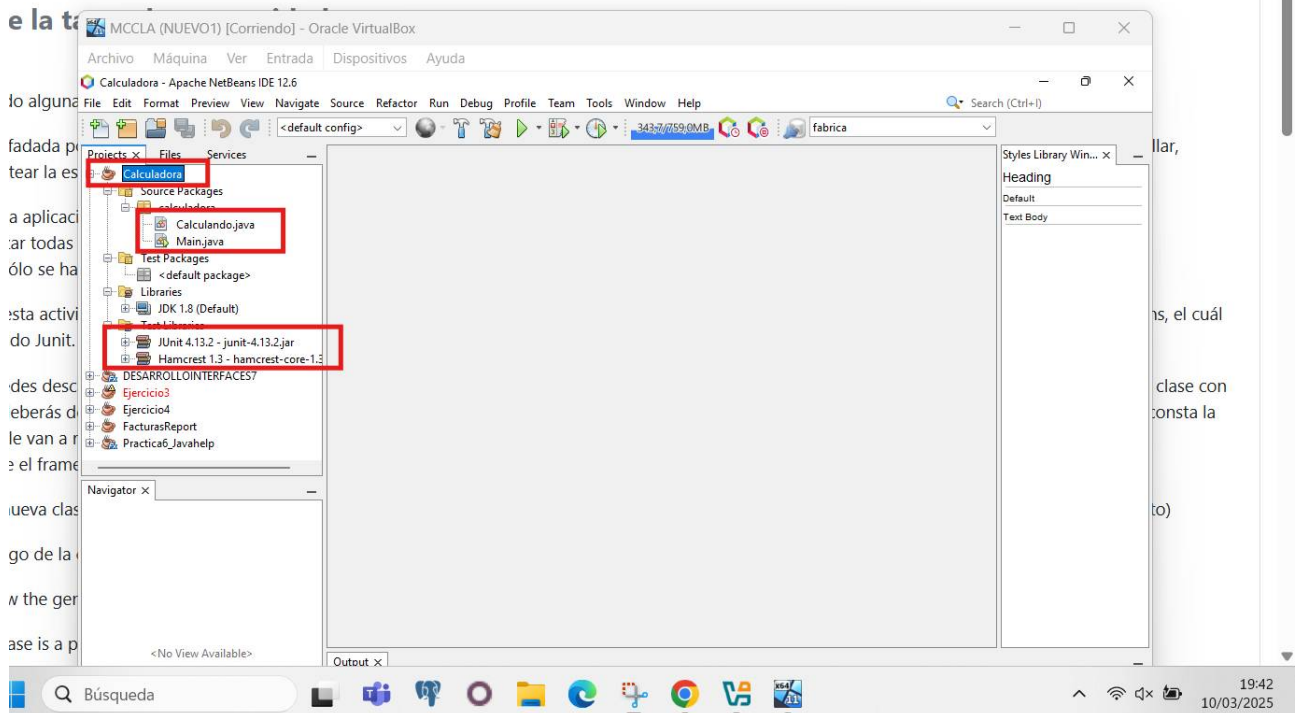
### PRACTICA:

Comenzamos con la importación del proyecto sobre el que se realizaran las diversas pruebas y test.

DIUO.

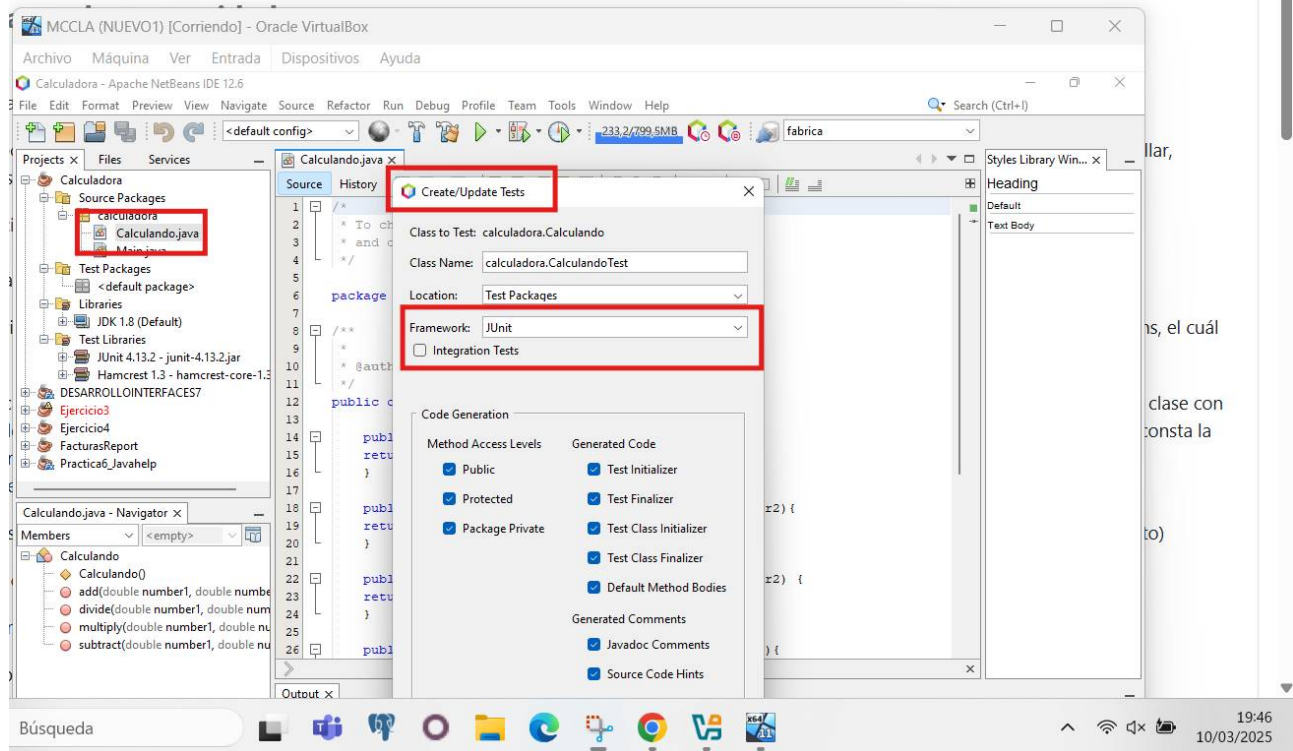


DIUO.



Creado el proyecto creamos la clase Junit para las pruebas

JO.



MCCLA (NUEVO1) [Corriendo] - Oracle VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Calculadora - Apache NetBeans IDE 12.6

File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config> 233.2/799.5MB fabrica

Projects Files Services

Calculadora

Source Packages

Calculadora

Calculando.java

Test Packages

<default package>

Libraries

JDK 1.8 (Default)

Test Libraries

JUnit 4.13.2 - junit-4.13.2.jar

Hamcrest 1.3 - hamcrest-core-1.3.jar

DESARROLLOINTERFACES7

Ejercicio3

Ejercicio4

FacturasReport

Practica6\_Javahelp

Calculando.java - Navigator x

Members

<empty>

Calculando

Calculando()

add(double number1, double number2)

divide(double number1, double number2)

multiply(double number1, double number2)

subtract(double number1, double number2)

Source History

Create/Update Tests

Class to Test: calculadora.Calculando

Class Name: calculadora.CalculandoTest

Location: Test Packages

Framework: JUnit

☐ Integration Tests

Code Generation

Method Access Levels

Public

Protected

Package Private

Generated Code

Test Initializer

Test Finalizer

Test Class Initializer

Test Class Finalizer

Default Method Bodies

Generated Comments

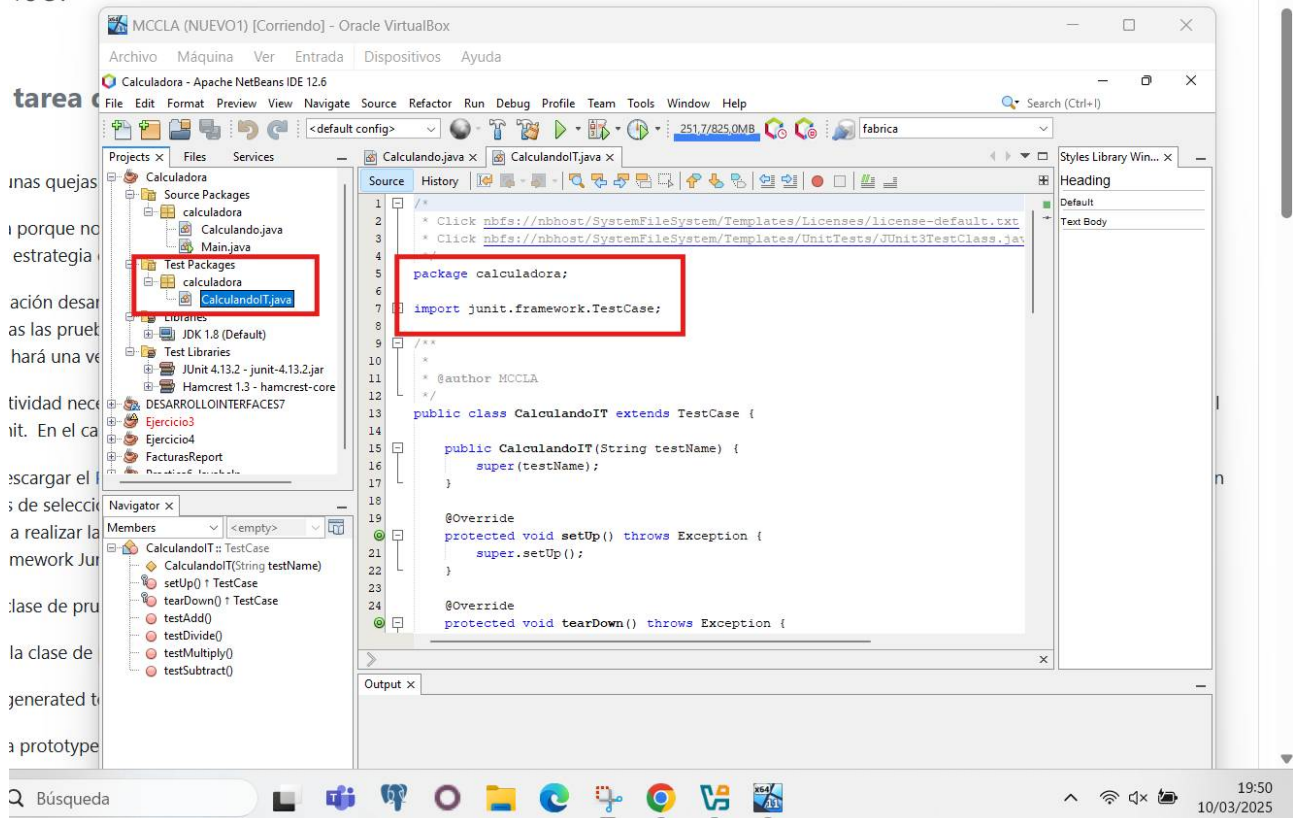
Javadoc Comments

Source Code Hints

Búsqueda

19:46 10/03/2025

UO.



MCCLA (NUEVO1) [Corriendo] - Oracle VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Calculadora - Apache NetBeans IDE 12.6

File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config> 251.7/825.0MB fabrica

Projects Files Services

Calculadora

Source Packages

calculadora

Calculando.java

Main.java

Test Packages

calculadora

CalculandoIT.java

Libraries

JDK 1.8 (Default)

Test Libraries

JUnit 4.13.2 - junit-4.13.2.jar

Hamcrest 1.3 - hamcrest-core-1.3.jar

DESARROLLOINTERFACES7

Ejercicio3

Ejercicio4

FacturasReport

Practica6\_Javahelp

CalculandoIT.java - Navigator x

Members

<empty>

CalculandoIT:: TestCase

CalculandoIT(String testName)

setUp() TestCase

tearDown() TestCase

testAdd()

testDivide()

testMultiply()

testSubtract()

Source History

CalculandoIT.java

```

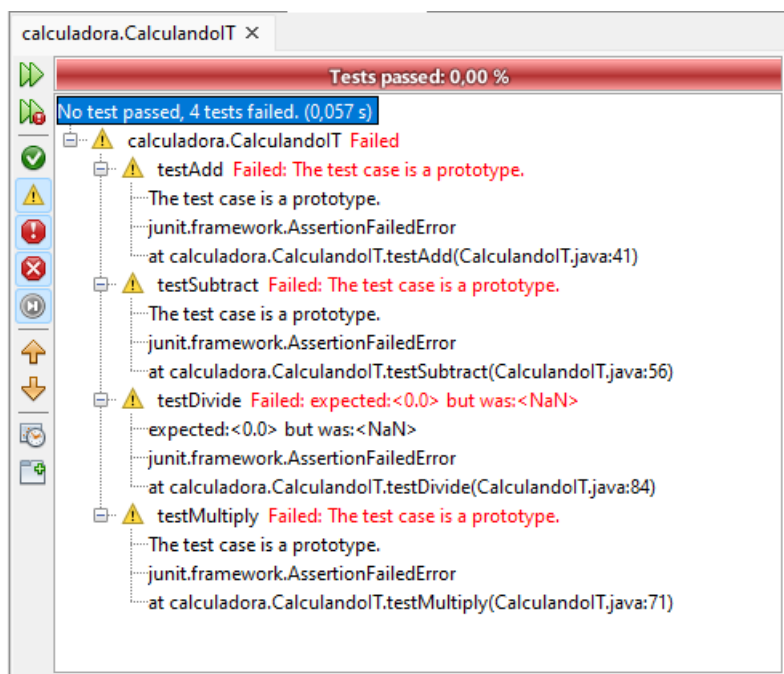
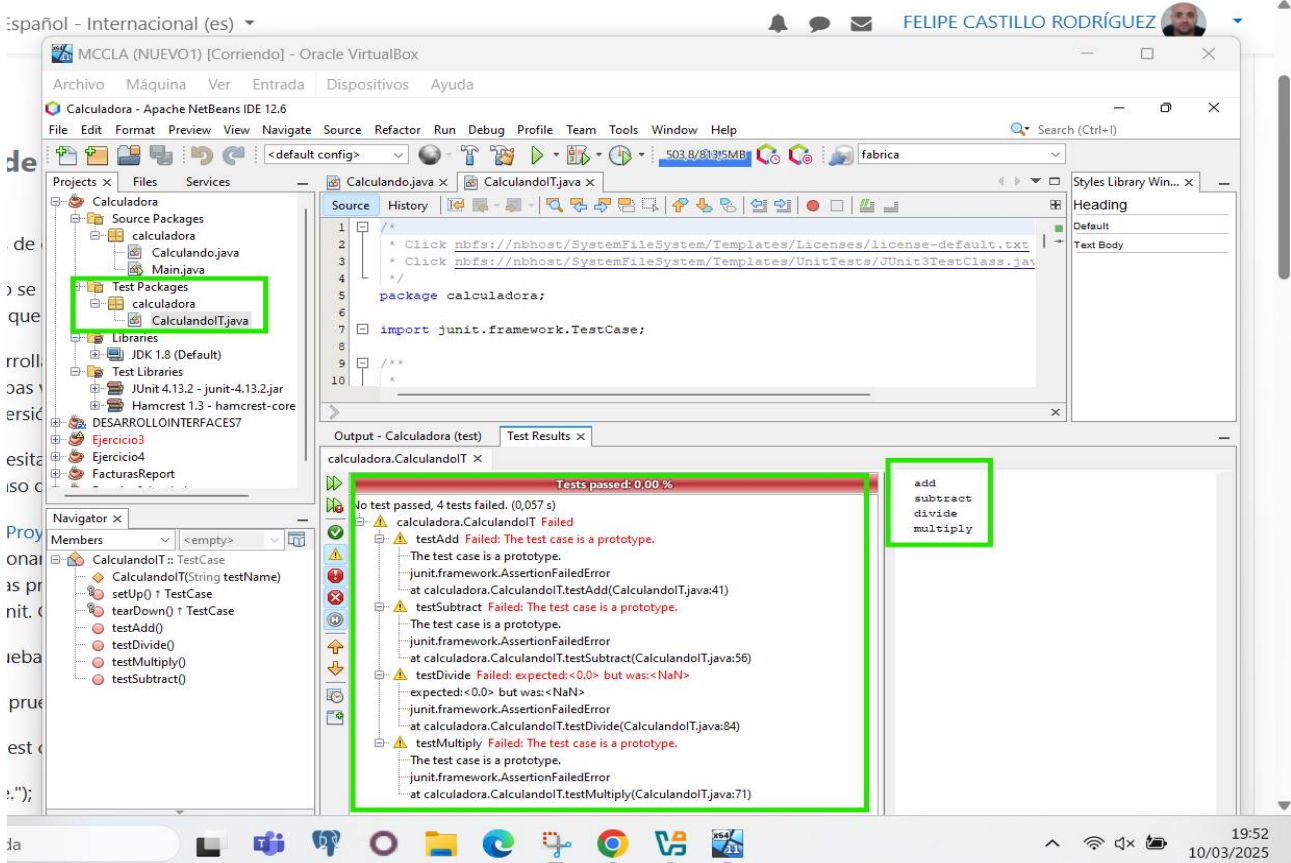
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
3  * Click nbfs://nbhost/SystemFileSystem/Templates/UnitTests/JUnit3TestClass.java
4  */
5  package calculadora;
6
7  import junit.framework.TestCase;
8
9  /**
10   *
11   * @author MCCLA
12   */
13  public class CalculandoIT extends TestCase {
14
15      public CalculandoIT(String testName) {
16          super(testName);
17      }
18
19      @Override
20      protected void setUp() throws Exception {
21          super.setUp();
22      }
23
24      @Override
25      protected void tearDown() throws Exception {
26          super.tearDown();
27      }
28  }

```

Búsqueda

19:50 10/03/2025

## Generado el Junit pasamos el test



Todos estos errores deberán ir desapareciendo con los pasos seguidos en la practica. Primer paso borrar instrucciones erróneas y segundo paso asignando valores a las variables, en la imagen de debajo marcadas las líneas que se van a borrar de cada método y adherido el resultado del test, el fallo que se ve es porque al no tener aún asignados valores la división por cero produce un error en su metodo.



calculadora.CalculandoIT x

Tests passed: 75,00 %

3 tests passed, 1 test failed. (0,064 s)

calculadora.CalculandoIT Failed

testDivide Failed: expected:<0.0> but was:<NaN>

expected:<0.0> but was:<NaN>

junit.framework.AssertionFailedError

at calculadora.CalculandoIT.testDivide(CalculandoIT.java:81)

CalculandoIT: TestCase

- CalculandoIT(String testName)
- setUp() ↑ TestCase
- tearDown() ↑ TestCase
- testAdd()
- testDivide()
- testMultiply()
- testSubtract()

double number1 = 0.0;

Calculando instance = new Calculando();

double expectedResult = 0.0;

double result = instance.subtract(number1, number2);

assertEquals(expResult, result, 0.0);

// TODO review the generated test code and remove the default call to fail("The test case is a prototype.");

fail("The test case is a prototype.");

Test of multiply method, of class Calculando.

public void testMultiply() {

System.out.println("multiply");

double number1 = 0.0;

double number2 = 0.0;

double result = instance.multiply(number1, number2);

assertEquals(expResult, result, 0.0);

}

calculadora.CalculandoIT x

Tests passed: 0.00 %

No test passed, 4 tests failed. (0,057 s)

calculadora.CalculandoIT Failed

add

subtract

divide

multiply

20:02 10/03/2025

Español - Internacional (es)

FEELPE CASTILLO RODRÍGUEZ

Calculadora - Apache NetBeans IDE 12.6

File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects x

- Calculadora
- Source Packages
  - calculadora
  - Calculando.java
  - Main.java
- Test Packages
  - calculadora
  - CalculandoIT.java
- Libraries
  - JDK 1.8 (Default)
  - Test Libraries
    - JUnit 4.13.2 - junit-4.13.2.jar
    - Hamcrest 1.3 - hamcrest-core
  - DESARROLLOINTERFACES7
  - Ejercicio3
  - Ejercicio4
  - FacturasReport

testDivide - Navigator x

Members

- CalculandoIT: TestCase
- CalculandoIT(String testName)
- setUp() ↑ TestCase
- tearDown() ↑ TestCase
- testAdd()
- testDivide()
- testMultiply()
- testSubtract()

Source History

43 /\*\*

44 \* Test of subtract method, of class Calculando.

45 \*/

46 public void testSubtract() {

47 System.out.println("subtract");

48 double number1 = 8.0;

49 double number2 = 5.0;

50 Calculando instance = new Calculando();

51 double expResult = 3.0;

52 double result = instance.subtract(number1, number2);

53 assertEquals(expResult, result, 0.0);

54 }

55 }

56 /\*\*

57 \* Test of multiply method, of class Calculando.

58 \*/

59 public void testMultiply() {

60 System.out.println("multiply");

61 double number1 = 7.0;

62 double number2 = 3.0;

63 Calculando instance = new Calculando();

64 }

65 }

calculadora.CalculandoIT x

Tests passed: 100.00 %

All 4 tests passed. (0,052 s)

add

subtract

divide

multiply

20:15 10/03/2025

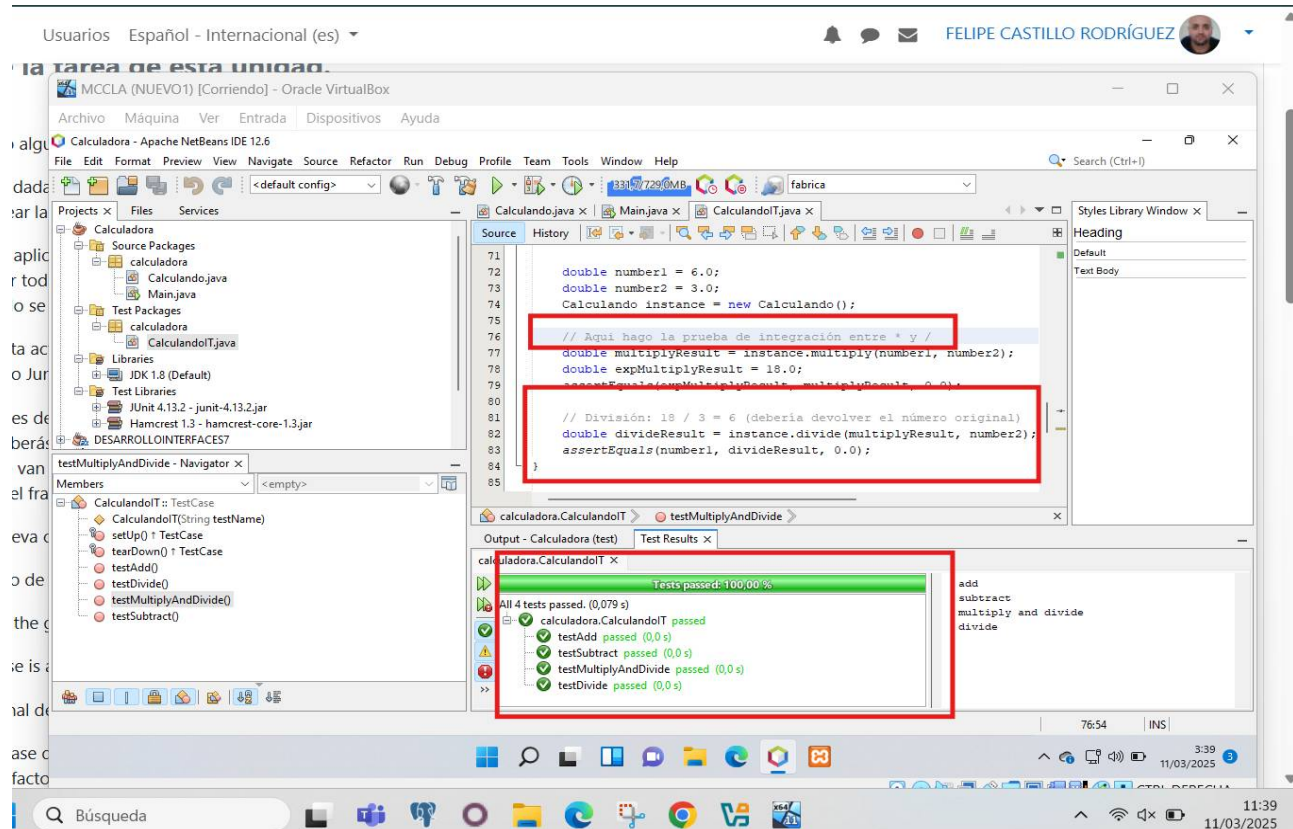
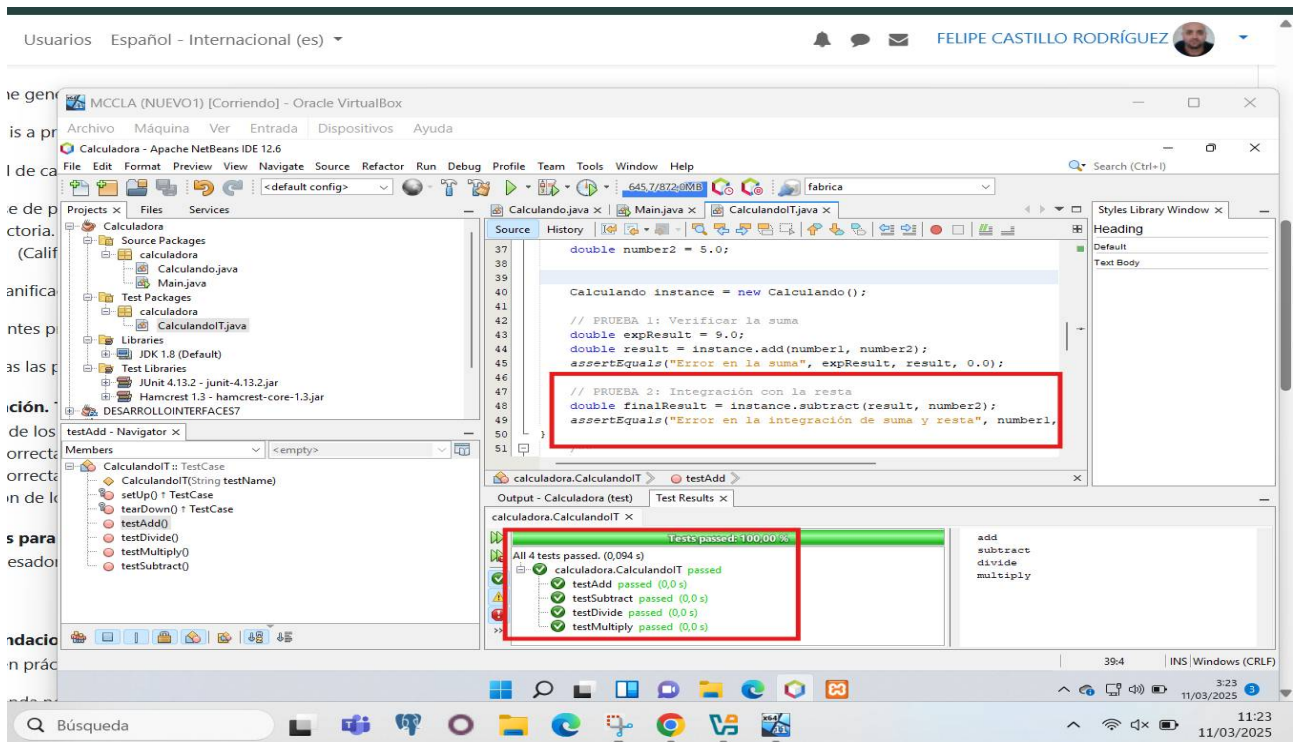
Asignando finalmente los valores a las variables los resultados revelan que todo funciona, desde aquí comenzamos con las diferentes pruebas ( En las próximas imágenes ):

**Integración :** pruebas de integración con `add()` + `subtract()` y `multiply()` + `divide()`

**Ascendente :** Se probaron métodos individuales primero y ahora probamos en conjunto.

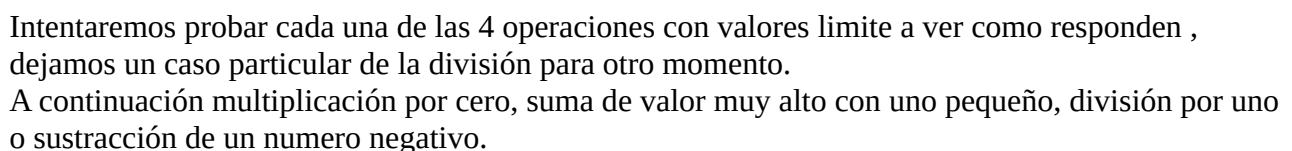
**Descendente:** Se verificará que los resultados de una operación pueden usarse en otra

**Regresión:** Con estos cambios también nos vamos a asegurar de que los cambios no rompen funcionalidad previa.



Las pruebas **Funcionales** también han sido realizadas, ya validamos que los métodos cumplen con los requisitos de la clase.

**Pruebas de capacidad y rendimiento :** pruebas con valores grandes o muchas, en las siguientes imágenes veremos por tanto algunas pruebas con valores limite.





## tarea de esta unidad.

MCCLA (NUEVO1) [Corriendo] - Oracle VirtualBox

Calculadora - Apache NetBeans IDE 12.6

File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects x Files Services

Source Packages

- calculadora
  - Calculando.java
  - Main.java
- Test Packages
  - calculadora
    - CalculandoIT.java
- Libraries
  - JDK 1.8 (Default)
  - Test Libraries
    - JUnit 4.13.2 - junit-4.13.2.jar
    - Hamcrest 1.3 - hamcrest-core-1.3.jar

DESARROLLOINTERFACES7

testMultiply - Navigator x

Members

CalculandoIT::TestCase

- CalculandoIT(String testName)
- setUp() 1 TestCase
- tearDown() 1 TestCase
- testAdd()
- testDivide()
- testMultiply()
- testSubtract()

Source

```

61 assertEquals(expResult, result, 0.0);
62
63 }
64
65 /**
66  * Test of multiply method, of class Calculando.
67  */
68
69 public void testMultiply() {
70     System.out.println("multiply");
71     double number1 = 10.0;
72     double number2 = 0.0;
73     Calculando instance = new Calculando();
74     double expResult = 0.0;
75     double result = instance.multiply(number1, number2);
76     assertEquals(expResult, result, 0.0);
77 }
    
```

Test Results x

calculadora.CalculandoIT x

Tests passed: 100.00 %

All 4 tests passed. (0,078 s)

- calculadora.CalculandoIT passed
- testAdd passed (0,016 s)
- testSubtract passed (0,0 s)
- testDivide passed (0,0 s)
- testMultiply passed (0,0 s)

Output - Calculadora (test)

add  
subtract  
divide  
multiply

73:29 INS

3:28 11/03/2025

11:28 11/03/2025

Búsqueda

## tarea de esta unidad.

MCCLA (NUEVO1) [Corriendo] - Oracle VirtualBox

Calculadora - Apache NetBeans IDE 12.6

File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects x Files Services

Source Packages

- calculadora
  - Calculando.java
  - Main.java
- Test Packages
  - calculadora
    - CalculandoIT.java
- Libraries
  - JDK 1.8 (Default)
  - Test Libraries
    - JUnit 4.13.2 - junit-4.13.2.jar
    - Hamcrest 1.3 - hamcrest-core-1.3.jar

DESARROLLOINTERFACES7

testAdd - Navigator x

Members

CalculandoIT::TestCase

- CalculandoIT(String testName)
- setUp() 1 TestCase
- tearDown() 1 TestCase
- testAdd()
- testDivide()
- testMultiply()
- testSubtract()

Source

```

34 double number1 = 999999.0;
35 double number2 = 1.0;
36
37
38
39 Calculando instance = new Calculando();
40
41
42 // PRUEBA 1: Verificar la suma
43 double expResult = 1000000.0;
44 double result = instance.add(number1, number2);
45 assertEquals("Error en la suma", expResult, result, 0.0);
46
47 // PRUEBA 2: Integración con la resta
48
    
```

Test Results x

calculadora.CalculandoIT x

Tests passed: 100.00 %

All 4 tests passed. (0,079 s)

- calculadora.CalculandoIT passed
- testAdd passed (0,0 s)
- testSubtract passed (0,0 s)
- testDivide passed (0,0 s)
- testMultiply passed (0,0 s)

Output - Calculadora (test)

add  
subtract  
divide  
multiply

50:5 INS

3:31 11/03/2025

11:31 11/03/2025

Búsqueda



Usuarios Español - Internacional (es) FELIPE CASTILLO RODRÍGUEZ

resguardar en proyecto Java, ahora con NetBeans. Observa los métodos definidos en la clase Calculando.java. Vamos a probar cada método de la clase con  
ás de seleccionar la clase y en el menú Herramientas deberás de seleccionar la opción Create JUnit4 Tests. Nos aparecerá una ventana donde consta la  
n a re MCCLA (NUEVO1) [Corriendo] - Oracle VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Calculadora - Apache NetBeans IDE 12.6

File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects x Files Services

Source

```
79 /**  
80  * Test of divide method, of class Calculando.  
81  */  
82  
83 public void testDivide() {  
84     System.out.println("divide");  
85     double number1 = 10.0;  
86     double number2 = 1.0;  
87     Calculando instance = new Calculando();  
88     double expResult = 10.0;  
89     double result = instance.divide(number1, number2);  
90     assertEquals(expResult, result, 0.0);  
91 }  
92  
93
```

testDivide - Navigator x

Members

CalculandoIT:: TestCase

- CalculandoIT(String testName)
- setUp() + TestCase
- tearDown() + TestCase
- testAdd()
- testDivide()
- testMultiply()
- testSubtract()

Output - Calculadora (test) Test Results x

calculadora.CalculandoIT x

Tests passed: 100/100 %

- All 4 tests passed. (0.141 s)
- calculadora.CalculandoIT passed
- testAdd passed (0.0 s)
- testSubtract passed (0.0 s)
- testDivide passed (0.0 s)
- testMultiply passed (0.0 s)

add  
subtract  
divide  
multiply

87:30 INS 3:26 11/03/2025 11:26 11/03/2025

español - Internacional (es) FELIPE CASTILLO RODRÍGUEZ

ejecútala de nuevo. Debes de corregir todos los errores asiánandole valores a las variables. Al final. debes de conseguir que la eiecución de  
solución MCCLA (NUEVO1) [Corriendo] - Oracle VirtualBox

1 punto) Archivo Máquina Ver Entrada Dispositivos Ayuda

Calculadora - Apache NetBeans IDE 12.6

File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects x Files Services

Source

```
52 /**  
53  * Test of subtract method, of class Calculando.  
54  */  
55  
56 public void testSubtract() {  
57     System.out.println("subtract");  
58     double number1 = 5.0;  
59     double number2 = -3.0;  
60     Calculando instance = new Calculando();  
61     double expResult = 8.0; // 5 - (-3) = 8  
62     double result = instance.subtract(number1, number2);  
63     assertEquals(expResult, result, 0.0);  
64 }  
65
```

testSubtract - Navigator x

Members

CalculandoIT:: TestCase

- CalculandoIT(String testName)
- setUp() + TestCase
- tearDown() + TestCase
- testAdd()
- testDivide()
- testMultiplyAndDivide()
- testSubtract()

Output - Calculadora (test) Test Results x

calculadora.CalculandoIT x

Tests passed: 100/100 %

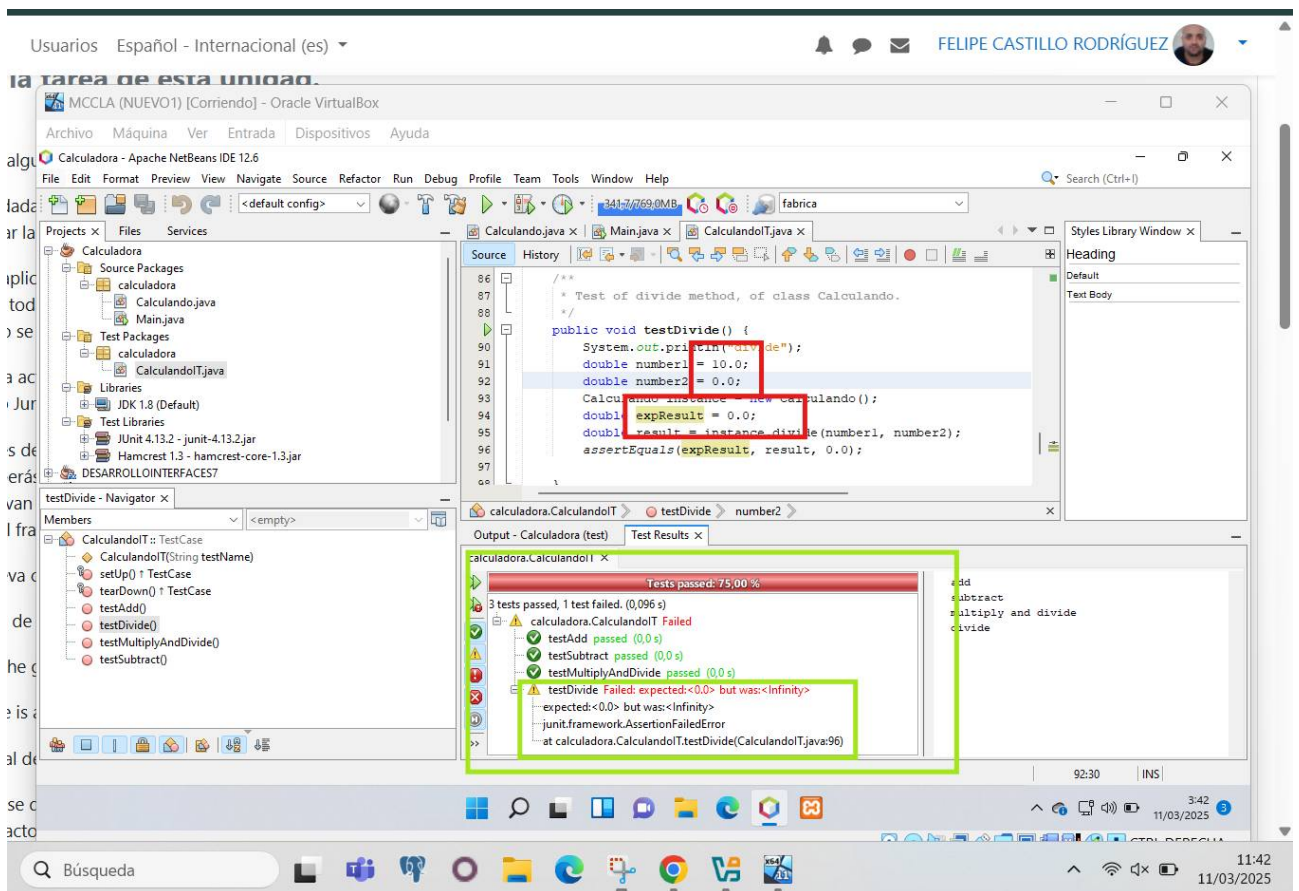
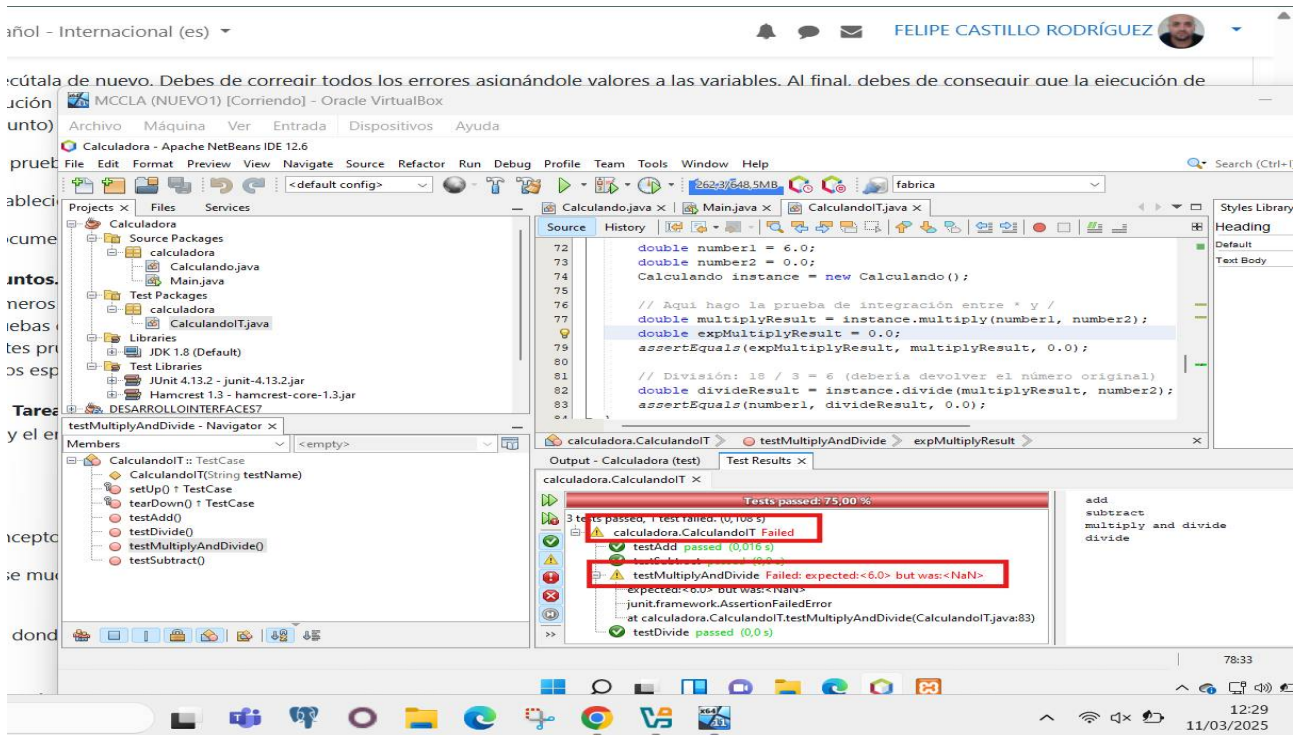
- All 4 tests passed. (0.093 s)
- calculadora.CalculandoIT passed
- testAdd passed (0.0 s)
- testSubtract passed (0.0 s)
- testMultiplyAndDivide passed (0.0 s)
- testDivide passed (0.0 s)

add  
subtract  
multiply and divide  
divide

60:45 12:35 11/03/2025

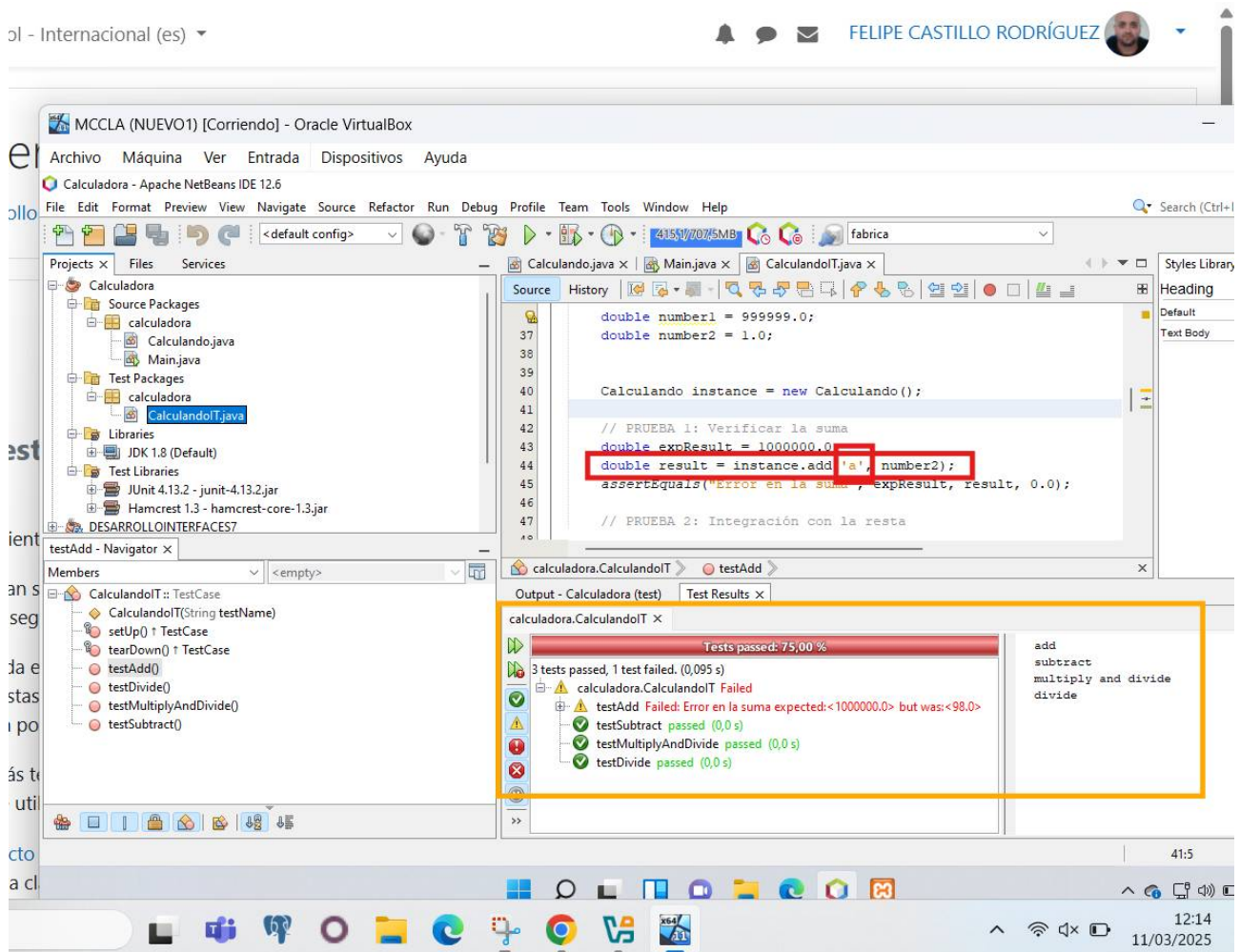
Visto el apartado anterior finalizamos con las siguientes pruebas a realizar :

**Recuperación:** prueba de manejo de errores, como **división por cero**, vemos que no existe el manejo de errores en nuestros métodos.



Por ultimo veremos si nos deja meter caracteres en nuestros métodos en lugar de números para una segunda demostración de que no hay manejo de errores y al mismo tiempo una :

**Prueba de seguridad :** ¿Cómo maneja el código entradas inesperadas o valores extremos? En este caso es un valor inesperado.



## Conclusión :

Concluidas las pruebas podemos decir que estamos ante un código adaptable y ligero con operaciones sencillas y eficaces, pero con algunos aspectos destacables, no existe la captura de errores y no existe la validación de los datos introducidos, ambos aspectos muy deseables para el código.

Una vez mas una practica muy útil , muchas gracias por su tiempo durante todo este curso para corregir estos trabajos, es genial poder aprender.



