

1. Justificación de la práctica
2. Creación de una clase de prueba
3. Personalizar SumaTest.java
4. Tipos de assert
5. SumaTest.java
6. RestaTest.java
7. Bibliografía y enlaces
8. Ejercicios



Temporalización:
27 oct – 17 nov

Autora: Cristina Álvarez Villanueva

Revisado por:
Fco. Javier Valero Garzón
M.^a Carmen Safont

GUARDAR LOS RESULTADOS DE LA PRÁCTICA EN UNA CARPETA

1. JUSTIFICACIÓN DE LA PRÁCTICA

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

2. CREACIÓN DE UNA CLASE DE PRUEBA

Desmarcar la opción de crear la clase Main.



Crea un proyecto nuevo en NetBeans por ejemplo (**JUnit01**). 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: **Ojo** con el return de Suma!!

```
public class Suma {  
  
    public double getSuma(double a, double b) {  
        // Se multiplica en vez de sumar a posta , para que los test fallen  
        return a * b;  
    }  
  
    public double incrementa(double a) {  
        return a + 1;  
    }  
}
```

[Suma.java]

```

public class Resta {

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

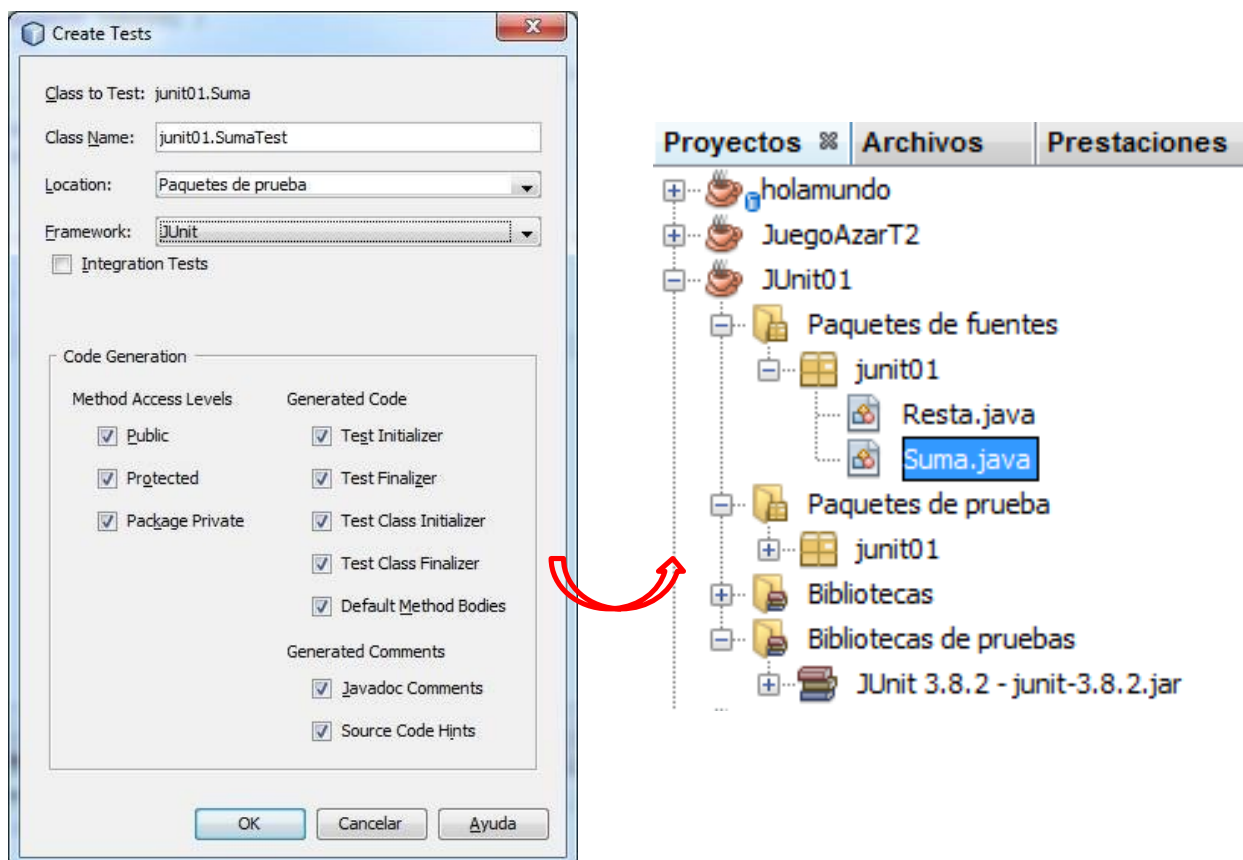
    public double decremента(double a) {
        return a - 1;
    }
}

```

[Resta.java]

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***. No es necesario llamar a estas clases con *****Test**, pero sí es recomendable para identificar rápidamente las clases que son de test automático.

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 Junit01, 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 **5.3.1**.

3. PERSONALIZAR SUMATEST.JAVA

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 el método **testGetSuma** generado por **NetBeans**:

```
public class SumaTest {  
    ...  
    public void testGetSuma () {  
        System . out . println ( " getSuma " );  
        double a = 1.0;  
        double b = 1.0;  
        Suma instance = new Suma ();  
        double expectedResult = 2.0;  
        double result = instance . getSuma ( a, b );  
        assertEquals ( expectedResult , result );  
    }  
    ...  
}
```

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()**, que en general admite dos parámetros: el **primero** es el valor **esperado** y el **segundo** el valor que hemos **obtenido**. Eliminar el tercer parámetro del Netbeans.

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)** indica el valor esperado que es 2.0 y el valor devuelto por la función *getSuma* que deberá calcularse. En nuestro caso **debe dar error**, ya que así lo definimos en la clase *Suma.java*

Modificaremos el método **testIncrementa** de este modo:

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

assertEquals(expResult, result); devolverá cierto si *incrementa(1)* devuelve 2. En nuestro caso el **Test será correcto**.

4. TIPOS DE ASSERT

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

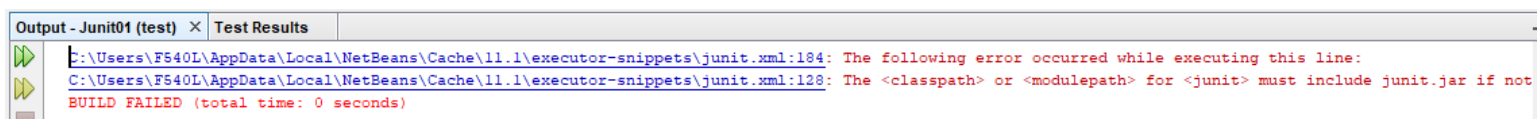
<code>static void assertEquals(int expected, int actual);</code>	Afirma que son iguales dos enteros.
<code>static void assertEquals(String message , int expected, int actual);</code>	Afirma que son iguales dos enteros con mensaje si no falla.
<code>static void assertEquals(java.lang.String message, double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

Vamos a pasarle el test a la clase **SumaTest**. Para ello, encima de la clase botón derecho y Ejecutar Archivo. Os saldrá el siguiente **error**:



```
Output - JUnit01 (test) × Test Results
E:\Users\F540L\AppData\Local\NetBeans\Cache\11.1\executor-snippets\junit.xml:184: The following error occurred while executing this line:
C:\Users\F540L\AppData\Local\NetBeans\Cache\11.1\executor-snippets\junit.xml:128: The <classpath> or <modulepath> for <junit> must include junit.jar if not
BUILD FAILED (total time: 0 seconds)
```

Nos encontramos con un problema grave de NetBeans 11.1

La versión de Junit 5.3.1 que viene por defecto, no funciona bien y por eso nos sale el mensaje de error.

Tras mucho buscar encontré este foro:

<https://translate.google.com/translate?hl=es&sl=en&u=https://stackoverflow.com/questions/54161715/netbeans-10-junit-jar-not-found&prev=search>

que básicamente dice que **volvamos a una versión anterior la junit 4.12** y que arreglemos una serie de cosas en el código para poder usar pruebas.

Vamos hacerlo.

1) en las librerías de test, haga clic derecho e **importe las Junit 4.12.**

2) **eliminar la versión Junit 5.3.1**

3) **Eliminar todos los import** de la parte superior del fichero **SumaTest.java** y añadir los de Junit4 que son:

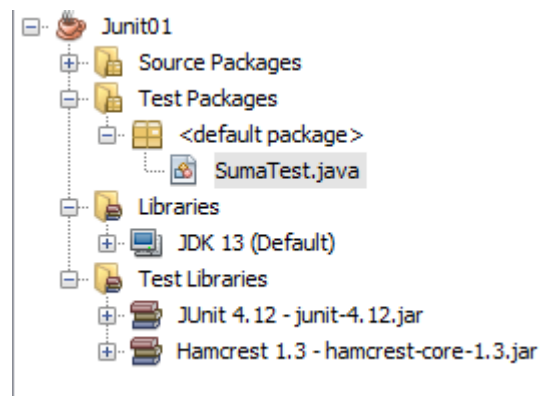
```
import static junit.framework.Assert.assertEquals;
import org.junit.Test;
```

4) **Borrar** las etiquetas **beforeAll** y otras **etiquetas** de prueba. Solo dejar el **@Test**. **También** eliminar las dos líneas de la etiqueta **fail("The test case is a prototype.");**

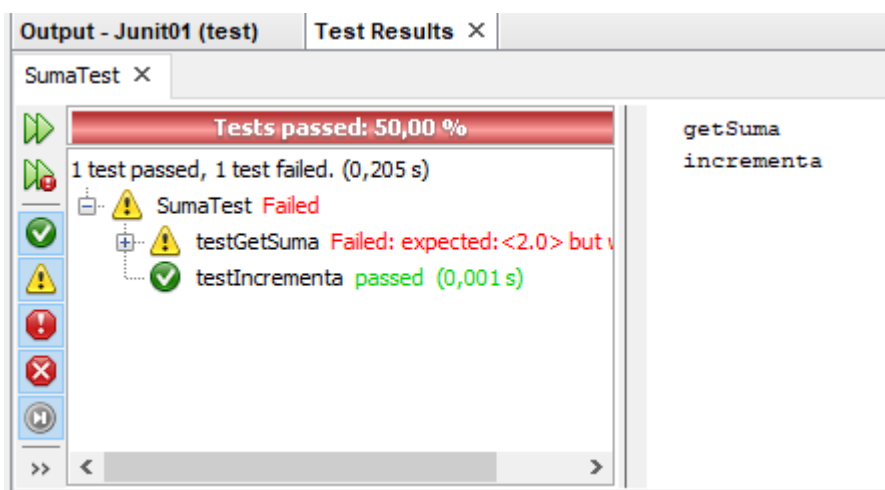
5) **guardar todo, cerrar NetBeans** y volver a abrir el proyecto.

Se quejará de que las bibliotecas **Hamcrest** para junit4 no se importan. Permitir que NetBeans resuelva el problema.

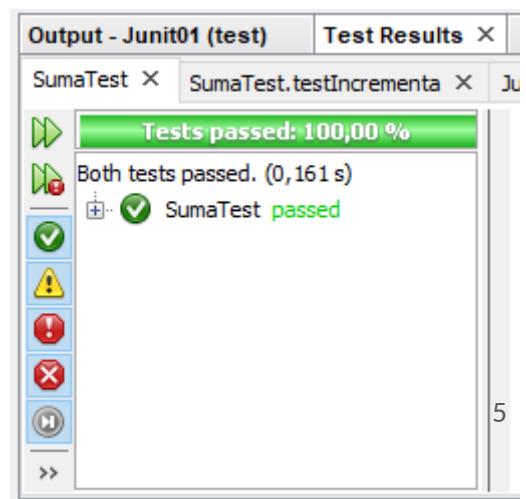
La apariencia final es:



Ahora por fin. Vamos a pasarle el test a la clase **SumaTest**. Para ello, encima de la clase botón derecho y Ejecutar Archivo.



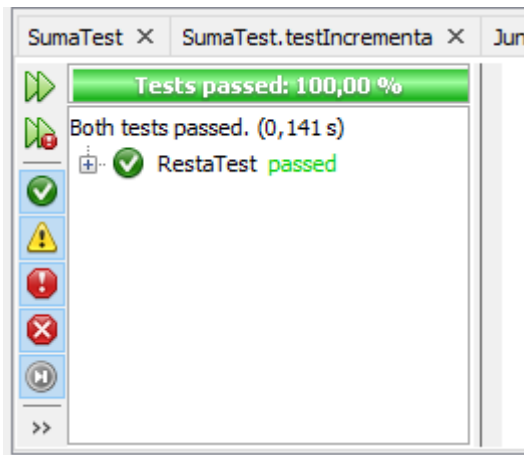
Como es lógico **testGetSuma falla** ya que así lo programamos. Arreglarlo para que no de fallo modificando Suma.java



6. RESTATEST.JAVA

Repetir los conceptos anteriores creando el test de la clase *Resta.java* que se llamará **RestaTest.java**

Cuando lo tengáis pasar el test a la clase **RestaTest** con ejecutar archivo:



7. EJERCICIOS

Realiza el ejercicio y guardate capturas de pantalla de los pasos por si os lo pedimos.

Ejercicio 1:

Utilizando el proyecto Junit1, añadir a una nueva clase llamada *Multiplica*.

Crear los test para el método *int multiplicados(int a, int b)*, para el siguientes caso:

- 4*2

8. BIBLIOGRAFÍA Y ENLACES

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