Junit 5 es la nueva versión del conocidísimo framework para la automatización de pruebas en Java. Ahora mismo se encuentra en su Milestone 2 (publicada el 23 julio de 2016).

El framework provee al usuario de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en su sistema y así asegurar su consistencia y funcionalidad.

Métodos.

A grandes rasgos, una clase de Prueba realizada para ser tratada por JUnit tiene una estructura con 4 tipos de métodos:

Método setUp: Asignamos valores iniciales a variables antes de la ejecución de cada test. Si sólo queremos que se inicialicen al principio una vez, el método se debe llamar "setUpClass"

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

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

Métodos auxiliares.

Anotaciones.

Se llama anotaciones a caracteres especiales, que se han incluido en la versión 4, para intentar simplificar más la labor del programador. Se trata de palabras clave que se colocan delante de los métodos definidos antes y que indican a las librerías JUnit instrucciones concretas.

A continuación, pasamos a ver las más relevantes:

@RunWith: Se le asigna una clase a la que JUnit invocará en lugar del ejecutor por defecto de JUnit

@Before: Indicamos que el siguiente método se debe ejecutar antes de cada test (precede al método setUp). Si tiene que preceder al método setUpClass, la notación será "@BeforeClass"

@After: Indicamos que el siguiente método se debe ejecutar después de cada test (precede al método tearDown). Si tiene que preceder al método tearDownClass, la notación será "@AfterClass"

@Test: Indicamos a Junit que se trata de un método de Test. En versiones anteriores de JUnit los métodos debían tener un nombre con la siguiente estructura: "Test". Con esta notación colocada delante de los métodos podemos elegir el nombre libremente.

Funciones de aceptación/rechazo.

Una vez hemos creado las condiciones para probar que una funcionalidad concreta funciona es necesario que un validador nos diga si estamos obteniendo el resultado esperado o no. Para esta labor se definen una lista de funciones (incluidas en la clase Assert) que se pueden ver detalladas en el javadoc de JUnit. Se detallan las más comunes:

sertArrayEquals: Recibe como parámetro dos arrays y comprueba si son iguales. Devuelve assertionError si no se produce el resultado esperado

sertEquals: Realiza la comprobación entre dos valores de tipo numérico. Devuelve assertionError si no se produce el resultado esperado

sertTrue: Comprueba si una condición se cumple. Devuelve assertionError si no se produce el resultado esperado

il: devuelve una alerta informando del fallo en el test