Seminario - parte 3

Junit: test unitarios

PROGRAMACION 3

David Rizo, Felipe Sánchez, Pedro J. Ponce de León
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

Contenidos

David Rizo, Felipe Sánchez, Pedro J. Ponce de León

Pruebas unitarias

- Una **prueba unitaria** es un fragmento de código que verifican un caso concreto de uso de un componente software según las especificaciones.
- Cada prueba se configura para probar un caso determinado de uso de la interfaz de una clase.
- Las pruebas se organizan en conjuntos o **suites** de pruebas. Cada 'suite' se asocia a una clase.
- Se prueban, por ejemplo, condiciones o valores límite en argumentos de métodos, o condiciones bajo las que un método genera excepciones.

David Rizo, Felipe Sánchez, Pedro J. Ponce de León

JUnit

- La herramienta más usada en Java para pruebas unitarias es **JUnit**.
- En Eclipse se configura en `Project > Properties > Java Build Path > Libraries > Add Library`

Actividad

Configura tu proyecto para que use *JUnit 4*.

David Rizo, Felipe Sánchez, Pedro J. Ponce de León

JUnit

- Separamos los ficheros de los tests unitarios del resto de código fuente
- Creamos directorio de código fuente `test` en el proyecto pulsando sobre éste en la vista de paquetes y pulsando `New > Source folder`
- Creamos el paquete *model* dentro de `test` (los archivos de código que contienen las pruebas pertenecen también al paquete `model`). Pega ahí los ficheros de pruebas de la práctica.
- Actualiza el proyecto en Eclipse (F5) y sube los cambios a Github.
- La ejecución de las pruebas se realiza pulsando sobre la clase que las contiene con el botón derecho y seleccionando `Run as > JUnit test`.
- También puedes depurar un test, mediante `Debug as > JUnit test`

David Rizo, Felipe Sánchez, Pedro J. Ponce de León

JUnit

Actividad

- Abrir el test unitario *CoordinatePreTest*
- Los métodos con anotaciones `@Before` configuran el test. Se ejecutan antes de cada método `@Test`.
- Los métodos `@Test` contienen una o más pruebas unitarias.
- `assertEquals` comprueba que el valor esperado coincide con el real. Los parámetros son por este orden: título (opcional), valor esperado, valor real, diferencia en valor absoluto permitida (opcional, útil para los reales) .
- `assertTrue, assertFalse` comprueban que su argumento devuelve `true` o `false`, respectivamente.
- `fail` produce un fallo en el test cuando es ejecutado.

David Rizo, Felipe Sánchez, Pedro J. Ponce de León

JUnit

Actividad

- Ejecuta los test: `Run / Debug as... -> JUnit Test` sobre el archivo de los test (los que contienen instrucciones `fail` fallarán). Se abre la pestaña `JUnit` donde podemos ver el resultado de la ejecución.
- Selecciona un test que falle. En el panel `Failure trace` haz doble click sobre la primera línea que indique `at model.CoordinatePreTest ...`. Te llevará a la línea que produjo el error.
- Modifica algún valor esperado en un test que funciona. Ahora fallará y seleccionando el test en el panel `Failure trace` podrás ver la causa en la primera línea.

David Rizo, Felipe Sánchez, Pedro J. Ponce de León

Nuevo test unitario

Para generar un nuevo test unitario sobre una clase, pulsar con el botón derecho sobre ésta en la vista de paquetes, y seleccionar `New > JUnit test case`.

- Seleccionar JUnit 4.
- En el directorio, seleccionar `test` en lugar de `src`.

Actividad

- Implementa un método en `Coordinate` que devuelva la suma de las componentes.
- Crea un test unitario (o varios) para comprobar que el método funciona correctamente.
- Para ejecutar todos los tests podemos pulsar con botón derecho sobre el proyecto y seleccionar `Run as > JUnit test`.
- También puedes ejecutar alguno en concreto (o sólo los que han fallado)
- Elimina el método cuando hayas terminado con esta actividad.

David Rizo, Felipe Sánchez, Pedro J. Ponce de León

Test unitarios pre-publicados

- En *CoordinatePreTest* comprobarás que hay algunos test sin implementar.
- En los comentarios de cada uno se especifica qué debe hacer ese test.
- Impleméntalos sirviéndote de los test ya implementados como ejemplo.
- ¡Hazlo bien! Esos test (y algunos más) los usaremos para la corrección de la práctica. Piensa si falta algo por probar y añade tus propios tests o amplia los que hay.

David Rizo, Felipe Sánchez, Pedro J. Ponce de León