CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

# UD 4

## DISEÑO Y REALIZACIÓN DE PRUEBAS

**ENTORNOS DE DESARROLLO 20/21**
**CFGS DAW**

### PARTE 3 DE 3 - PRÁCTICA JUNIT EN NETBEANS

Autora: Cristina Álvarez Villanueva
Revisado por: Sergio Badal
sergio.badal@ceedcv.es
Fecha: 25/11/20
Licencia Creative Commons
versión 1.0

**Reconocimiento – NoComercial – CompartirIgual (by-nc-sa)**: No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

## 1. JUSTIFICACIÓN DE LA PRÁCTICA

En la práctica anterior vimos **JUnit con Eclipse. Aquí** se muestra su funcionamiento **con NetBeans** (es casi idéntico) con otro ejemplo distinto, de manera que sirva de repaso.

Usaremos **APACHE NETBEANS 12**. Si tienes una versión inferior a la 8, puede que algún comando esté cambiado de sitio. Busca en Google la solución o consulta en los foros y te ayudaremos.

## 2. CREACIÓN DE UNA CLASE DE PRUEBA

Desmarcar la opción de crear la clase Main.

Crea un proyecto nuevo en NetBeans por ejemplo (**CalculadoraNB**). Vamos a hacer un par de clases, una clase **Suma**.java y otra clase **Resta**.java. Ambas tienen dos métodos, la primera **getSuma**() e **incrementa**(), la segunda **getDiferencia**() y **decrementa**().

Los códigos son los siguientes:

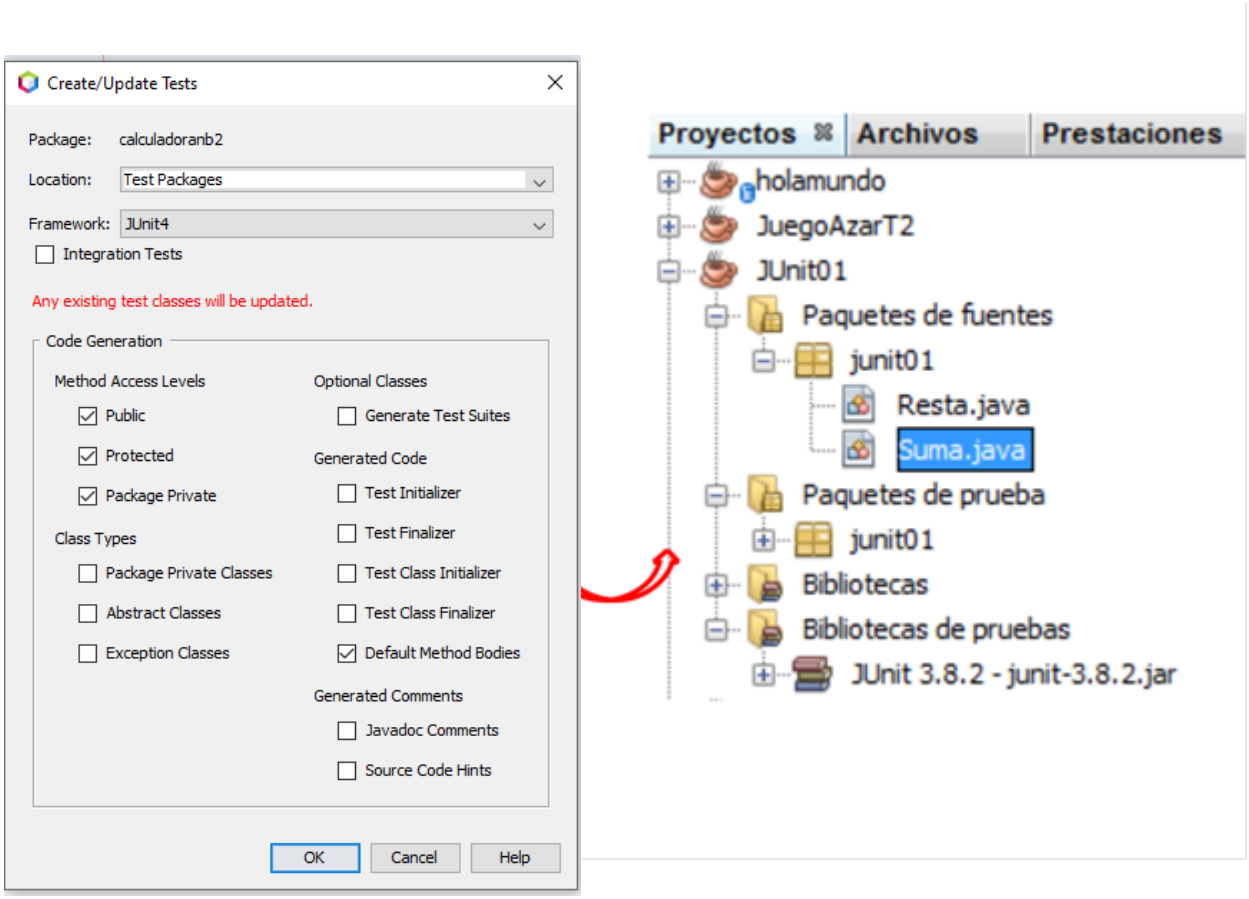
```
package calculadoranb;

public class Suma {

    public double getSuma(double a, double b)
    {
        return a + b;
    }

    public double incrementa(double a)
    {
        return a + 1;
    }

}
```

```
package calculadoranb;

public class Resta {

    public double getResta(double a, double b)
    {
        return a - b;
    }

    public double decrementa(double a)
    {
        return a - 1;
    }

}
```

Vamos a usar JUnit para hacer los test de estas clases y de sus métodos. Para nuestros programas de prueba podemos hacer una o más clases de prueba, pero lo normal es hacer una clase de prueba por cada clase a probar o bien, una clase de prueba por cada conjunto de pruebas que esté relacionado de alguna manera.

Nosotros tenemos dos clases; **Suma** y **Resta**, así que hacer dos clases de prueba **SumaTest** y **RestaTest**.

Comenzamos por el método Suma(). Hay varias maneras de abrir JUnit en Netbeans: con el ***menú contextual > Tools > Create/Update Tests*** sobre la clase a probar o ***Menú Herramientas > Crear Actualizar Test***. Aparece un menú contextual, donde nos da a elegir el tipo de pruebas a hacer (lo dejamos de momento por defecto y le damos a Aceptar).




Nos crea automáticamente **SumaTest.java.** En el Tab Proyectos debajo del nombre **CalculadoraNB**, nos aparecerán dos cosas nuevas: en ***Paquetes de pruebas*** nos saldrá la clase creada ***SumaTest.java*** que acabamos de crear, y en la librería *de pruebas* nos saldrá la **librería de Junit** y su versión **4.** Ten en cuenta que **Apache NetBeans 12 no es compatible con JUnit 5.**

## 3. PERSONALIZAR CLASES TEST

Realizar lo mismo con **Resta**, ahora tenemos **dos clases; *Suma* y *Resta*, y dos de prueba *SumaTest* y *RestaTest*.**

Ahora tenemos que hacer los métodos de test. Cada método probará alguna cosa de la clase. En el caso de ***SumaTest***, vamos a hacer dos métodos de test, de forma que cada uno pruebe uno de los métodos de la clase *Suma, que son* ***getSuma e Incrementa***.

Por ejemplo, para probar el método ***getSuma()*** de la clase *Suma*, vamos a hacer un método de prueba ***testGetSuma().*** Este método solo tiene que instanciar la clase *Suma*, llamar al método *getSuma()* con algunos parámetros y comprobar que el resultado es el esperado.

Modificaremos los métodos autogenerados por **NetBeans**:

```
@Test
public void testGetSuma() {
    System.out.println(" getSuma ");
    double a = 1.0;
    double b = 1.0;
    Suma instance = new Suma();
    double expResult = 2.0;
    double result = instance.getSuma(a, b);
    assertEquals(expResult, result, 0);
}

@Test
public void testIncrementa() {
    System.out.println("incrementa");
    double a = 1.0;
    Suma instance = new Suma();
    double expResult = 2.0;
    double result = instance.incrementa(a);
    assertEquals(expResult, result, 0);
}
```

```
@Test
public void testGetResta() {
    System.out.println(" getResta ");
    double a = 3.0;
    double b = 2.0;
    Resta instance = new Resta();
    double expResult = 1.0;
    double result = instance.getResta(a, b);
    assertEquals(expResult, result, 0);
}

@Test
public void testDecrementa() {
    System.out.println("decrementa");
    double a = 3.0;
    Resta instance = new Resta();
    double expResult = 2.0;
    double result = instance.decrementa(a);
    assertEquals(expResult, result, 0);
}
```

Vamos a sumar 1 más 1, que ya sabemos que devuelve 2. ¿Cómo comprobamos ahora que el resultado es el esperado?.

Uno de los métodos más usados es **assertEquals(expResult, result, sensibilidad)** , que en general admite dos parámetros: el primero es el valor esperado y el segundo el valor que hemos obtenido. **El tercer parámetro es la sensibilidad de la comparación y solo te lo pedirá si estás usando Apache NetBeans (versión > 8). Dejalo siempre a cero.**

El método **testGetSuma** está indicando que tome como parámetros a y b los valores 1.0 y 1.0 el método **getSuma**, y debe dar como resultado 2.0.

El método **assertEquals(expResult, result, sensibilidad)** indica el valor esperado que es 2.0 y el valor devuelto por la función **getSuma** que deberá calcularse.

## 4. TIPOS DE ASSERT

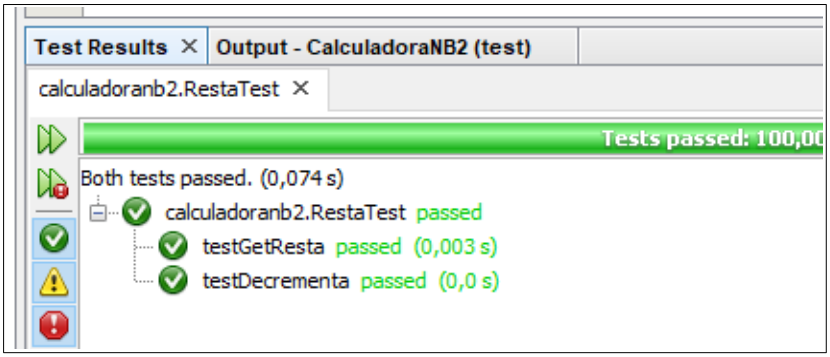
Existen varios tipos de assert (afirmaciones), se resumen en:

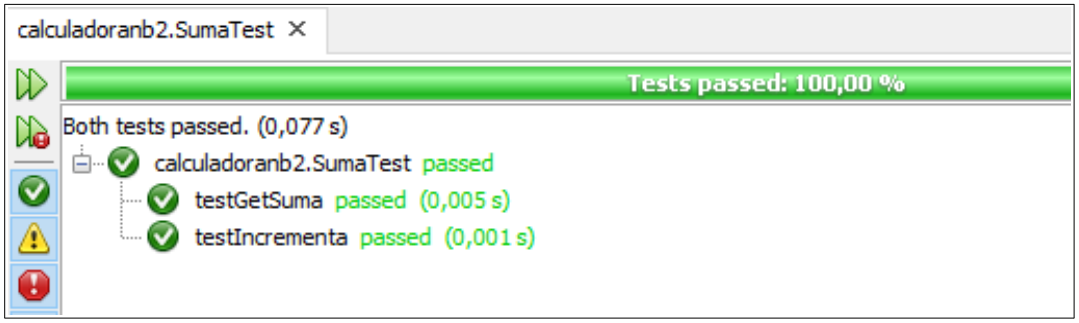
| | |
|---|---|
| static void assertEquals(int expected, int actual); | Afirma que son iguales dos enteros. |
| static void assertEquals(String message , int expected, int actual); | Afirma que son iguales dos enteros con mensaje si no falla. |
| static void assertEquals(java.lang.String message, double expected, double actual, double delta) | Afirma si sin iguales dos doubles. Siendo:<br>*expected* es el primer tipo Double que se va a comparar. Es el tipo Double que la prueba unitaria espera.<br>*actual* es el segundo tipo Double que se va a comparar. Es el tipo Double producido por la prueba unitaria.<br>*delta* es la precisión necesaria. Se producirá un error en la aserción sólo si expected es diferente de actual en más de delta. |
| static void assertNotNull( Objetct object) | Afirma que un un objeto no es null |
| static void faul ( String message) | Falla un test con un mesage dado. |

Hay más tipos que se pueden consultar en el API:
https://junit.org/junit5/docs/5.3.1/api/org/junit/platform/runner/JUnitPlatform.html

## 5. EJECUCIÓN DE LAS PRUEBAS

Si ejecutamos cada una de las clases de prueba obtendremos los resultados similares a los que vimos con Eclipse:




## 6. BIBLIOGRAFÍA Y ENLACES

Álvarez, C. (2014): “Entornos de desarrollo”, CEEDCV