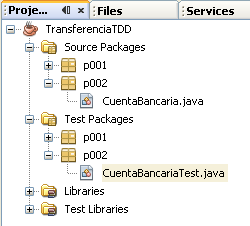
**Aplicación transferencia bancaria usando TDD**

**Parte 2**

Bien, llegados a este punto vamos crear un nuevo paquete que llamaremos p002, tanto en la rama ‘Source Packages’ como en la rama ‘Test Packages’, y copiaremos las respectivas clases. Esto lo hacemos para ir teniendo una versión de cada estadio de nuestro desarrollo TDD. Por supuesto, es algo que hacemos con fines académicos y no lo haríamos en un entorno profesional.

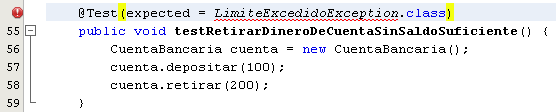
Una vez hecho lo anterior deberíamos tener una disposición de paquetes y clases como la siguiente:



A partir de ahora ya nos podemos olvidar del contenido de p001.

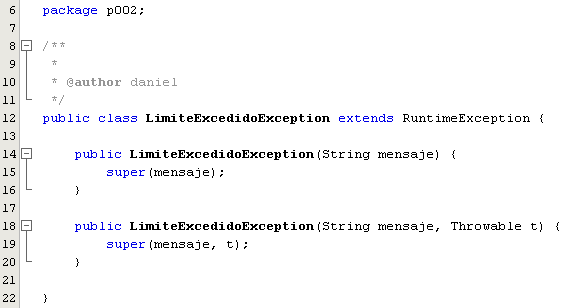
Para estrenar p002, vamos a añadir un test mediante el que queremos comprobar que si se efectúa un reintegro mayor que la cantidad depositada la aplicación lanzará una excepción de negocio, es decir, una clase programada por nosotros que hereda de la jerarquía de clases Exception de Java.

El código es el siguiente:



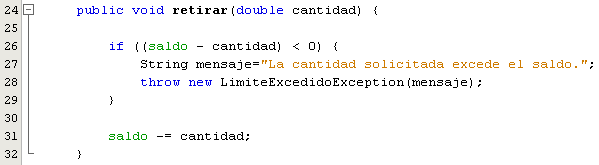
Notad la sintaxis que se emplea en JUnit para indicar el hecho de que el método del SUT debe lanzar una determinada excepción. Notad también que en este caso no es necesario que el método finalice realizando ningún tipo de assert.

Por otra parte necesitamos crear la excepción que esperamos que lance el método retirar() del SUT. Por tanto, creamos la siguiente clase de excepción en paquete p002 de la rama ‘Source Package’, junto a la clase CuentaBancaria:

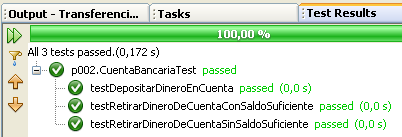


Notad que la clase extiende de RuntimeException. Con esto hacemos que la excepción no sea verificada y por tanto, evitamos que quien llame al método retirar() tenga que tratar la excepción.

Por último, tenemos que modificar el SUT para que lance la excepción en caso de que la cantidad a retirar exceda el saldo de la cuenta:

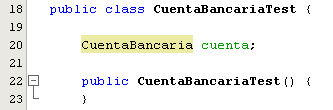


Ahora podemos ejecutar los test:

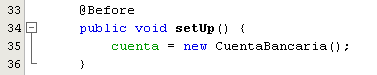


Ahora podría ser un buen momento para hacer algunas mejorar en el código de los test. Si nos fijamos, cada método crea un objeto CuentaBancaria, lo que sin ser malo es un poco redundante. Podemos declara un atributo de instancia de este tipo e inicializarlo en el método setUp() -anotado con @Before-

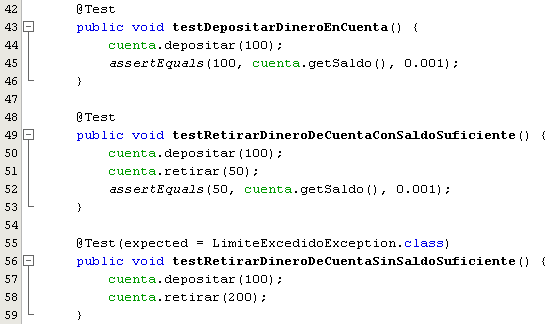
Por tanto, creamos la propiedad cuenta:



Creamos la instancia en el método setUp(), dado que sabemos que se ejecutará antes de ejecutar cada nuevo test:

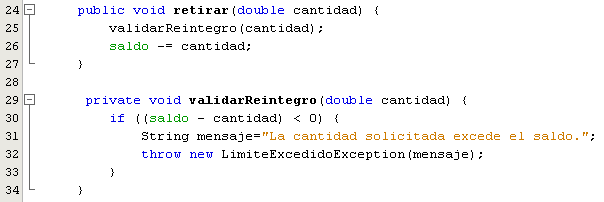


Y ya podemos eliminar de cada uno de los test la línea que creaba e inicializaba cada cuenta:



Otra mejora que podemos hacer consiste en separar las diferentes responsabilidades que observemos que se estén realizando en un mismo método. Esto hará que el método sea más sencillo de leer y de modificar. En nuestro caso, el método retirar() del SUT está haciendo dos cosas: comprobando si hay saldo suficiente para la cantidad solicitada y, en caso afirmativo, restando esa cantidad al saldo.

Por tanto, extraemos la responsabilidad de la comprobación y la encapsulamos en un método privado. Entonces hacemos que el método retirar() invoque al método de comprobación antes de realizar la resta:



Una vez hecho lo anterior, volvemos a ejecutar los test para descartar cualquier posible problema.