P3- Automatización de las pruebas: Drivers

Implementación de *drivers* con JUnit

El objetivo de esta práctica es automatizar el resultado del diseño de casos de prueba (tabla de casos de prueba) que hemos obtenido en la práctica anterior. Tal y como hemos visto en clase, una vez que hemos diseñado las pruebas, necesitamos ejecutar el elemento a probar, que hemos denominado SUT, utilizando los datos de entrada especificados en la tabla, de forma que obtenemos el resultado real y comprobamos si el dicho resultado coincide con el esperado, para finalmente emitir un informe. El software encargado de ejecutar el SUT y finalmente emitir el informe se denomina driver o conductor de pruebas.

En esta sesión, por lo tanto, vamos a implementar drivers para los casos de prueba de la práctica anterior (implementaremos un driver para cada caso de prueba). Utilizaremos JUnit para implementar los tests, y posteriormente utilizaremos Maven para construir el proyecto y lanzar la ejecución de dichos tests, con el objetivo de detectar defectos en las SUT correspondientes. De momento, nuestro código a probar, o SUT, hace referencia a una unidad (método de una clase Java), sobre la que realizamos pruebas dinámicas.

Bitbucket

El trabajo de esta sesión también debes subirlo a Bitbucket. Todo el trabajo de esta práctica, tanto el código fuente, como cualquier otro documento con vuestras notas de trabajo, deberán estar en la carpeta **P3** de vuestro repositorio**.**

Creación de proyectos Maven con Netbeans

En esta sesión tendréis que crear un nuevo proyecto Maven con Netbeans (File-> New Project-> Maven -> Java Application). El nuevo proyecto creado se llamará "**practica3**". Asegúrate de que lo creas en la carpeta **P3**. La coordenada groupId debe ser "ppss". En la ventana para crear el nuevo proyecto os preguntará cuál será el nombre del paquete en el que estará vuestro código fuente (campo de texto "Package"). Poned como nombre del paquete "ppss"**.**

Observaremos que nos crea la estructura de directorios estándar de Maven, y un fichero pom.xml, qué únicamente contiene las coordenadas del proyecto y la definición de algunas propiedades.

Para implementar los tests debemos modificar el pom.xml y añadir la librería JUnit en la sección de dependencias del pom, tal y como hemos visto en clase (trabajaremos con la versión 4.12 de JUnit),.

También podemos observar que NO nos ha creado la carpeta /src/test (si estamos en la ventana Projects veremos que la carpeta "Test Packages" no está). Esta carpeta se creará de forma automática cuando empecemos a implementar los *tests* (cuando hablamos de implementar un test nos estamos refiriendo a implementar los *drivers*).

Ejercicios

1. **Drivers para calculaTasaMatricula().** Dada la tabla de casos de prueba que obtuvisteis para el método Matricula.calculaTasaMatricula(), se pide lo siguiente:

Nota: En caso de que no dispongas de la tabla de casos de prueba, puedes utilizar la siguiente:

	Datos de entrada			Resultado esperado
	edad	familia numerosa	repetidor	tasa
C1	20	falso	cierto	2000
C2	70	falso	cierto	250
C3	20	cierto	cierto	250
C4	20	falso	falso	500
C5	60	falso	cierto	400

A) Implementa la clase Matricula con el método calculaTasaMatricula() (éste será nuestro SUT, utiliza el código del enunciado de la práctica anterior). Ahora implementa los drivers correspondientes para automatizar dicha tabla de casos de prueba (utiliza para ello la clase MatriculaTestSinParametros). Usa fixtures, y sentencias assert, y no utilices tests con parámetros.

Creación de tests Junit con Netbeans

Para crear la clase con las implementaciones de los drivers utiliza el menú contextual del proyecto, y selecciona la opción New-> JUnit Test. Si no te aparece la opción "JUnit Tests" en el menú la primera vez, selecciona Others->Unit Tests -> JUnit Test. Se abrirá una ventana, en la que te solicitará el nombre de la clase que va a contener los drivers, y por defecto te aparecerá indicada la ubicación "Test Packages". Recuerda que debes indicar el nombre del paquete al que pertenecerán los tests, así como la ubicación física de los mismos, siguiendo las indicaciones de clase. En el contexto de JUnit, se denomina test JUnit a cada una de las clase que contienen los drivers (métodos anotados con @Test).

Los drivers deben tener el sufijo "testCx". Siendo Cx el identificador de cada caso de prueba en la tabla, en función del camino independiente que se está probando en el driver correspondiente. Si por ejemplo la tabla de casos de prueba tiene 5 filas identificadas como C1, C2, ..., C5, deberéis implementar 5 tests que tengan como sufijo "testC1", "testC2",... "testC5".

Importante!!

Antes de continuar con el ejercicio, asegúrate de que la opción "Compile On Save" del proyecto está DESACTIVADA. Dicha opción la encontrarás en las propiedades del proyecto, desde Build->Compile

Una vez implementados los drivers, ¿Qué artefactos y dónde se generan si ejecutas la fase "compile"? ¿Qué artefactos y dónde se generan si ejecutas la fase "test-compile"? ¿Qué artefactos y dónde se generan si ejecutas la fase "test"? Verás que al invocar la fase "test" NO se ha ejecutado ningún driver. ¿Por qué pasa esto? Arregla el problema para que se ejecuten efectivamente los tests de forma automática al invocar la fase test de Maven.

Ejecuta ahora la fase *clean* de Maven (o borra directamente el directorio target). Ejecuta la goal *surefire:test*. Explica lo que ocurre.

B) Implementa en otra clase con nombre TestMatriculaConParametros los mismos drivers pero utilizando tests con parámetros. Ejecuta los tests utilizando la fase de maven correspondiente.

2. **Drivers para validaNif().** Dada la tabla de casos de prueba que obtuvisteis para el método Alumno.validaNif(), se pide lo siguiente:

   Nota: En caso de que no dispongas de la tabla de casos de prueba, puedes utilizar la siguiente:

	Datos de entrada	Resultado esperado
	nif	válido?
C1	"123"	falso
C2	"1234567AA"	falso
C3	"-12345678"	falso
C4	"00000000X"	falso
C5	"00000000T"	cierto

   A) Implementa la clase Alumno, con el método validaNif() utilizando el código del enunciado de la práctica anterior. Implementa los drivers correspondientes para automatizar la tabla de casos de prueba . A continuación implementa los drivers correspondientes para automatizar dicha tabla de casos de prueba (utiliza para ello la clase TestAlumnoSinParametros). Usa fixtures, y sentencias assert, y no utilices tests parametrizados, siguiendo la misma convención de nombres para los test que hemos indicado en el ejercicio anterior.

   B) Ejecuta los drivers. Si detectas algún defecto en la SUT que estás probando, depúralo, de forma que todos los tests estén en verde.

   C) Implementa en otra clase (puedes llamar a la clase TestAlumnoConParametros) los mismos drivers pero utilizando tests parametrizados

3. **Drivers para realizaReserva().** Razona qué tiene de diferente la implementación de este método, que hace que sea insuficiente disponer del código del método para automatizar las pruebas unitarias sobre dicho código implementando los drivers correspondientes. Si intentas implementar los drivers (no es necesario que lo hagas) verás que no podrás ejecutar las pruebas correctamente (al menos, teniendo en cuenta lo que hemos explicado hasta ahora en clase).

4. **Categorías.** Crea las categorías "ConParametros", y "SinParametros" de forma que podamos ejecutar de forma selectiva los tests parametrizados, o los tests sin parámetros. Ejecuta los tests con parámetros y posteriormente realiza otra construcción con maven, pero ejecutando sólo los tests sin parámetros. En clase hemos explicado dos formas de hacerlo. Razona ventajas e inconvenientes de hacerlo según cada una de ellas.

   Nota: Para ejecutar los tests sin modificar el pom, puedes hacerlo desde el menú contextual del proyecto, con la opción Custom->Goals. Tendrás que indicar las propiedades a utilizar desde el cuadro de texto "Properties". Ten en cuenta que la sintaxis desde Netbeans es **nombrePropiedad=valorPropiedad**, es decir, no se usa el "-D" ni se ponen comillas. Si marcas la casilla "Remember as", y asignas un nombre, te aparecerá el nuevo comando en el menú contextual desde la opción Custom.. Prueba a ejecutar el comando que hemos visto en clase pero desde un terminal. Deberás obtener el mismo resultado.