


JUnit

Pruebas Unitarias

JUnit



- JUnit es una herramienta para realizar pruebas unitarias automatizadas
- JUnit está integrado en NetBeans y Eclipse



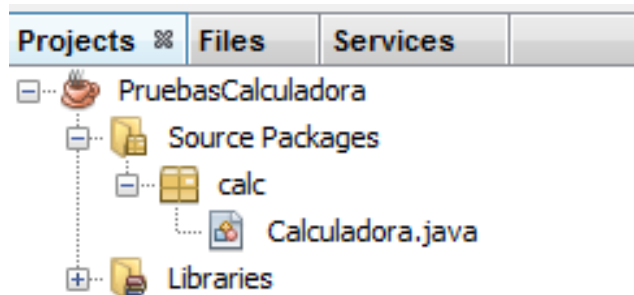
Ejercicio Guiado

JUnit

- Las pruebas unitarias se realizan sobre una Clase (por ejemplo, Calculadora) para probar su comportamiento de forma aislada independientemente del resto de clases de la aplicación
 - ▮ Se creará una Clase de Prueba (por ejemplo, Calculadora**Test**)
 - ▮ Esta clase se va a insertar en un nuevo paquete de nuestro proyecto denominado **Test Packages**

Ejercicio Guiado

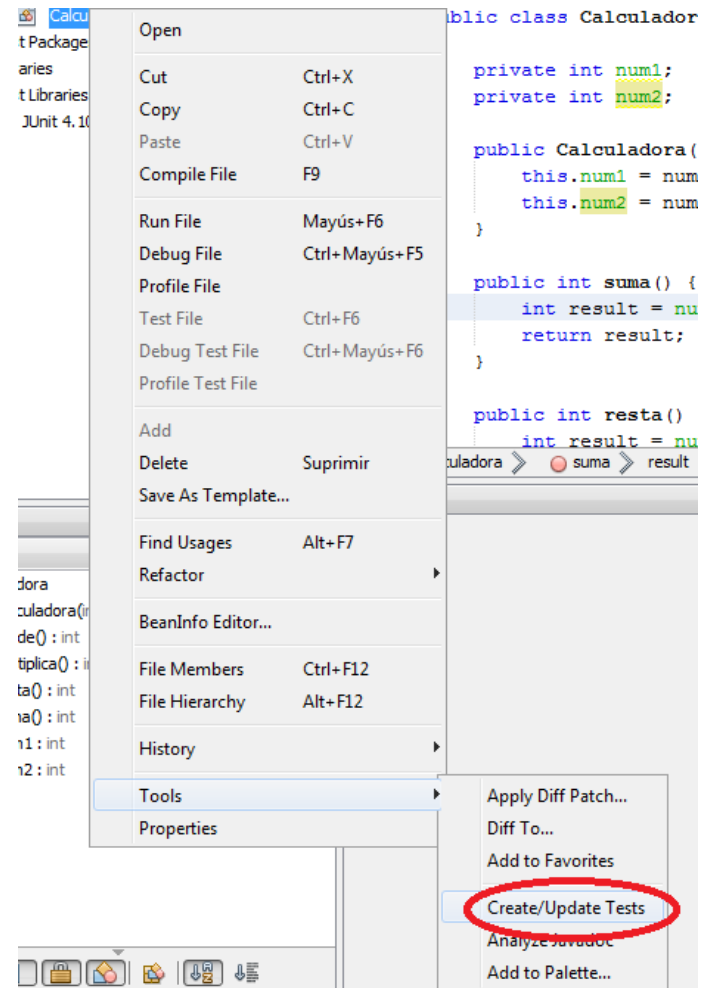
- En NetBeans creamos un nuevo proyecto y creamos la clase Calculadora



```
Calculadora.java
Source History
1 package calc;
2 public class Calculadora {
3     private int num1;
4     private int num2;
5
6     public Calculadora(int num1, int num2) {
7         this.num1 = num1;
8         this.num2 = num2;
9     }
10    public int suma() {
11        int result = num1 + num2;
12        return result;
13    }
14    public int resta() {
15        int result = num1 - num2;
16        return result;
17    }
18    public int multiplica() {
19        int result = num1 * num2;
20        return result;
21    }
22    public int divide() {
23        int result = num1 / num2;
24        return result;
25    }
}
```

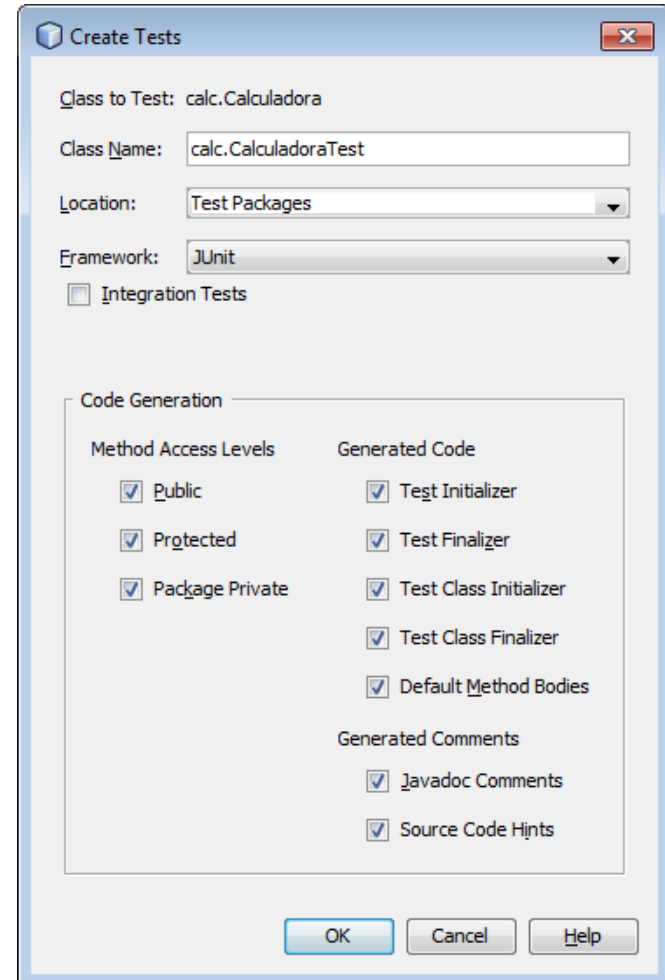
Ejercicio Guiado

- ❑ Creamos la Clase de Tests:
 - ▮ Seleccionamos el clase Calculadora > BotónDerecho > Tools > Create/UpDate Tests
 - ▮ (También desde el menu Tools-> Create/UpDate Tests)



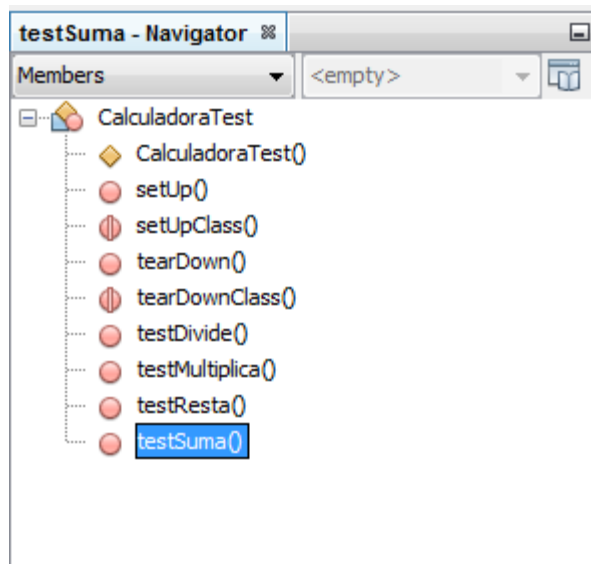
Ejercicio Guiado

- ❑ Puesto que vamos a probar la clase Calculadora, por convenio es recomendable llamar a la clase de prueba CalculadoraTest
 - ▮ Esta clase se va a insertar en un nuevo paquete de nuestro proyecto denominado **Test Packages** (*Paquete de pruebas*)
- ❑ Nos abre la ventana Create Tests
 - ▮ Rellenamos como en la imagen y pulsamos OK



Ejercicio Guiado

- Nos ha generado la clase **CalculadoraTest**:



The screenshot shows the source code of the 'CalculadoraTest.java' file. The code is as follows:

```
18  /*
19  public class CalculadoraTest {
20
21      public CalculadoraTest() {...2 lines }
22
23
24      @BeforeClass
25      public static void setUpClass() {...2 lines }
26
27
28      @AfterClass
29      public static void tearDownClass() {...2 lines }
30
31
32      @Before
33      public void setUp() {...2 lines }
34
35
36      @After
37      public void tearDown() {...2 lines }
38
39      /** Test of suma method, of class Calculadora ...3 lines */
40      @Test
41      public void testSuma() {...9 lines }
42
43
44      /** Test of resta method, of class Calculadora ...3 lines */
45      @Test
46      public void testResta() {...9 lines }
47
48
49      /** Test of multiplica method, of class Calculadora ...3 lines */
50      @Test
51      public void testMultiplica() {...9 lines }
52
53
54      /** Test of divide method, of class Calculadora ...3 lines */
55      @Test
56      public void testDivide() {...9 lines }
57
58  }
```


Ejercicio Guiado

Vemos que:

- Se crea un método de prueba por cada método
 - ▮ Estos métodos son públicos, no devuelven nada y no reciben ningún argumento
 - ▮ El nombre de cada método incluye la palabra test al principio:
 - ▮ testSuma(), testResta(), testMultiplica(), testDivide()
- Sobre cada método aparece la notación @Test que indica al compilador que es un método de prueba
- Cada método tiene una llamada a fail() con mensaje fallo
 - ▮ Este método hace que el test termine con fallo y lanzando el mensaje

Ejercicio Guiado

Vemos que:

- El método `testSuma()` es un método para probar el método `suma` de la clase `Calculadora`

Lo modificamos:

- ▮ Creamos una instancia de la clase `Calculadora` con los valores a operar (20 y 10)
- ▮ Llamamos al método `Suma`
- ▮ Comprobamos los resultados utilizando método `JUnit assertEquals()`

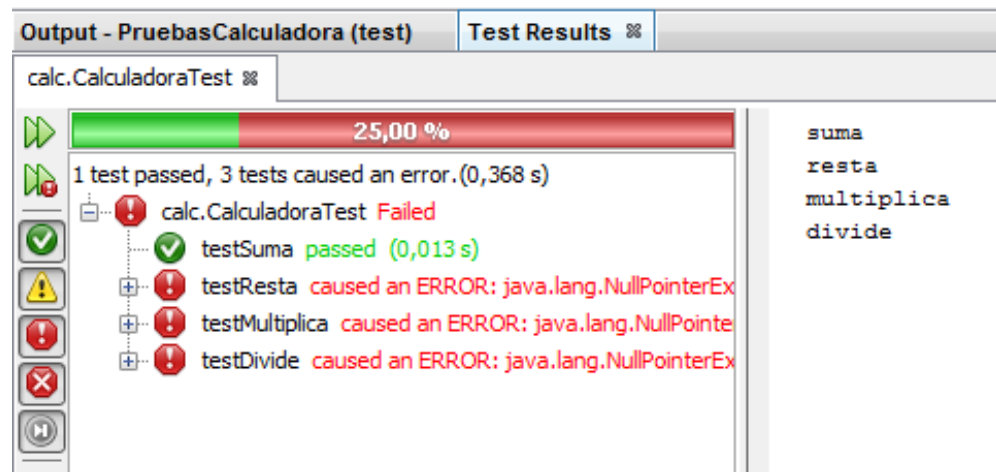
Ejercicio Guiado

□ Modifica testSuma()

```
public void testSuma() {  
    /* System.out.println("suma");  
    Calculadora instance = null;  
    int expectedResult = 0;  
    int result = instance.suma();  
    assertEquals(expectedResult, result);  
    // TODO review the generated test code and remove the default call to fail  
    fail("The test case is a prototype.");  
    */  
  
    System.out.println("suma");  
    Calculadora instance = new Calculadora(20,10);  
    int expectedResult = 30;  
    int result = instance.suma();  
    assertEquals(expectedResult, result);  
    // TODO review the generated test code and remove the default call to fail  
    //fail("The test case is a prototype.");  
  
}
```

Ejercicio Guiado

- Ejecutamos el test:
 - ▮ Seleccionamos la clase Calculadora > BotónDerecho> Test File
- Se nos abre ventana Junit donde aparecen resultados
- Solo se ha realizado satisfactoriamente la prueba con el metodo testSuma()



Métodos JUnit

□ Aserciones

- ▮ Los método assertXXX() se utilizan para hacer las pruebas
- ▮ Estos métodos permiten comprobar si la salida del método que se está probando concuerda con los valores esperados
- ▮ Todos devuelven tipo void

Métodos	Misión
<code>assertNull (Object objeto)</code> <code>assertNull (string mensaje, Object objeto)</code>	Comprueba que el objeto sea Null. Si no es null y se incluye el String al producirse el error devolverá el mensaje.
<code>assertNotNull (Object objeto)</code> <code>assertNotNull (string mensaje, Object objeto)</code>	Comprueba que el objeto no sea Null. Si es null y se incluye el String al producirse el error devolverá el mensaje.

Métodos JUnit

Métodos	Misión
assertTrue (boolean expresión) assertTrue (String mensaje, boolean expresión)	Comprueba que la expresión se evalúe a <u>true</u> . Si no es true y se incluye String, devolverá mensaje de producirse error
assertFalse (boolean expresión) assertFalse (String mensaje, boolean expresión)	Comprueba que la expresión se evalúe a <u>false</u> . Si no es false y se incluye String, devolverá mensaje de producirse error
assertEquals (valorEsperado, valorReal) assertEquals (String mensaje, valorEsperado, valorReal)	Comprueba que el valorEsperado sea igual al valorReal. Si no son iguales y se incluye el String, entonces se lanzará el mensaje. ValorEsperado y valorReal pueden ser de diferentes tipos.
assertEquals (ValorEspDouble, valorRealDobouble, precision)	Comprueba dos valores double o float con la precisión indicada

Métodos JUnit

Métodos	Misión
<code>assertSame (Object objetoEsperado, Object objetoReal)</code> <code>assertSame (String mensaje, Object objetoEsperado, Object objetoReal)</code>	Comprueba que el objetoEsperado y objetoReal sean el mismo objeto. Si no son el mismo y se incluye el String, al producirse error se lanzará el mensaje.
<code>assertNotSame (Object objetoEsperado, Object objetoReal)</code> <code>assertNotSame (String mensaje, Object objetoEsperado, Object objetoReal)</code>	Comprueba que el objetoEsperado y objetoReal no sean el mismo objeto. Si son el mismo y se incluye el String, al producirse error se lanzará el mensaje.
<code>fail()</code> <code>fail (String mensaje)</code>	Hace que la prueba falle. Si se incluye un String la prueba falla lanzando el mensaje.

Más información y opciones:

<https://junit.org/junit4/javadoc/latest/org/junit/Assert.html>

Ejercicio Guiado

- Al lado de cada prueba aparece icono con una marca:

- ▮ Verde indica exitosa
- ▮ Amarillo indica fallo
- ▮ Rojo indica error

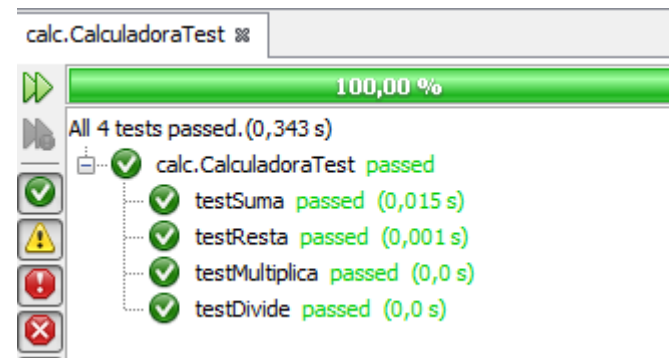


- Ojo no es lo mismo fallo que error!!
 - ▮ fallo es una comprobación que no se cumple
 - ▮ error es una excepción durante la ejecución de código

Ejercicio Guiado

- Implementa los métodos para probar los métodos resta, multiplica y divide de la clase Calculadora

```
public void testResta() {  
    System.out.println("resta");  
    Calculadora instance = new Calculadora(20, 10);  
    int expectedResult = 10;  
    int result = instance.resta();  
    assertEquals(expectedResult, result);  
}  
  
/** Test of multiplica method, of class Calculadora ...3 li  
@Test  
public void testMultiplica() {  
    System.out.println("multiplica");  
    Calculadora instance = new Calculadora(20, 10);  
    int expectedResult = 200;  
    int result = instance.multiplica();  
    assertEquals(expectedResult, result);  
}  
  
/** Test of divide method, of class Calculadora ...3 lines  
@Test  
public void testDivide() {  
    System.out.println("divide");  
    Calculadora instance = new Calculadora(20, 10);  
    int expectedResult = 2;  
    int result = instance.divide();  
    assertEquals(expectedResult, result);  
}
```



Diferencia fallo y error

- Provocamos fallo en el método multiplica()
 - ▮ hacemos que el valor esperado no coincida con el resultado
 - ▮ incluimos String con mensaje para si se produce fallo muestre el mensaje

```
/**
 * Test of multiplica method, of class Calculadora.
 */
@Test
public void testMultiplica() {
    System.out.println("multiplica");
    Calculadora instance = new Calculadora(320, 10);
    int expResult = 200;
    int result = instance.multiplica();
    assertEquals("Fallo en el Producto", expResult, result);
}
```

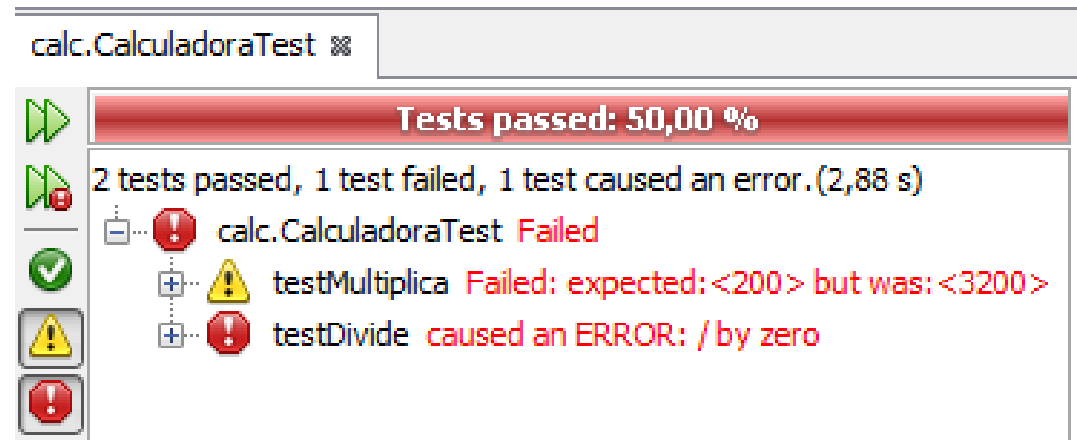
Diferencia fallo y error

- Provocamos error en el método divide()
 - ▮ al crear objeto calculadora asignamos 0 al segundo parámetro
 - ▮ Al dividir por 0 producirá excepción

```
/**
 * Versión 1. Test of divide method, of class Calculadora.
 */
@Test
public void testDivide() {
    System.out.println("divide");
    Calculadora instance = new Calculadora(20, 0);
    int expResult = 2;
    int result = instance.divide();
    assertEquals(expResult, result);
}
```

Diferencia fallo y error

- ❑ Ojo no es lo mismo fallo que error!!
 - ▮ fallo es una comprobación que no se cumple
 - ▮ error es una excepción durante la ejecución de código



Capturando Excepción

- Nos interesa capturar la excepción siempre que se produzca una división entre cero
 - ▮ Vemos la implementación:

```
@Test
public void testDivide() {
    System.out.println("divide");
    try {
        Calculadora instance = new Calculadora(20, 0);
        int resultado = instance.divide();
        //assertEquals(expResult, result);
        fail("Fallo, Deberia lanzar la excepcion");
    } catch (ArithmeticException e) {
        //prueba satisfactoria
    }
}
```