JUnit

v.1.0

UT3 – PARTE 3

1º DAM

¿Qúe es?

- Framework de pruebas unitarias creado por Erich Gamma y Kent Beck
- Es una herramienta de código abierto.
- Posibilidad de crear informes en HTML
- Organización de las pruebas en Suites de pruebas.
- Es la herramienta de pruebas más extendida para el lenguaje Java.
- Los entornos de desarrollo para Java incorporan un plugin para JUnit.
- Herramienta para realizar pruebas unitarias automatizadas.
- Se realizan sobre una clase para probar su comportamiento de modo aislado independientemente del resto de clases de la aplicación.

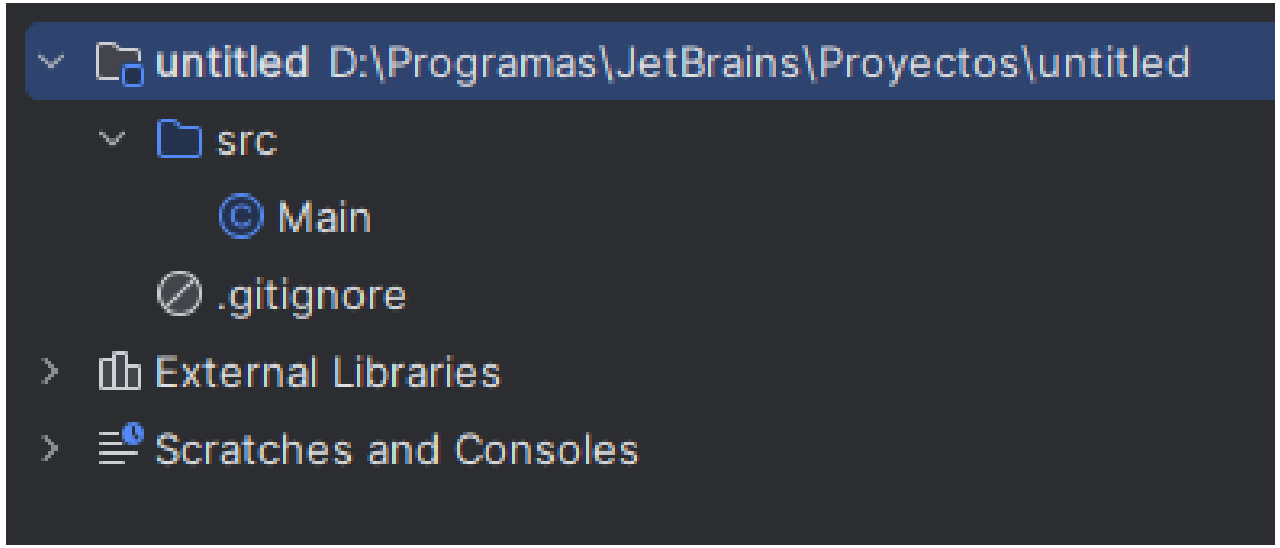
AUTOMATIZACIÓN DE LA PRUEBA

¿Qúe es?

- Los entornos de desarrollo integran frameworks que permiten automatizar las pruebas.
- Una vez diseñados los casos de prueba, pasamos a probar la aplicación.
- **JUnit**
  - Creación de la aplicación
  - En NetBeans → En el **.java** o **.class** → botón derecho → Herramientas → Create/Update Test.
  - Se crea una clase de prueba en Test Packages cuyo código se ha de modificar.
  - En IntelliJ un poco más complejo (lo veremos más adelante).

IntelliJ – Carpeta de Test

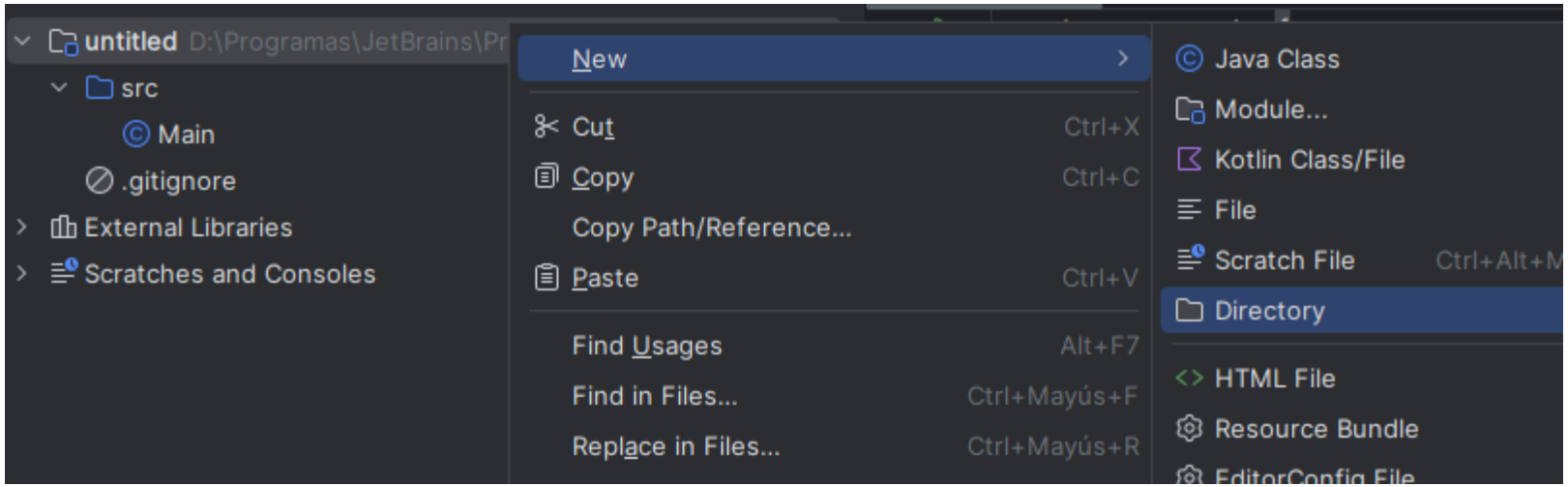
Una vez que abrimos nuestro proyecto y localizamos nuestro **fichero con el código java** (en general estará en la carpeta SRC) para poder realizar los test necesitamos tener nuestra carpeta de **test**.



IntelliJ - Carpeta “test”

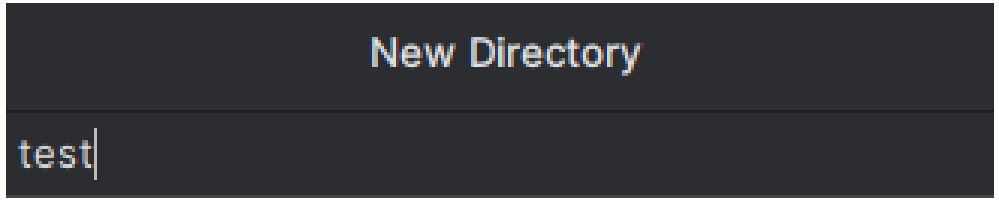
Si la carpeta **test** no existe, debemos crearla para poder realizar las pruebas.

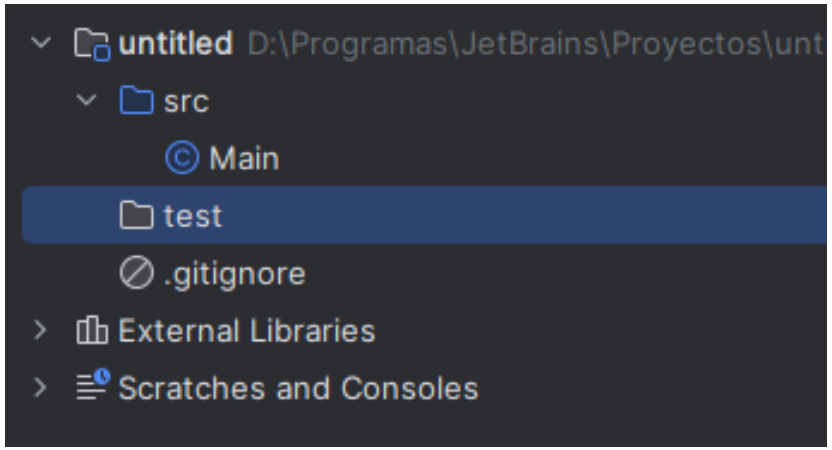
Hacemos click derecho sobre el nombre de nuestro proyecto, en nuestro ejemplo “EDUT05Ejemplo” y creamos un nuevo directorio



IntelliJ - Carpeta “test”

A este directorio le ponemos el nombre ***test***



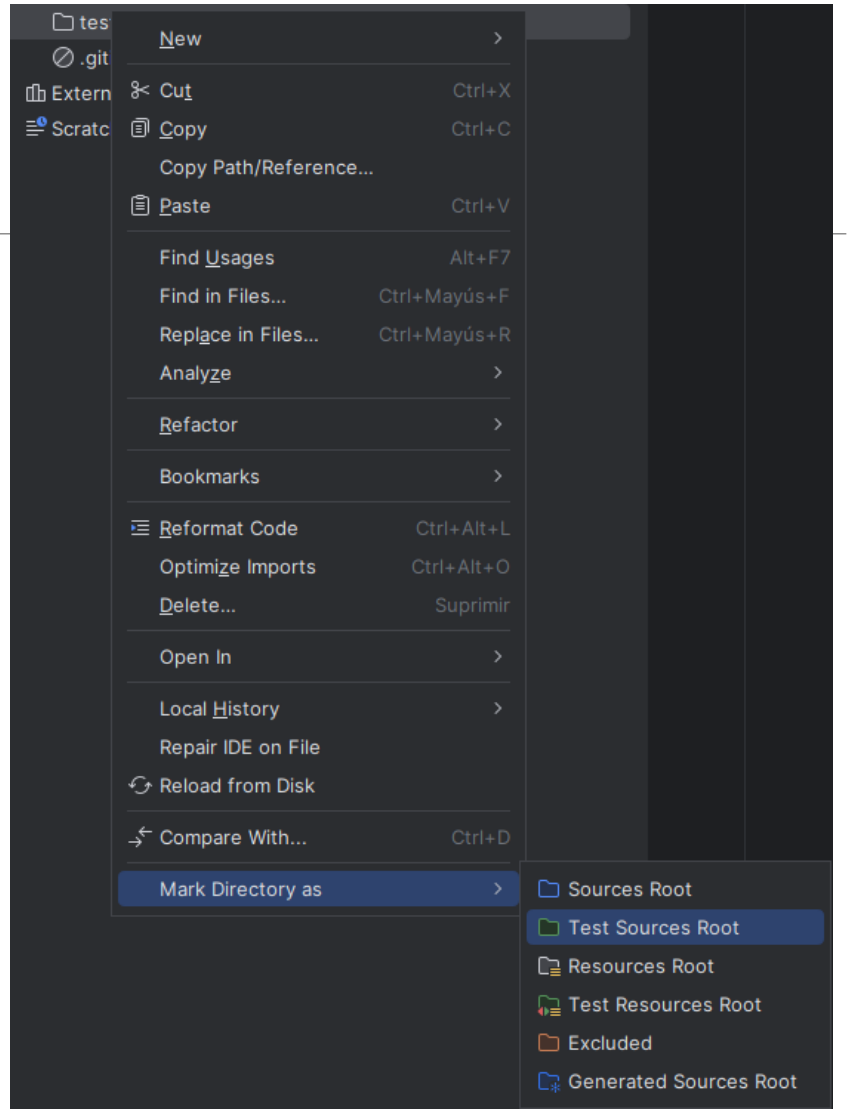


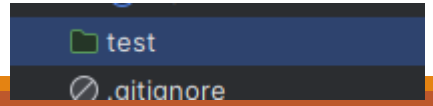
IntelliJ - Carpeta “test”

Ahora identificamos la carpeta como carpeta que tiene propiedades de test.

Hacemos clic derecho sobre ella y seleccionamos: ***Mark Directory as: Test Sources Root***

Vemos que cambia su color (a verde)

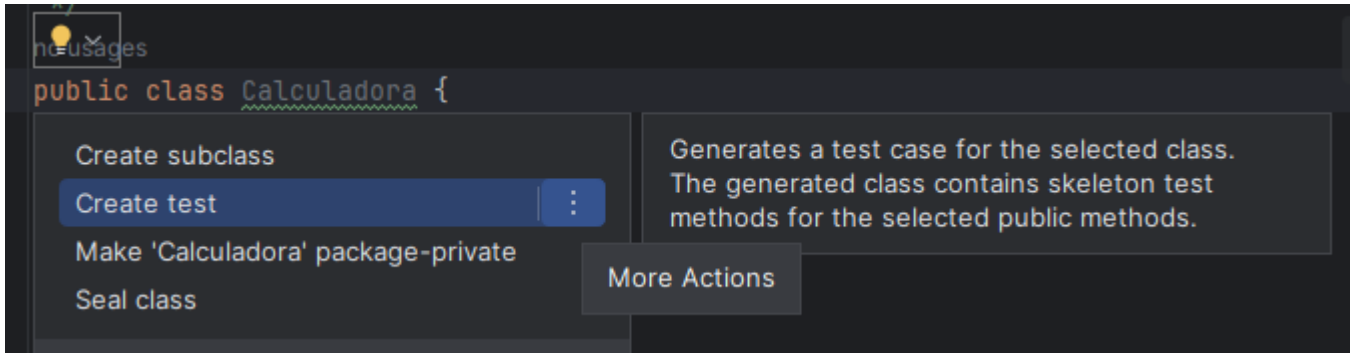




IntelliJ - Creación de Test

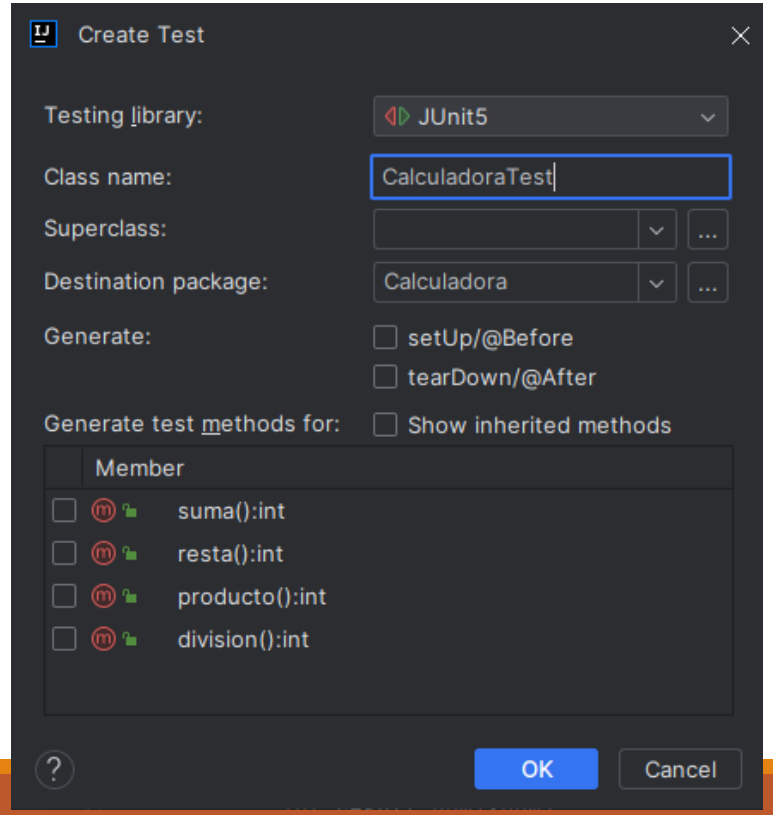
Existen varias formas en IntelliJ

La mas sencilla es ponernos sobre la clase de nuestro programa que queremos hacerle un test y en la bombilla amarilla seleccionar "***Create test***"



IntelliJ - Creación de Test

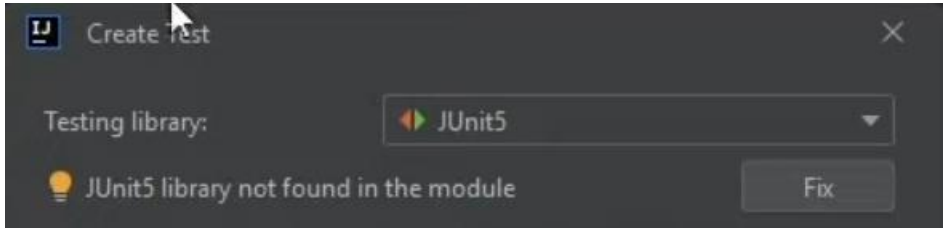
Se nos abrirá la siguiente ventana donde seleccionaremos las opciones que nos interesará para hacer el test:

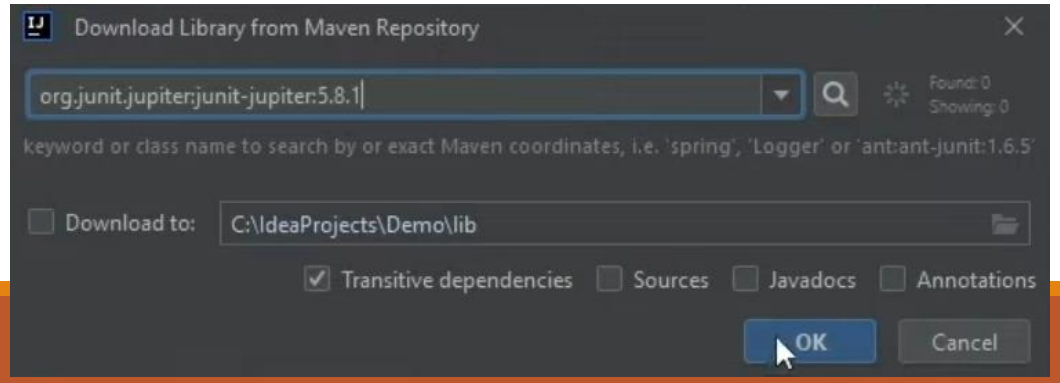


IntelliJ - Creación de Test

NOTA: Es importante que la primera vez que lo ejecutemos nos aparezca el siguiente mensaje debido a que no tenemos instalado el módulo JUnit.

Pulsamos en ***Fix*** para corregir esto y ya no tendremos que volver a hacerlo.





Creación de una CLASE TEST/PRUEBA

Estructura del proyecto

Información general.

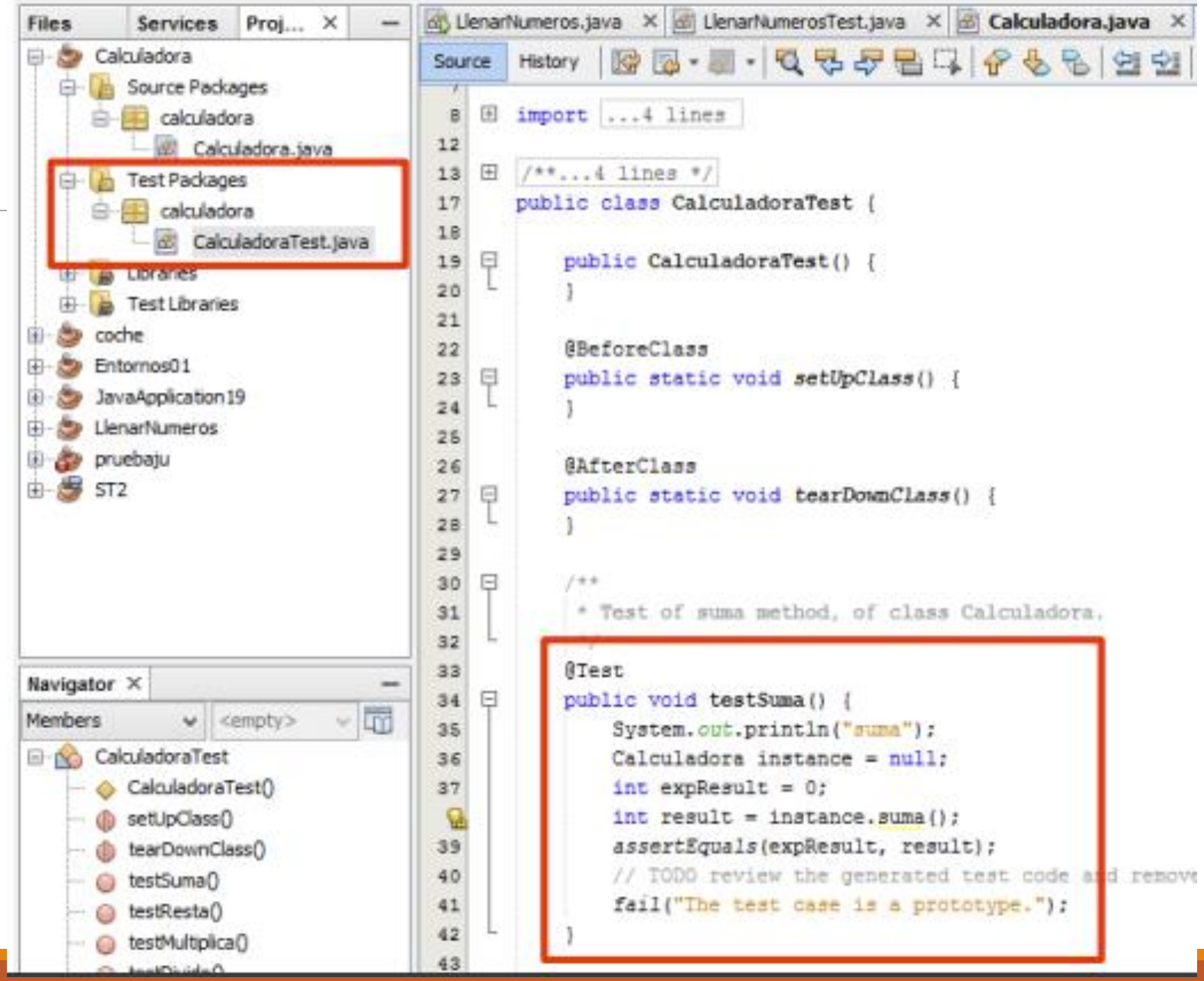
- Los métodos de prueba son **métodos públicos.**
- Ni devuelven ni reciben información.
- El nombre del método antepone la palabra ***test***.
- **@Test** indica al compilador que es un método prueba.

Etiquetas de JUnit

- **@Test**: define un método test.
- **@Before**: ejecuta el método antes de cada test.
- **@After**: ejecuta el método después de cada test.
- **@BeforeEach**: se ejecuta antes de cualquier método de prueba. Se puede utilizar para inicializar datos. Puede haber varios métodos con esta anotación JUnit5.
- **@AfterEach**: se ejecuta después de la ejecución de cada método de prueba. Se puede utilizar para limpiar datos. Puede haber varios métodos en la clase de prueba con esta anotación JUnit5.
- **@BeforeClass**: se ejecuta una vez antes de todos los test.
- **@AfterClass**: se ejecuta una vez después de todos los test.
- **@Test (expected = Exception.class):** Falla si el método no lanza una excepción.
- **@Ignore**: ignora el test.

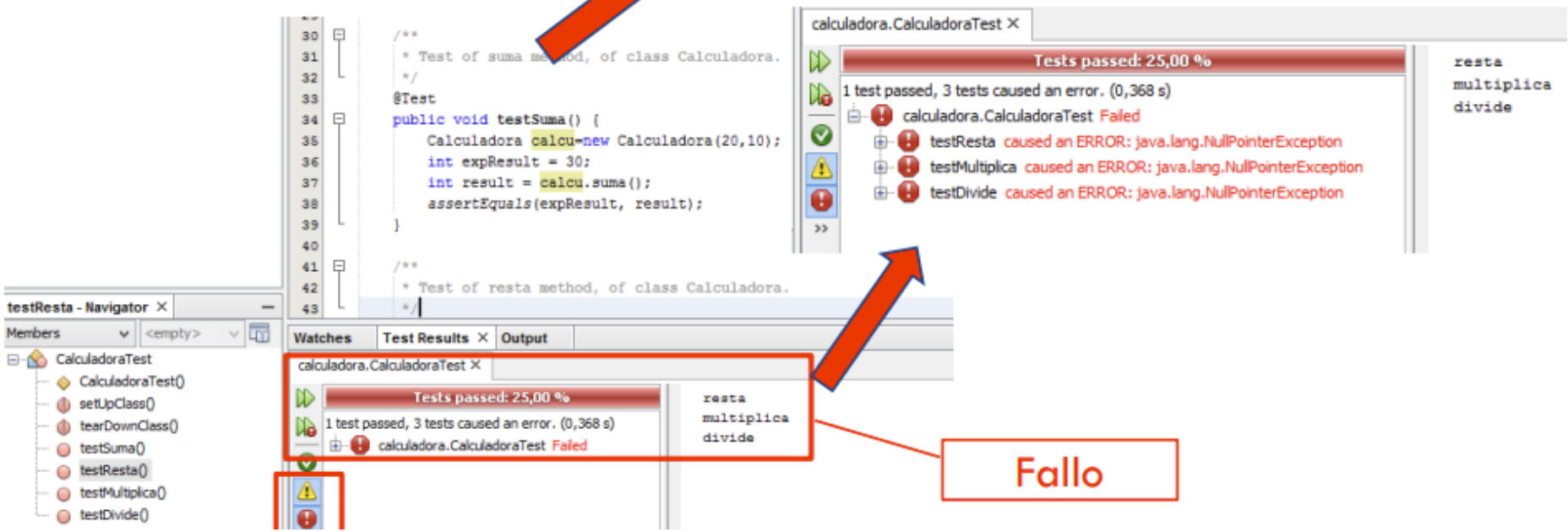
MÉTODOS	MISIÓN
assertTrue(boolean expresión) **assertTrue(String mensaje, boolean expresión)**	Comprueba que la expresión se evalúe a true. Si no e en el **String,** al producirse el error se lanzará el mens
assertFalse(boolean expresión) **assertFalse(String mensaje, boolean expresión)**	Comprueba que la expresión se evalúe a false. Si no incluye en el **String,** al producirse el error se lanzará
assertEquals(valorEsperado, valorReal) **assertEquals(String mensaje, valorEsperado, valorReal)**	Comprueba que el **valorEsperado** es igual al **valorRe** iguales y se incluye el String, se lanzará el mensaje. P diferentes tipos.
assertNull(Object objeto) **assertNull(String mensaje, Object objeto)**	Comprueba que el objeto sea **null.** Si no es **null** y se producirse el error se lanzará el **String**
assertNotNull(Object objeto) **assertNotNull(String mensaje, Object objeto)**	Comprueba que el objeto no sea **null.** Si es **null** y se producirse el error se lanzará el **String**
assertSame(Object objetoEsperado, Object objetoReal) **assertSame(String mensaje, Object objetoEsperado, Object objetoReal)**	Comprueba que el **objetoEsperado** y **objetoReal** sea Si no son el mismo y se incluye el String, se lanzará el
assertNotSame(Object objetoEsperado, Object objetoReal) **assertNotSame(String mensaje, Object objetoEsperado, Object objetoReal)**	Comprueba que el **objetoEsperado** y **objetoReal** no s objeto. Si son el mismo y se incluye el **String,** se lanza
fail() **fail(String mensaje)**	Hace que la prueba falle. Si se incluye un **String,** la p lanzando el mensaje

Ejemplo: Calculadora



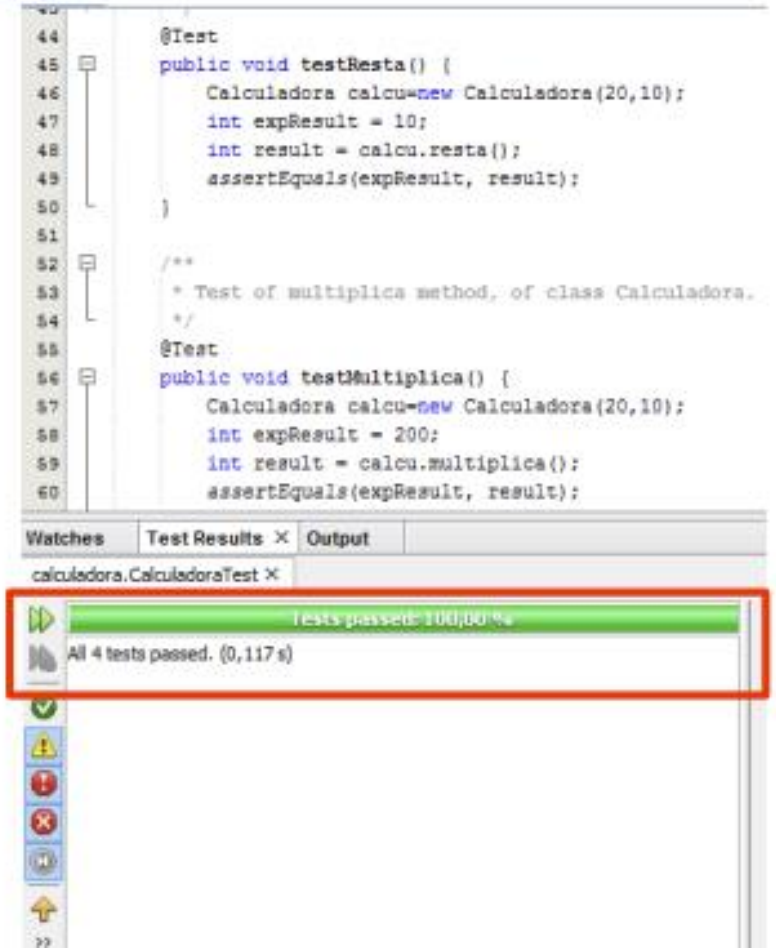
Ejemplo: Calculadora

```
@Test
public void testSuma() {
    Calculadora calcu=new Calculadora(20,10);
    int expResult = 30;
    int result = calcu.suma();
    assertEquals(expResult, result);
}
```



Ejemplo: Calculadora

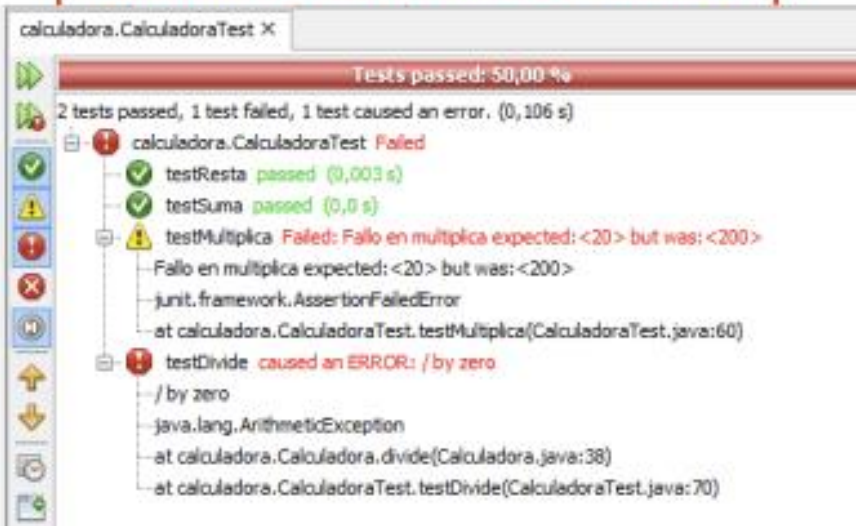
```java
@Test
public void testResta() {
    Calculadora calcu=new Calculadora(20,10);
    int expResult = 10;
    int result = calcu.resta();
    assertEquals(expResult, result);
}

/**
 * Test of multiplica method, of class Calculadora.
 */
@Test
public void testMultiplica() {
    Calculadora calcu=new Calculadora(20,10);
    int expResult = 200;
    int result = calcu.multiplica();
    assertEquals(expResult, result);
}

/**
 * Test of divide method, of class Calculadora.
 */
@Test
public void testDivide() {
    Calculadora calcu=new Calculadora(20,10);
    int expResult = 2;
    int result = calcu.divide();
    assertEquals(expResult, result);
}
```



Fallo vs error

Fallo → modifica el método para que, en *multiplica*(), el resultado esperado no coincida con el real y añade un mensaje de aviso.

Error → asignamos 0 al segundo parámetro para el método *divide*()



Ejemplo Calculadora: Modifica y Completa los métodos

- Modifica el **divide()** para que pase la prueba.
- Modifica el método **resta()** y añade **resta2()** y **divide2()**. Crea después los test para probar los tres métodos.
- Utiliza los métodos **assertTrue, assertFalse, assertNull, assertNotNull** o **assertEquals**, según convenga.

```
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;
}
```