

JUnit

v.1.0

UT3 – PARTE 3

1º DAM

¿Qué 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.

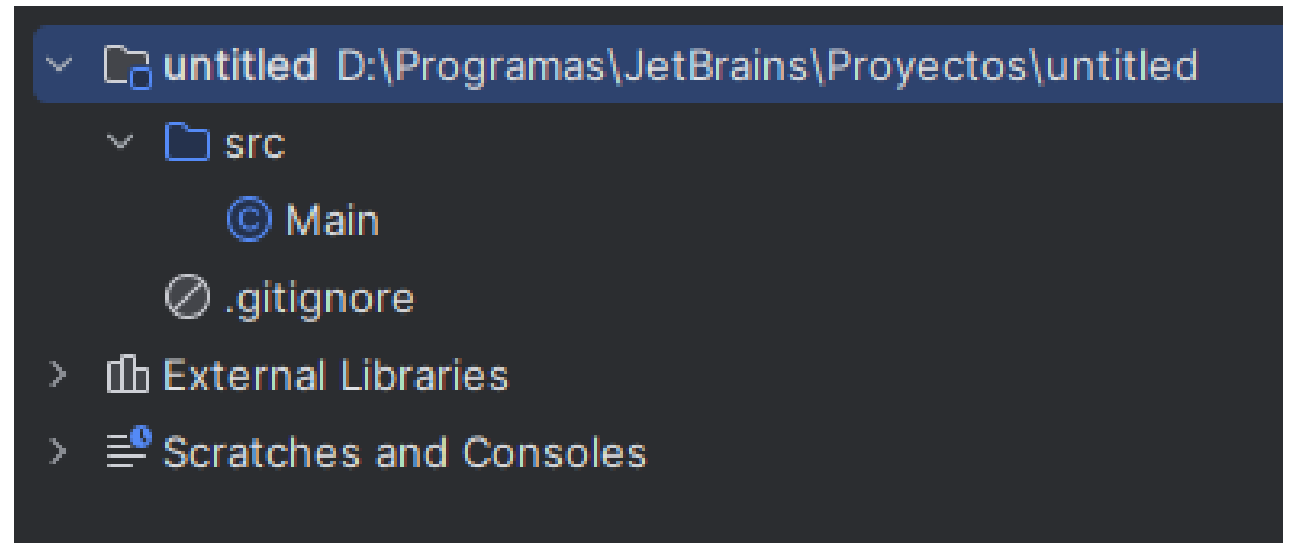
AUTOMATIZACIÓN DE LA PRUEBA

¿Qué 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

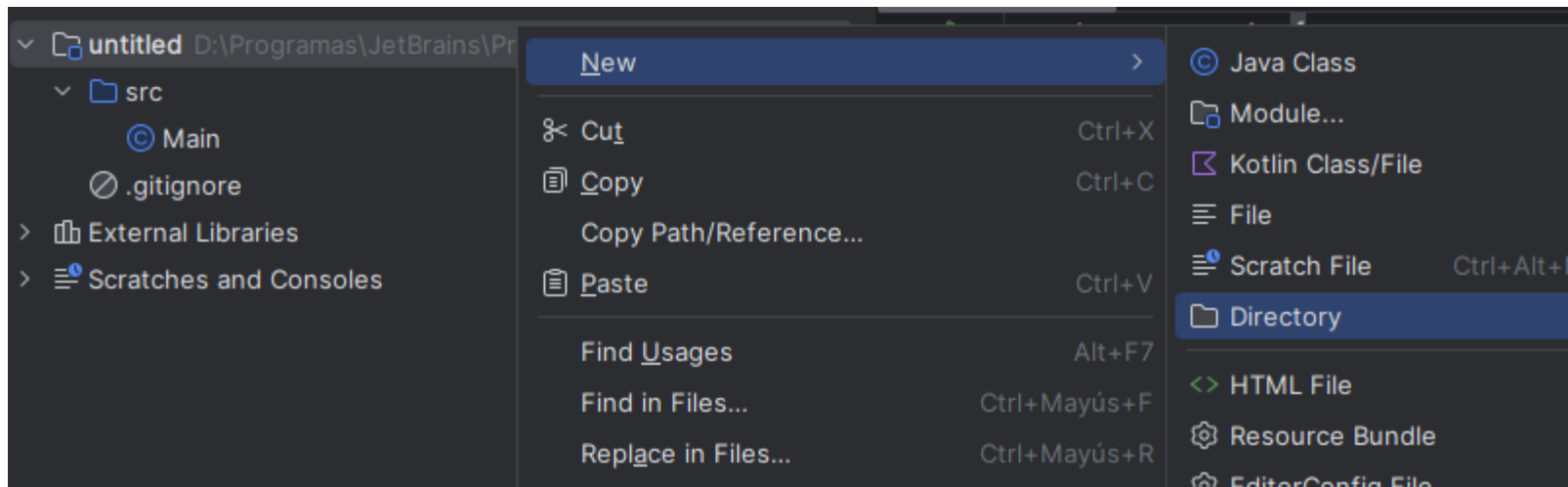
Una vez que abrimos nuestro proyecto y localizamos nuestro **fichero con el código java** (en general estará en la carpeta SRC) para poder realizar los test necesitamos tener nuestra carpeta de **test**.



IntelliJ - Carpeta “test”

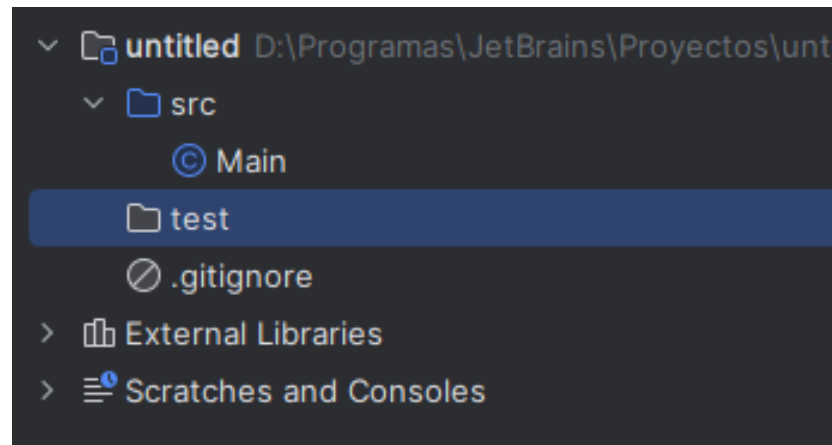
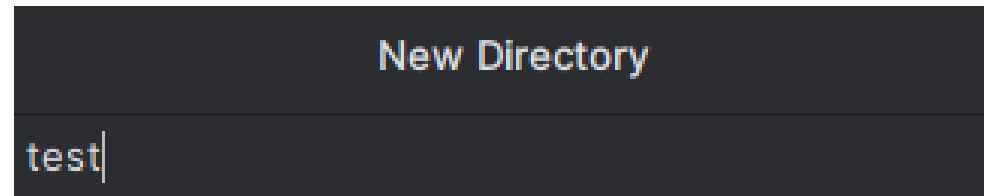
Si la carpeta **test** no existe, debemos crearla para poder realizar las pruebas.

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*

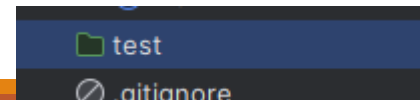
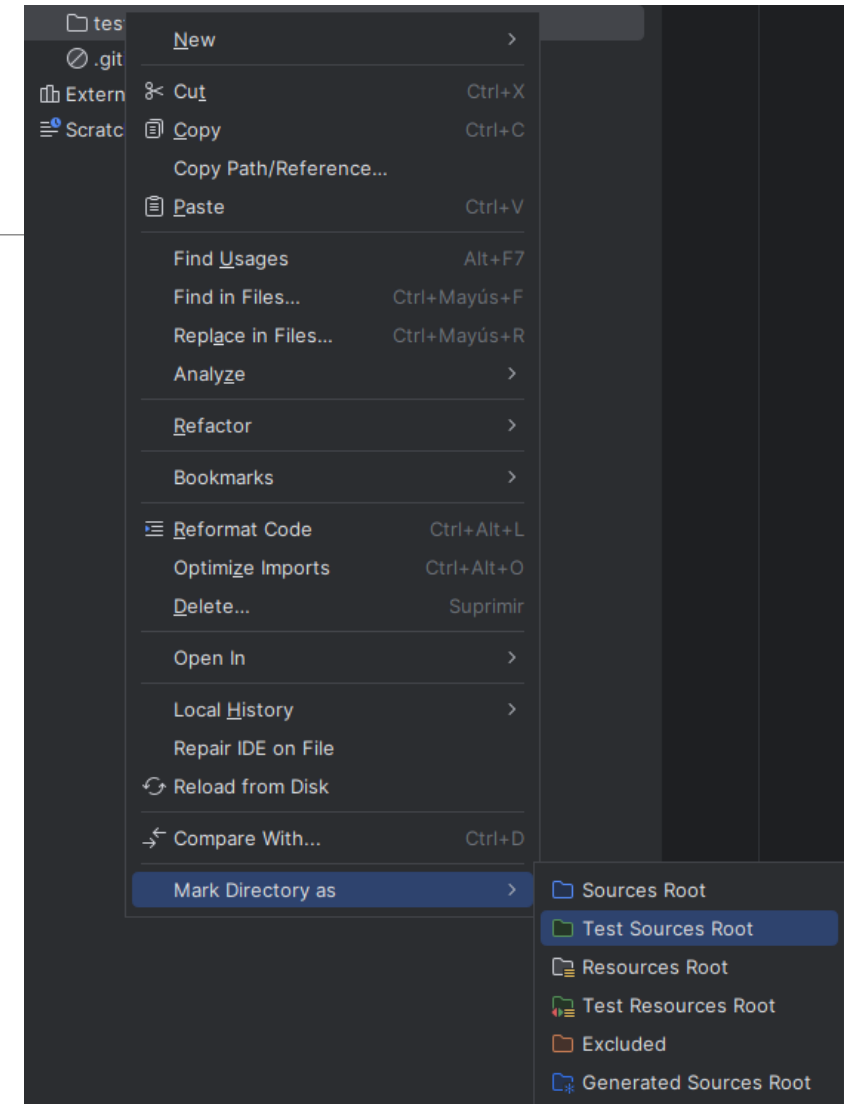


IntelliJ - Carpeta “test”

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

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

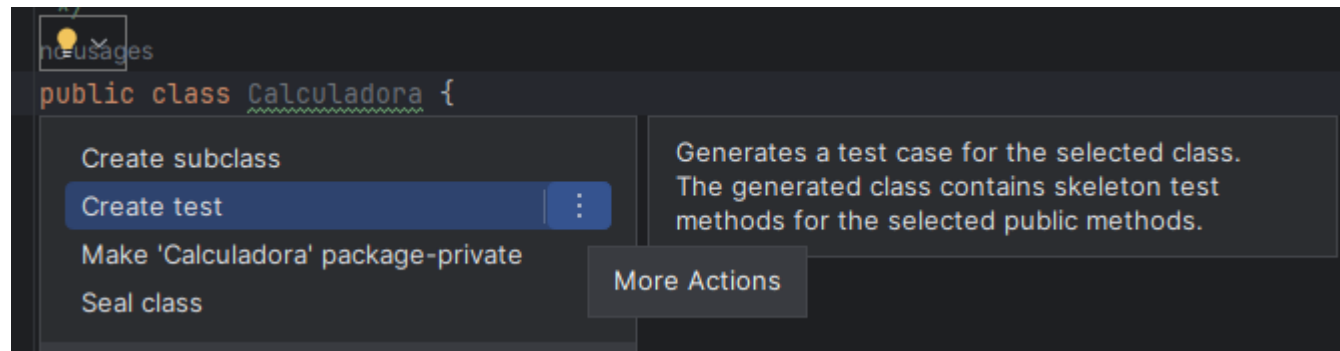
Vemos que cambia su color (a verde)



IntelliJ - Creación de Test

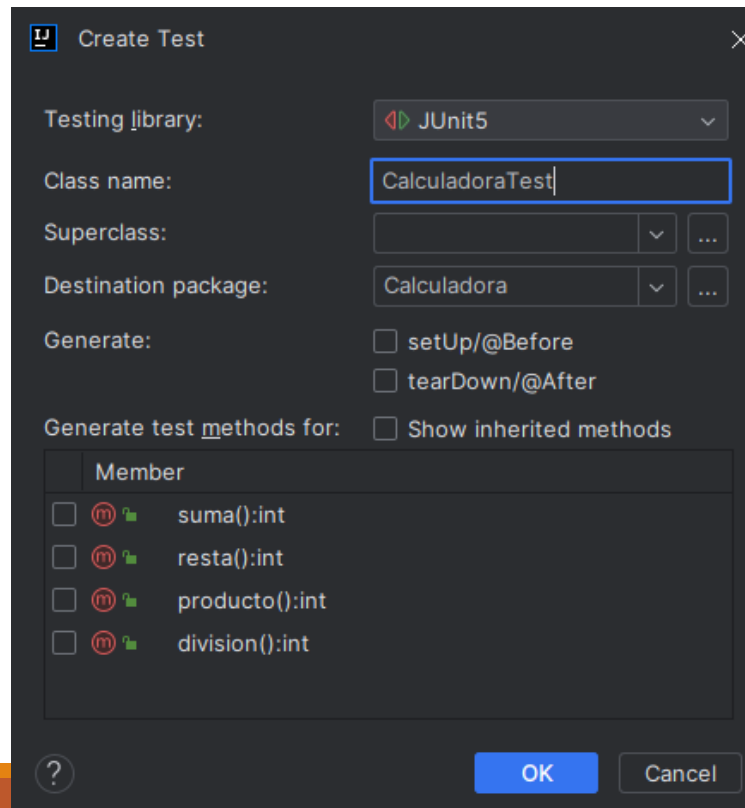
Existen varias formas en IntelliJ

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

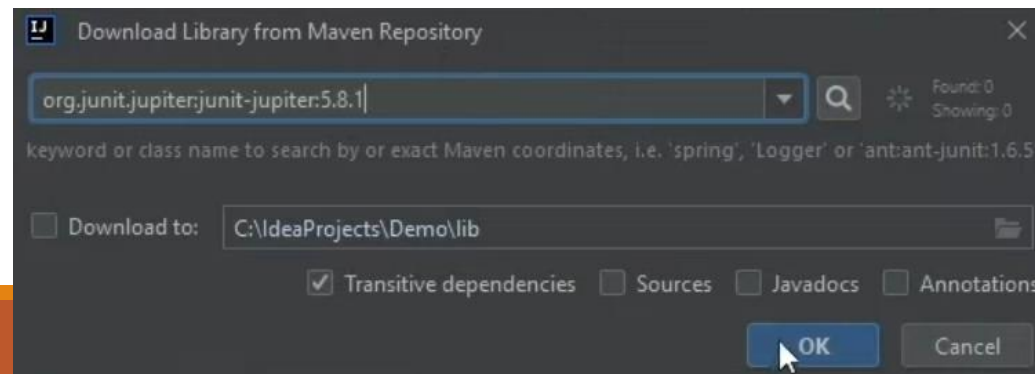
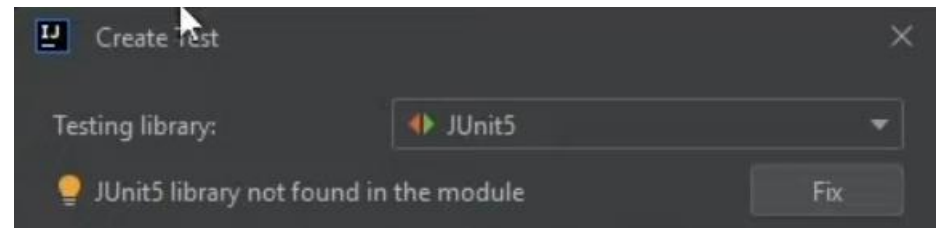
Se nos abrirá la siguiente ventana donde seleccionaremos las opciones que nos interesará para hacer el test:



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 es así y se incluye en el String , al producirse el error se lanzará el mensaje
assertFalse(boolean expresión) assertFalse(String mensaje, boolean expresión)	Comprueba que la expresión se evalúe a false. Si no es así y se incluye en el String , al producirse el error se lanzará el mensaje
assertEquals(valorEsperado, valorReal) assertEquals(String mensaje, valorEsperado, valorReal)	Comprueba que el valorEsperado es igual al valorReal . Si no son iguales y se incluye el String, se lanzará el mensaje. Permite comparar diferentes tipos.
assertNull(Object objeto) assertNull(String mensaje, Object objeto)	Comprueba que el objeto sea null . Si no es null y se incluye el String al 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 incluye el String al 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 sean el mismo objeto. Si no son el mismo y se incluye el String, se lanzará el mensaje
assertNotSame(Object objetoEsperado, Object objetoReal) assertNotSame(String mensaje, Object objetoEsperado, Object objetoReal)	Comprueba que el objetoEsperado y objetoReal no sean el mismo objeto. Si son el mismo y se incluye el String , se lanzará el mensaje
fail() fail(String mensaje)	Hace que la prueba falle. Si se incluye un String , la prueba fallará lanzando el mensaje

Ejemplo: Calculadora

The screenshot displays an IDE with the following components:

- Files Explorer:** Shows the project structure. The **Test Packages** folder is highlighted with a red box, containing a **calculadora** sub-package which includes **CalculadoraTest.java**.
- Navigator:** Shows the members of the **CalculadoraTest** class, including **CalculadoraTest()**, **setUpClass()**, **tearDownClass()**, **testSuma()**, **testResta()**, **testMultiplica()**, and **testDivide()**.
- Source Editor:** Displays the source code of **CalculadoraTest.java**. The code includes imports, package declarations, class declarations, and test methods. The **testSuma()** method is highlighted with a red box.

```
import ...4 lines
12
13 /**...4 lines */
17 public class CalculadoraTest {
18
19     public CalculadoraTest() {
20     }
21
22     @BeforeClass
23     public static void setUpClass() {
24     }
25
26     @AfterClass
27     public static void tearDownClass() {
28     }
29
30     /**
31      * Test of suma method, of class Calculadora.
32
33     @Test
34     public void testSuma() {
35         System.out.println("suma");
36         Calculadora instance = null;
37         int expResult = 0;
38         int result = instance.suma();
39         assertEquals(expResult, result);
40         // TODO review the generated test code and remove
41         fail("The test case is a prototype.");
42     }
43 }
```


Ejemplo: Calculadora

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

testResta - Navigator X

Members

- CalculadoraTest
- CalculadoraTest()
- setUpClass()
- tearDownClass()
- testSuma()
- testResta()
- testMultiplica()
- testDivide()

Test Results X

calculadora.CalculadoraTest X

Tests passed: 25,00 %

1 test passed, 3 tests caused an error. (0,368 s)

- calculadora.CalculadoraTest Failed
- testResta caused an ERROR: java.lang.NullPointerException
- testMultiplica caused an ERROR: java.lang.NullPointerException
- testDivide caused an ERROR: java.lang.NullPointerException

resta
multiplica
divide

Fallo

Ejemplo: Calculadora

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

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

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

```
44 @Test
45 public void testResta() {
46     Calculadora calculo=new Calculadora(20,10);
47     int expectedResult = 10;
48     int result = calculo.resta();
49     assertEquals(expectedResult, result);
50 }
51
52 /**
53  * Test of multiplica method, of class Calculadora.
54  */
55 @Test
56 public void testMultiplica() {
57     Calculadora calculo=new Calculadora(20,10);
58     int expectedResult = 200;
59     int result = calculo.multiplica();
60     assertEquals(expectedResult, result);
61 }
```

Watches Test Results X Output

calculadora.CalculadoraTest X

Tests passed: 100/00 %

All 4 tests passed. (0,117 s)

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()*

calculadora.CalculadoraTest X

Tests passed: 50,00 %

2 tests passed, 1 test failed, 1 test caused an error. (0,106 s)

- calculadora.CalculadoraTest Failed
 - testResta passed (0,003 s)
 - testSuma passed (0,0 s)
 - testMultiplica Failed: Fallo en multiplica expected: <20> but was: <200>
 - Fallo en multiplica expected: <20> but was: <200>
 - junit.framework.AssertionFailedError
 - at calculadora.CalculadoraTest.testMultiplica(CalculadoraTest.java:60)
 - testDivide caused an ERROR: / by zero
 - / by zero
 - java.lang.ArithmeticException
 - at calculadora.Calculadora.divide(Calculadora.java:38)
 - at calculadora.CalculadoraTest.testDivide(CalculadoraTest.java:70)

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;
}
```