



PRUEBAS UNITARIAS

JUnit



Introducción:

En programación, una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código. Por ejemplo en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase.

La idea es escribir casos de prueba para cada función no trivial o método en el módulo, de forma que cada caso sea independiente del resto.

En el momento de dar por finalizado un módulo, se comprueba el funcionamiento del mismo.

Pruebas unitarias:

- Pruebas de **caja negra**: comprobación de la respuesta de un módulo a partir de los datos de entrada (comprobación con valores válidos y no válidos)
- Pruebas de **caja blanca**: prueba de cada uno de los posibles “caminos” internos del módulo (condiciones, bucles, ...). Comprobación mediante DEBUG

Pruebas Unitarias: JUnit

JUnit es un conjunto de clases que permite realizar la ejecución pruebas unitarias sobre código Java.

CARACTERÍSTICAS:

- Herramienta para la realización de pruebas unitarias automatizadas.
- Pruebas de una clase para comprobar su comportamiento de modo independiente al resto de la aplicación.
- Eclipse trae integrado esta funcionalidad.



Pruebas Unitarias: JUnit, Métodos

Método setUp: Asignamos valores iniciales a variables antes de la ejecución de cada test. Si solo queremos que se inicializan al principio una vez, el método se debe llamar “setUpBeforeClass”

Método tearDown: Es llamado después de cada test y puede servir para liberar recursos o similar. Igual que antes, si queremos que solo se llame al final de la ejecución de todos los test, se debe llamar “tearDownAfterClass”

Métodos Test: Contienen las pruebas concretas que vamos a realizar.

Pruebas Unitarias: JUnit, Funciones de Aceptación

Funciones de Aceptación

assertArrayEquals(): Recibe como parámetro 2 arrays y comprueba si son iguales. Devuelve `assertionError` si no se produce el resultado esperado.

assertEquals(): Realiza la comprobación entre 2 valores de tipo numérico. Devuelve `assertionError` si no se produce el resultado esperado.

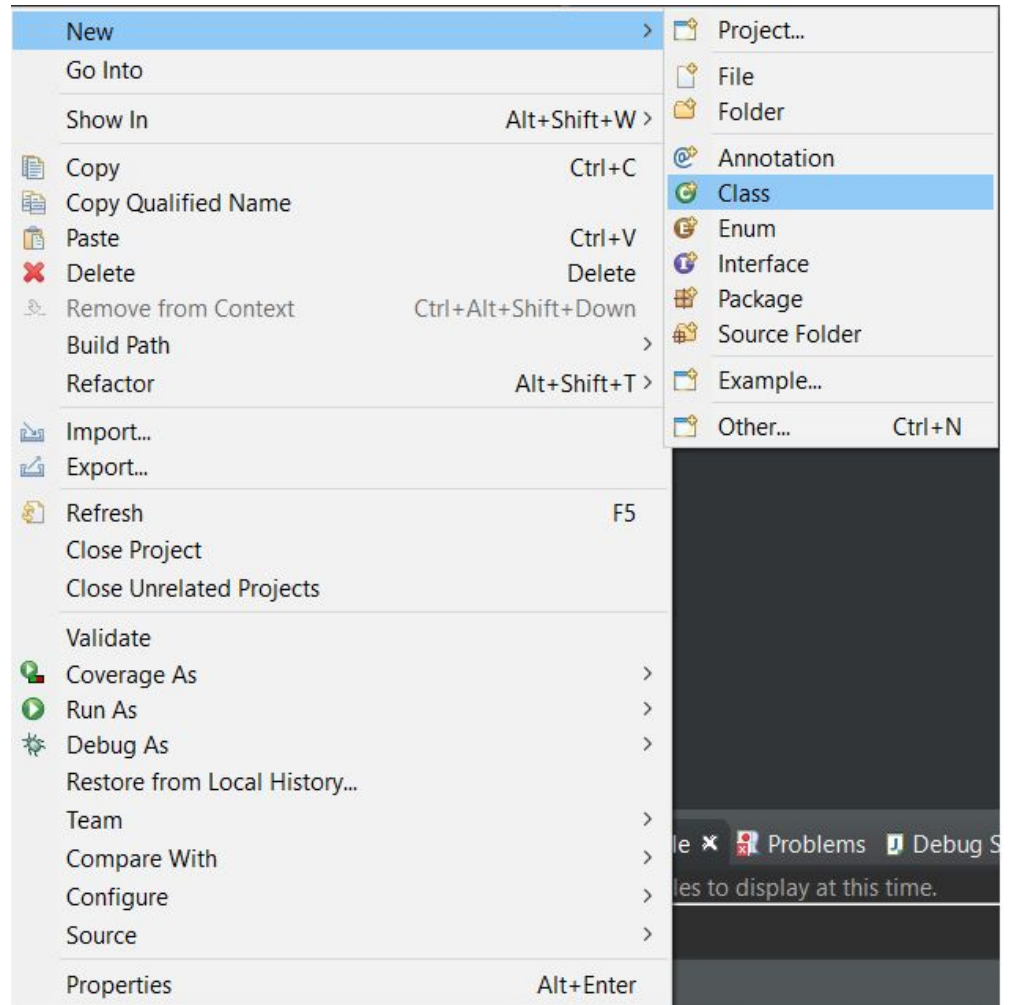
assertTrue(): Comprueba si una condición se cumple. Devuelve `assertionError` si no se produce el resultado esperado

assertFalse(): Comprueba si una condición no se cumple. Devuelve `assertionError` si no se produce el resultado esperado

fail(): devuelve una alerta informando del fallo en el test


Creamos un nuevo Java Project

Y dentro de éste, creamos una nueva clase



New Java Class

Java Class

 The use of the default package is discouraged.



Source folder:

[Browse...](#)

Package: (default)

[Browse...](#)

☐ Enclosing type:

[Browse...](#)

Name:

Modifiers: ☐ public ☐ package ☐ private ☐ protected

☐ abstract ☐ final ☐ static

Superclass:

[Browse...](#)

Interfaces:

[Add...](#)

[Remove](#)

Which method stubs would you like to create?

☐ `public static void main(String[] args)`

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

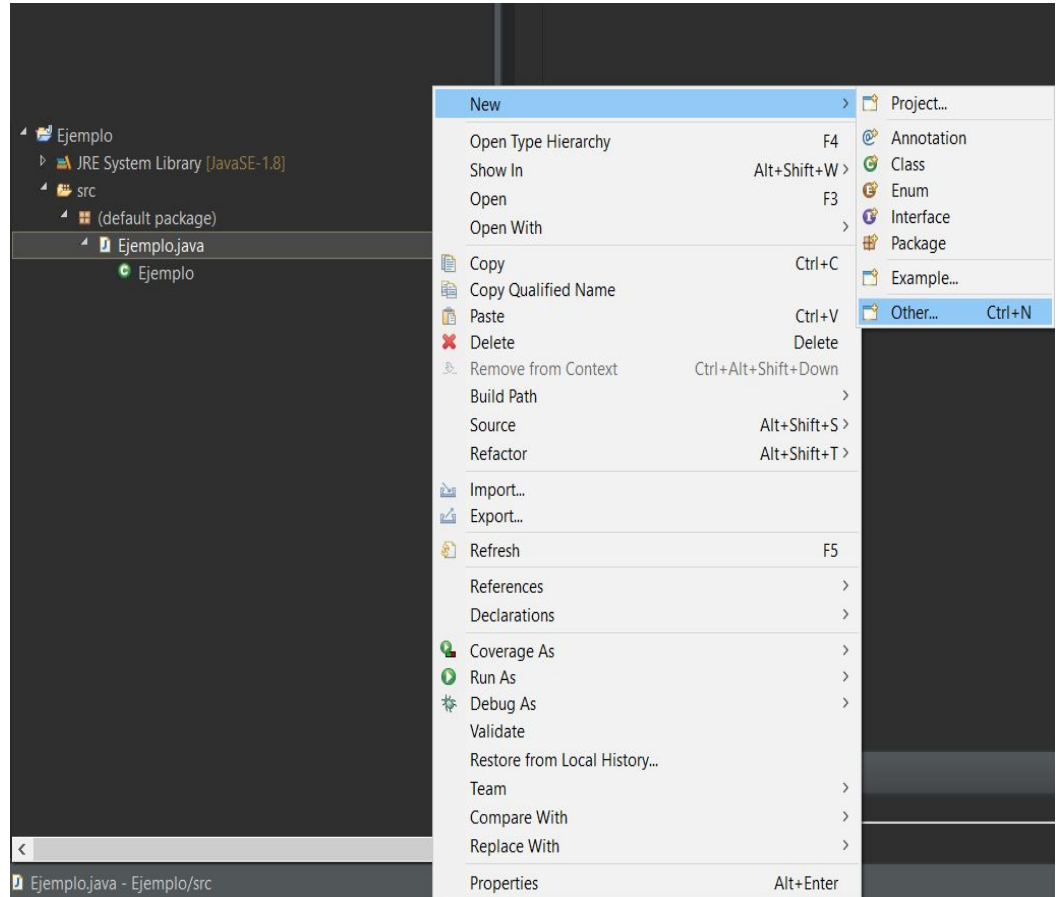


[Finish](#)

[Cancel](#)

JUnit test case

Después de crear la clase,
creamos un caso de Test de
JUnit.



New

Select a wizard

Create a JUnit Test Case



Wizards:

- Java
 - Annotation
 - Class
 - Enum
 - Interface
 - Java Project
 - Java Project from Existing Ant Buildfile
 - Java Working Set
 - Package
 - Source Folder
 - Java Run/Debug
 - JUnit
 - JUnit Test Case
 - JUnit Test Suite
- Maven
- Oomph



< Back


Next >

Finish

Cancel

New JUnit Test Case

JUnit Test Case

 The use of the default package is discouraged.

☒ New JUnit 3 test

☒ New JUnit 4 test

☐ New JUnit Jupiter test

Source folder:

Ejemplo/src

Browse...

Package:

(default)

Browse...

Name:

EjemploTest

Superclass:

java.lang.Object

Browse...

Which method stubs would you like to create?

☒ setUpBeforeClass()

☒ tearDownAfterClass()

☒ setUp()

☒ tearDown()

☐ constructor


Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Class under test:

Ejemplo

Browse...



< Back

Next >

Finish

Cancel

```
Ejemplo.java x EjemploTest.java x
1 import static org.junit.jupiter.api.Assertions.*;
2
3
4
5
6
7
8
9 class EjemploTest {
10
11     @BeforeAll
12     static void setUpBeforeClass() throws Exception {
13     }
14
15     @AfterAll
16     static void tearDownAfterClass() throws Exception {
17     }
18
19     @BeforeEach
20     void setUp() throws Exception {
21     }
22
23     @AfterEach
24     void tearDown() throws Exception {
25     }
26
27     @Test
28     void test() {
29         fail("Not yet implemented");
30     }
31 }
32
33
```

FIN

Trabajo realizado:

Juan Carlos Durán Caballero

José Luis Rubio Alcalde

Jose Antonio Del Cueto Gonzalez

Antonio Borja Morón Pozo