

Prueba de programas Java mediante JUnit

¿Qué es una prueba de unidad?

La prueba de programas es una aproximación dinámica a la verificación de los mismos, en la cual se ejecuta el software con unos datos o casos de prueba para analizar el buen funcionamiento de los requisitos esperados del programa. Podemos decir que la prueba es un tipo de verificación que se basa en la ejecución del código.

La prueba de unidad es la prueba de un módulo concreto dentro de un software que incluirá muchos otros módulos. Un módulo es una unidad de código que sirve como bloque de construcción para la estructura física de un sistema. El concepto de “módulo” se debe interpretar en términos del lenguaje de programación que estemos utilizando. Por ejemplo, podemos considerar que una clase Java o C++ es un módulo, pero también lo es un módulo del lenguaje Modula-2 o una unidad en TurboPascal.

Una prueba de unidad es un programa que prueba a otro programa. En términos de Java, es una clase con el objetivo de probar a otra clase.

En los últimos años, aparecido una metodología de desarrollo de programas denominada **Extreme Programming (XP)** (Beck 1999), que hace un énfasis especial en las prácticas y técnicas de prueba unitaria. Como resultado de ese énfasis, se han creado una serie de frameworks para ayudar a realizar pruebas unitarias en diferentes lenguajes. Al conjunto de esos frameworks se les denomina xUnit.

Nos centramos en JUNIT, que es “el XUNIT para Java”, los conceptos son muy similares a los que nos encontramos en los frameworks XUNIT para otros lenguajes (incluyendo a C++, Delphi, Smalltalk y Visual Basic).

Object
└─Assert

protected	Assert ()
public static void	assertEquals (String message, Object expected, Object actual)
public static void	assertEquals (Object expected, Object actual)
public static void	assertEquals (String message, <i>[double float]</i> expected, <i>[double float]</i> actual, <i>[double float]</i> delta)
public static void	assertEquals (<i>[double float]</i> expected, <i>[double float]</i> actual, <i>[double float]</i> delta)
public static void	assertEquals (String message, <i>[boolean byte char int long short]</i> expected, <i>[boolean byte char int long short]</i> actual)
public static void	assertEquals (<i>[boolean byte char int long short]</i> expected, <i>[boolean byte char int long short]</i> actual)
public static void	assertNotNull (String message, Object object)
public static void	assertNotNull (Object object)
public static void	assertNull (String message, Object object)
public static void	assertNull (Object object)
public static void	assertSame (String message, Object expected, Object actual)
public static void	assertSame (Object expected, Object actual)
public static void	assertTrue (String message, boolean condition)
public static void	assertTrue (boolean condition)
public static void	fail (String message)
public static void	fail ()

Método	Qué hace
assertTrue(boolean condicion)	Falla si la condición es false; pasa en otro caso.
assertEquals(Object esperado, Object actual)	Falla si esperado y actual no son iguales, según el método equals(); pasa en otro caso.
assertEquals(int esperado, int actual)	Falla si esperado y actual no son iguales, según el operador ==; pasa en otro caso. Este método está sobrecargado para cada tipo primitivo: int, float, double, char, byte, long, short, y boolean.
assertSame(Object esperado, Object actual)	Falla si esperado y actual se refieren a diferentes objetos en memoria; pasa si se refieren al mismo objeto en memoria.
assertNull(Object object)	Pasa si es null; En otro caso falla.

Además están los métodos `assertFalse()`, `assertNotSame()` y `assertNotNull()`.

Para aritmética con punto flotante :

`assertEquals([String message], expected, actual, tolerance)`

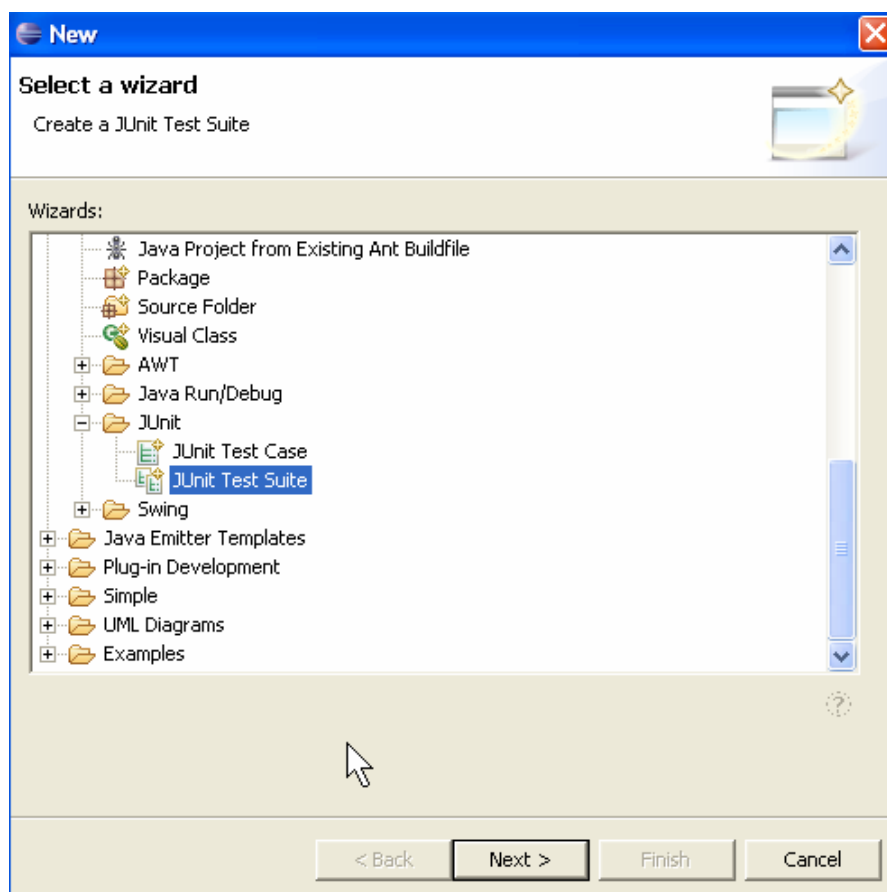
Ejemplo: `assertEquals (expectedDouble, actualDouble, 0.0001d).`

Pruebas JUnit en Eclipse

Eclipse facilita la tarea de crear y ejecutar pruebas unitarias utilizando el framework JUnit [www.junit.org]. Para poder utilizar el framework JUnit, es necesario colocar la librería junit.jar en el classpath del proyecto.

El plugin JDT incluye un wizard (asistente) para la creación de casos de prueba (JUnit Test Cases) muy similar al propio wizard de creación de clases, explicado en este mismo documento.

Crear un nuevo TestCase, como decía, es muy similar a crear una nueva clase. De igual forma, se puede utilizar el botón de creación de clases (en la barra de herramientas principal, con la Perspectiva Java activa). En el menú desplegable que se muestra, en lugar de seleccionar Class o Interface, como se había explicado, se debe seleccionar la opción TestCase, lo cual mostrará el wizard JUnit.



Toggle Class Load Breakpoint

Run As

Debug As

Team

Compare With

Replace With

1 Java Bean

2 JUnit Test Alt+Shift+X, T

Run...

Package Explorer Hierarchy JUnit

Finished after 0,016 seconds

Runs: 5/5 Errors: 0 Failures: 3

Failures Hierarchy

- com.mp.practica2.ejercicio4.test.MayorTest
 - testNormal
 - testOrden
 - testNegativo
 - testDuplicados
 - testSoloUno

Failure Trace

```
junit.framework.AssertionFailedError: expected: <-7> but was: <-8>  
at com.mp.practica2.ejercicio4.test.MayorTest.testNegativo(MayorTest.java:3)  
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java)  
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccesso
```

New JUnit Test Suite

JUnit Test Suite

Warning: Test suite already exists, suite() method will be replaced.

Source folder:

Package:

Name:

Test classes to include in suite:

0 classes selected

Would you like to create a method stub for main?

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

☐ Add TestRunner statement for:

```

package com.mp.todostests;

import com.mp.practical1.ejercicio2.test.LeerDatosTest;
import com.mp.practica2.ejercicio1.test.FigurasTest;
import com.mp.practica2.ejercicio4.test.MayorTest;
import com.mp.practica3.ejercicio1.test.SolucionesPolinomioTest;
import com.mp.practica3.ejercicio2.test.SolucionesSubsecuenciaTest;

import junit.framework.Test;
import junit.framework.TestSuite;

public class AllTests {

    public static Test suite() {
        TestSuite suite = new TestSuite("Test for com.mp.todostests");
        // $JUnit-BEGIN$
        //suite.addTestSuite(LeerDatosTest.class);
        suite.addTestSuite(FigurasTest.class);
        suite.addTestSuite(MayorTest.class);
        suite.addTestSuite(SolucionesPolinomioTest.class);
        suite.addTestSuite(SolucionesSubsecuenciaTest.class);

        // $JUnit-END$
        return suite;
    }
}

```