Práctica 3

Documentación mediante UML y Javadoc y pruebas con JUnit y EclEmma

Grado Ingeniería Telemática
GSyC, Universidad Rey Juan Carlos

1. Introducción

Esta práctica tiene como objetivo familiarizarse con la correcta codificación, documentación y pruebas de código. Para ello, se hará uso de diagramas *UML*, documentación *Javadoc*, pruebas unitarias con *JUnit* y comprobación de pruebas con *EclEmma*.

2. Diagramas *UML*

El lenguaje unificado de modelado (*UML*[1], *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y estandarizado. Se compone de varios elementos gráficos que se combinan para confeccionar los diagramas. Es un lenguaje; por tanto, cuenta con reglas para combinar tales elementos.

El fin de un diagrama es ilustrar las diferentes perspectivas (o modelos) de un sistema. Así, un modelo es una representación simplificada de la realidad. De ahí que un modelo UML describa lo que hace un sistema, pero no cómo implementarlo.

Existen diversos diagramas UML, según los conceptos a representar. Nosotros nos centraremos en los **Diagramas de Clases**. Éstos describen la estructura estática de un sistema mediante rectángulos que representan a las clases, conectados por líneas que representan las asociaciones que existen entre ellas (composición, herencia, etc.).

Para realizar un diagrama UML se puede emplear un simple editor gráfico, pero resulta más conveniente y, sobre todo, más sencillo, usar una herramienta propia de diseño UML. Existen numerosas de estas aplicaciones; en nuestro caso, usaremos `UMLet`, una herramienta potente, pero a la vez sencilla e intuitiva. Además, su instalación es muy simple, pues está disponible en el sistema de paquetes de cualquier sistema Linux.

3. *Javadoc*

Para documentar el detalle de las distintas clases que conforman el programa usaremos la funcionalidad de `Javadoc`, que nos ofrece el lenguaje `Java`.

Para generar Javadoc con Eclipse basta con dirigirse a *Project - Generate Javadoc... - Use standard doclet*. De este modo se generarán distintos ficheros *.html* que contendrán la documentación de las clases.

No hay que olvidar que sólo se generará una buena documentación si hemos empleado las palabras reservadas que ofrece `Java` para añadir `Javadoc` en el código implementado.

[1] www.uml.org

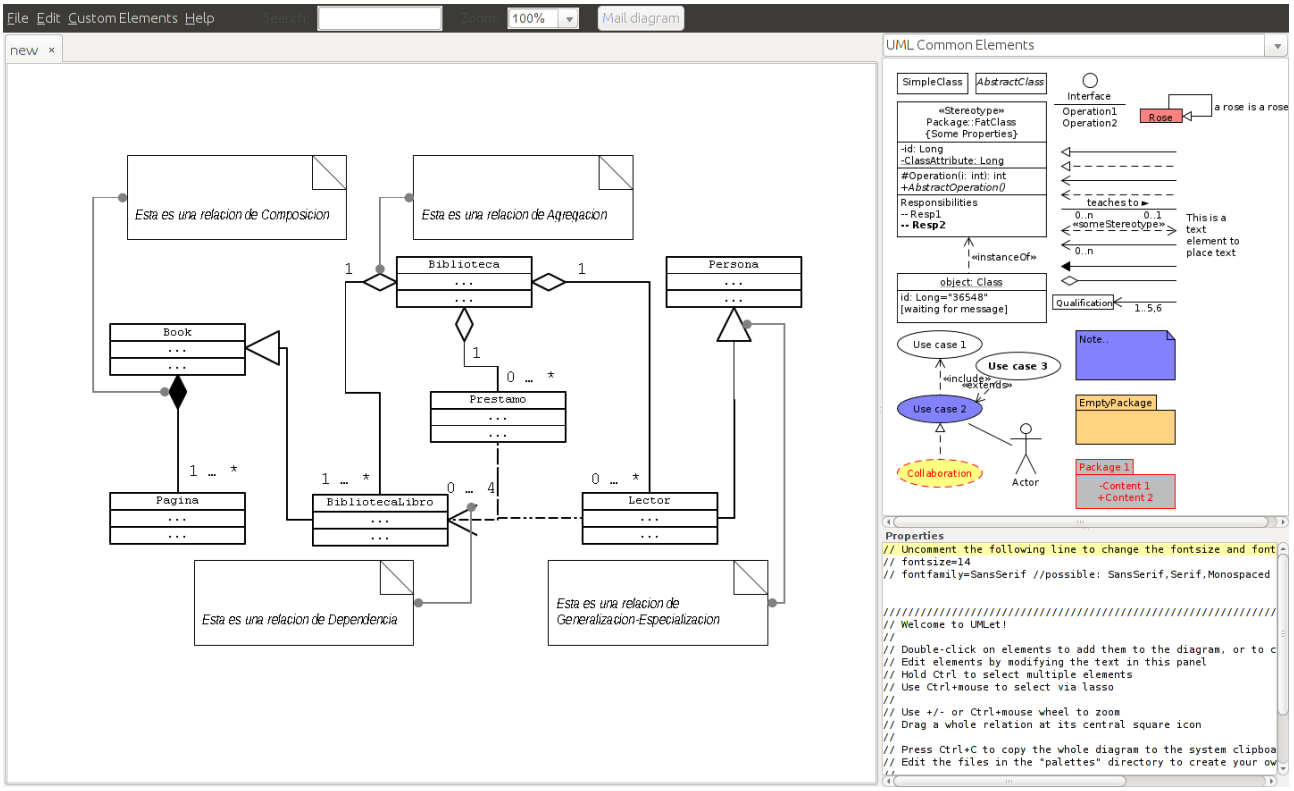

Figura 1: Ejemplo con herramienta UMLet

4. *JUnit* y *EclEmma*

Como ya hemos visto en clase, estas dos herramientas van de la mano. La primera sirve para hacer pruebas unitarias del código; la segunda, para comprobar de forma amena qué se ha probado y qué no.

4.1. Pruebas unitarias con JUnit

Usa el código `JUnitBasico` que se facilita adjunto en esta práctica para adquirir soltura con el uso `JUnit`. Los pasos a seguir para usar este *plug-in* son los siguientes:

1. Creamos proyecto y añadimos la clase que queremos probar: `Suma.java`.
2. Añadimos ahora la clase que nos servirá para probar la clase Suma: `SumaTest.java`.
3. Veremos que el código nos da errores porque no encuentra `@Test` y `assertTrue`, que son propios de la librería de `JUnit`. Hacemos clic en el icono de ese error y ya tenemos la opción de *Add JUnit 4/5 to the build path*.
4. También veremos errores en los `asserTrue`. Igualmente, hacemos clic en tal error, y *Add static import org.junit.Assert.**. Con esto ya estaría todo resuelto.
5. Activamos la vista `JUnit` en `Eclipse`. Para ello hay que pulsar en: *Window - Show View - Other... - Java - JUnit*.
6. Y finalmente ejecutamos el proyecto como test `JUnit`: botón derecho sobre el proyecto *Run as - JUnit Test*. Y ya podremos ver los resultados en la vista `JUnit`.

4.2. Comprobación de pruebas con EclEmma

Siguiendo con ese código de `JUnitBasico`, el siguiente paso para completar una buena fase de pruebas es usar el complemento `EclEmma`, para ver qué se ha probado y qué no.

Esta herramienta suele venir ya por defecto con `Eclipse`; no obstante, si necesitamos instalarla tenemos varias vías: desde `Eclipse Marketplace`, desde la página de EclEmma (`https://www.eclemma.org`), y manualmente. Lo más sencillo es seguir la primera opción:

1. Menú *Help - Eclipse Marketplace.*
2. *Find: EclEmma.* Comprobar si ya está *installed*; si no, lo instalamos.

Para usar esta herramienta, veremos que nos aparece un botón *Coverage* (cobertura) junto al de *Run* y *Debug*. Si no, lo ponemos manualmente: *Window - Perspective - Customize Perspective.* Una vez ejecutado el proyecto de este modo, veremos que nos resalta en rojo aquel código que está sin probar; y en verde, lo que sí se está probando.

Ejercicios

Ejercicio 1. Diagrama UML de la Práctica 2

Haciendo uso de la herramienta `UMLet`, documenta la arquitectura de los paquetes resultantes de la práctica 2. Se podrán emplear para ello uno o varios diagramas de clases UML. Si fuese preciso, se puede utilizar algún diagrama de secuencia para documentar alguna interacción compleja (normalmente no será necesario). Cada diagrama de clases debe documentarse explicando:

1. Lo que resuelve el diagrama.
2. Una breve descripción de las responsabilidades de cada clase.

Esto mejorará notablemente la calidad de la documentación de diseño. Es importante explicar sólo lo relevante de un diagrama de clases; es decir, lo necesario para poder entender el diseño.

Ejercicio 2. Documentación Javadoc de la Práctica 2

Comenta convenientemente la `Práctica 2` usando Javadoc. Recuerda que no es necesario poner comentarios a todo, porque esto ocasionará el efecto contrario al que se persigue, que es clarificar el código; pero sí resulta interesante centrarse en los siguientes puntos:

1. Cabecera del fichero-clase describiendo qué hace.
2. Cabecera de los métodos que consideres más peculiares.
3. En general, cualquier bloque de código que desees remarcar.

Ejercicio 3. Pruebas unitarias con JUnit y EclEmma

Usaremos como hemos visto `JUnit` para la implementación de casos de prueba unitarios relativos a los casos de uso mencionados en la práctica 2. Siguiendo el ejemplo visto en clase, haremos uso de las palabras clave que nos ofrece `Java` para estos casos de prueba (`@Test`, `assertTrue`, etc.).

Ve comprobando progresivamente con EclEmma qué está probado y qué no.

5. Entrega

Para la entrega de esta práctica, deja todo el código fuente, la documentación y los diagramas UML contenidos en un fichero `peticiones2.zip` que, como siempre, habrá que dejar adjunto en la tarea (Moodle).

Los diagramas UML resultantes se incluirán en la carpeta `doc` que genera el propio `Eclipse` al usar `Javadoc`. Y deben ser entregados en formato PDF (*File - Export as - PDF*).