



Pruebas Parametrizadas

Pruebas Parametrizadas

- Supongamos queremos ejecutar una prueba varias veces con distintos valores de entrada
 - ▣ Por ejemplo: vamos a probar el método *divide()* con *diferentes valores*
- JUnit nos permite generar parámetros para lanzar varias veces una prueba con dichos parámetros
- Para conseguir esto seguimos dos pasos:

Pruebas Parametrizadas

□ Paso 1:

- ▣ Añadimos la etiqueta `@RunWith(Parameterized.class)` a la clase `test`
 - Requerirá nuevos import
- ▣ Con esto indicamos a la clase que va a ser usada para realizar una batería de pruebas
- ▣ En esta clase se debe declarar
 - un atributo por cada uno de los parámetros de la prueba
 - y un constructor con tantos argumentos como parámetros en cada prueba

□ Ejemplo:

- ▣ Para probar el método `divide` (o cualquier otro) definiremos 3 parámetros, dos de ellos para los números con los que se realiza la operación y el tercero para recoger el resultado

Pruebas Parametrizadas

- Creamos una nueva clase Calculadora4Test

```
import org.junit.Test;
import static org.junit.Assert.*;
import org.junit.runner.RunWith;
- import org.junit.runners.Parameterized;
|
@RunWith(Parameterized.class)
public class Calculadora4Test {

    private int num1;
    private int num2;
    private int resul;

    public Calculadora4Test(int num1, int num2, int resul) {
        this.num1 = num1;
        this.num2 = num2;
        this.resul = resul;
    }
- }
```

Pruebas Parametrizadas

- Paso 2:
 - ▣ Definimos un método anotado con la etiqueta `@Parameters` que será el encargado de devolver la lista de valores a probar
 - ▣ En este método se definirán filas de valores para `num1`, `num2` y `resul` (en el mismo orden que en el constructor)

Pruebas Parametrizadas

□ Ejemplo:

- Un grupo de valores de prueba seria {20, 10, 2}
 - Para la división equivale a $\text{num1} / \text{num2} = \text{resul}$, esto es, $20 / 10 = 2$ (sería un caso de prueba correcto)
- Otros grupos de valores de prueba seria {30, -2, -15} (correcto) y {5, 2, 3} (incorrecto)

```
@Parameters  
public static Collection<Object[]> numeros() {  
    return Arrays.asList(new Object[][]{{20, 10, 2}, {30, -2, -15}, {5, 2, 3}});  
}
```

Atención! Quédate con la idea aunque no entiendas todo el código Java utilizado

Pruebas Parametrizadas

- El método `testDivide()` de la clase `Calculadora4Test` podría ser:

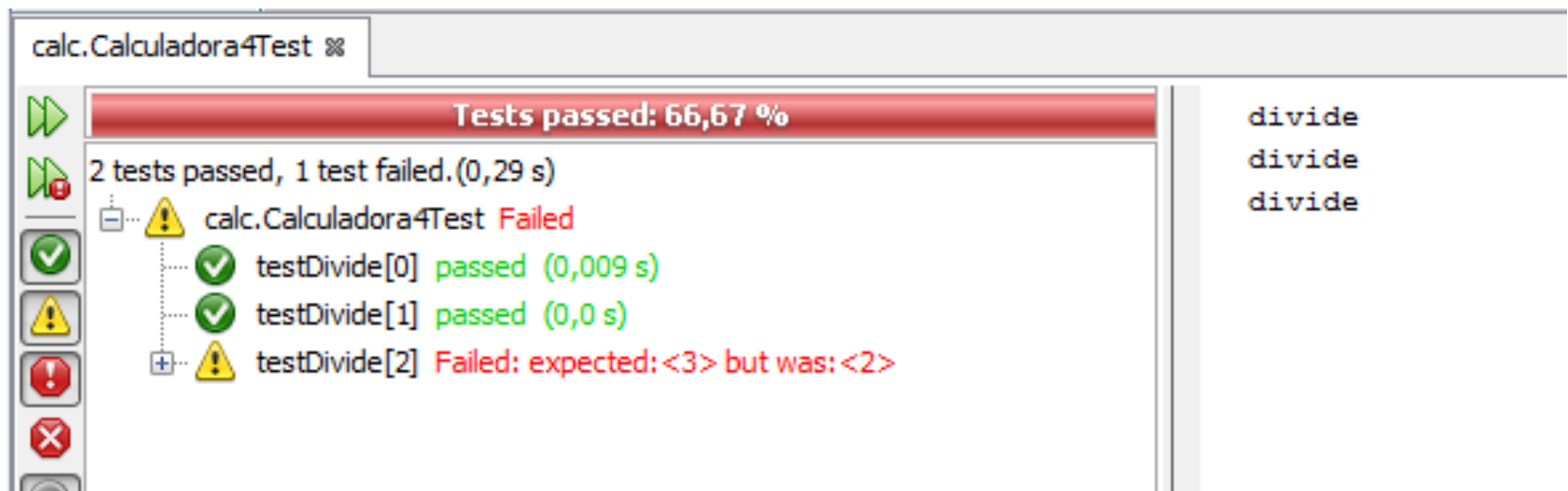
```
@Test
public void testDivide() {

    System.out.println("divide");
    Calculadora instance = new Calculadora(this.num1, this.num2);
    int division = instance.divide();
    assertEquals( this.resul, division);

}
```

Pruebas Parametrizadas

- La ejecución produce la salida siguiente:



- Al lado del método se muestra entre corchetes la prueba que se trata



Suite de Pruebas

Suite de Pruebas

- JUnit nos proporciona el mecanismo llamado Test Suites que agrupa varias Clases de Prueba para que se ejecuten una tras otra
- Vamos a ver el procedimiento de creación de la suite de pruebas con un Ejercicio Guiado:
 1. Creamos distintas Clases de Prueba
 2. Creamos el Test Suite

Ejercicio Guiado

- Creamos pruebas parametrizadas para los métodos suma(), resta() y multiplica()

- El nombre para las clases de prueba es:

CalculadoraSumaTest

CalculadoraRestaTest

CalculadoraMultiplicaTest

- Atención! Debes crear las otras clases siguiendo el ejemplo

```
@RunWith(Parameterized.class)
public class CalculadoraSumaTest {

    private int num1;
    private int num2;
    private int resul;

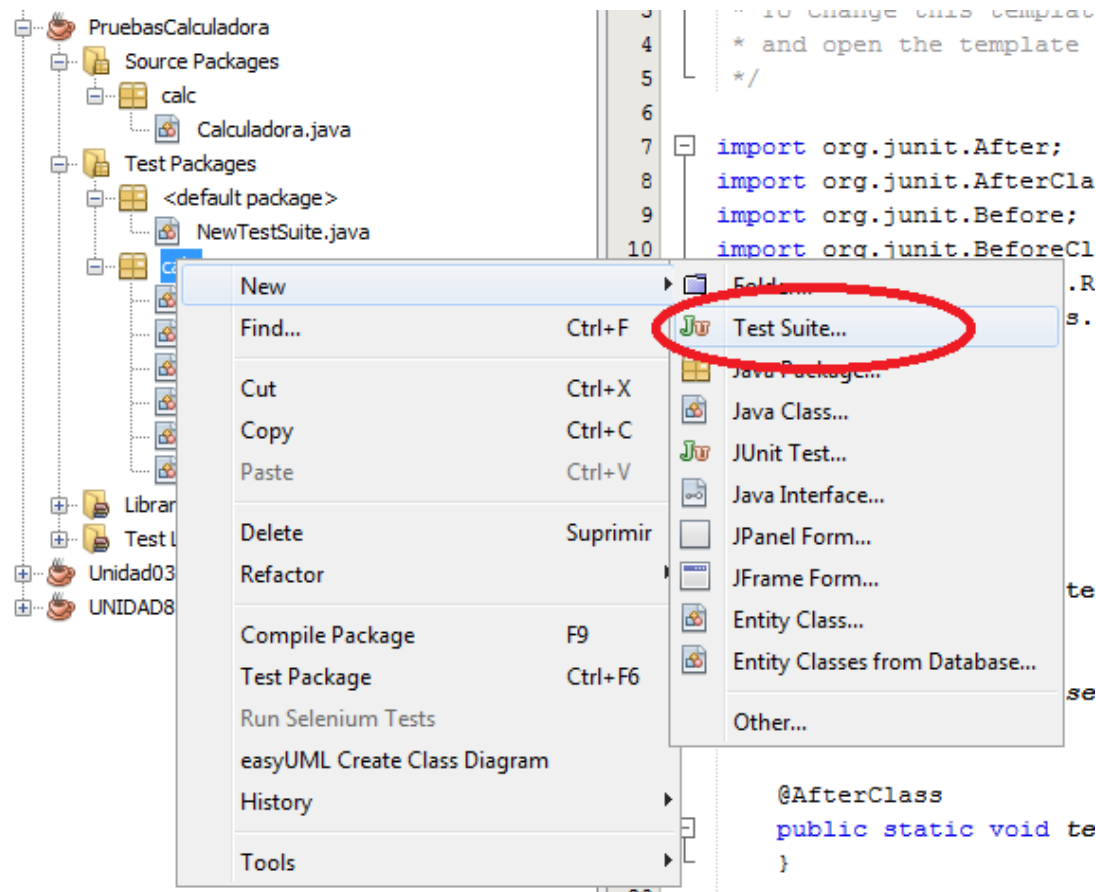
    public CalculadoraSumaTest(int num1, int num2, int resul) {
        this.num1 = num1;
        this.num2 = num2;
        this.resul = resul;
    }

    @Parameters
    public static Collection<Object[]> numeros() {
        return Arrays.asList(new Object[][]{
            {20, 10, 30}, {30, -2, 28}, {5, 2, 7}
        });
    }

    /** Test of suma method, of class Calculadora ...3 lines */
    @Test
    public void testSuma() {
        System.out.println("Suma");
        Calculadora instance = new Calculadora(this.num1, this.num2);
        int calculo = instance.suma();
        assertEquals(this.resul, calculo);
    }
}
```

Ejercicio Guiado

□ Creamos la Suite de Pruebas (Test Suite)



Ejercicio Guiado

- Damos nombre a la clase: PruebaTestSuite

New Test Suite

Steps

1. Choose File Type
2. **Name and Location**

Name and Location


Class Name:

Project:

Location:

Package:

Created File:

The created test suite will comprise tests for all classes in the selected package. 

Generated Code

- ☒ Test Initializer
- ☒ Test Finalizer
- ☒ Test Class Initializer
- ☒ Test Class Finalizer

Generated Comments

- ☒ Source Code Hints

< Back Next > **Finish** Cancel Help

Ejercicio Guiado

- El asistente genera una SuiteTest con todas las Clases de Prueba del Paquete
- Podemos revisar las que nos interesan:

```
@RunWith(Suite.class)
```

@RunWith(Suite.class)

Indica a Junit que es una suite de pruebas

```
@Suite.SuiteClasses({calc.CalculadoraSumaTest.class,  
    calc.CalculadoraMultiTest.class, calc.CalculadoraRestaTest.class})
```

@Suite.SuiteClasses()

Indica las clases que forman parte del conjunto de pruebas y que son las que se van a ejecutar

Ejercicio Guiado

- Vemos que dentro de la clase no se genera ninguna línea de código
- Únicamente resta ejecutar y se ejecutarán las clases una detrás de otra

