DOCUMENTACIÓN

CODE UD5 – Tema 1

IES Plurilingüe Antón Losada Diéguez

Adrián Fernández González

Tabla de contenido

Documentación

1. Introducción

Una parte fundamental de todo sistema es la documentación, pues facilita su mantenimiento, modificación y actualización, así como su comprensión, tanto por miembros del propio equipo como por terceras personas.

La documentación incluye toda descripción del sistema, comentarios en el código, los manuales que se entregarán a los usuarios finales, los diagramas que describen cada clase, etc.

Existen dos tipos fundamentales de documentación, la externa, también denominados manuales y la interna, incluida en el propio código.

2. Documentación externa

La documentación externa o manuales son los documentos que van adjuntos con todo software y contienen la descripción detallada del sistema.

Estos manuales se dividen en dos tipos dependiendo de su utilidad, el de usuario y el técnico. El primero es el que describe cómo utilizar el sistema por parte de aquel que lo va a utilizar, es decir, el usuario final del mismo. El segundo, describe el funcionamiento interno, las tecnologías, las distintas partes que lo conforman para su mantenimiento y modificación.

Ambos manuales han de ser creados a la par que el sistema, ya sea a priori o en paralelo, pero no deben ser realizados a posteriori.

Mantener actualizada la documentación también es fundamental, pues su abandono puede acarrear problemas futuros.

Hay que tener en cuenta que los manuales han de ser lo más formal posible a la par que amigables y sencillos de leer para aquel que lo lea.

Es fundamental incluir imágenes descriptivas, sobre todo en los pasos a seguir, para que el lector comprenda fácilmente cada paso.

2.1. Manual de usuario

El manual de usuario o guía describe cómo ha de instalarse y usarse el sistema sin entrar en los detalles de su codificación.

El manual de usuario ha de abarcar puntos como la instalación, la creación de usuarios, el inicio de sesión, el uso de cada una de las partes del mismo, posibles requisitos técnicos… todo aquello que deba conocer el usuario.

Hay que tener en cuenta que, si el sistema va a ser utilizado por distintos perfiles, es decir, que no todo el mundo tendrá acceso a las mismas partes del sistema, ha de realizarse un manual de usuario para cada uno de dichos perfiles, de tal forma que cada uno conozca solo lo que ha de utilizar.

Aunque no hay un estándar, hay una serie de buenas prácticas que establecen qué debería incluir un manual de usuario.

2.1.1. Descripción funcional

Este apartado es básico e incluye:

- Los **requisitos** para usar el sistema, tanto del hardware como del software.
- Ha de especificar qué **conocimientos previos** ha de tener el usuario y el nivel de manejo de las tecnologías.
- **Descripción** breve del sistema, qué hace y qué no.
- Lo que se denomina **prefacio**, los documentos complementarios del manual.
- **Licencia** y derechos de autor.
- Toda aquella información necesaria para su uso.

2.1.2. Instalación

Este apartado es un paso a paso del proceso de instalación del sistema por parte del usuario.

Esto solo se incluye si el usuario tuviese que instalar algo él mismo. Si el software es un sistema en un servidor, por ejemplo, este apartado no sería necesario, pues el usuario ya tendría acceso al sistema.

2.1.4. Manual de uso

La sección fundamental del manual, pues incluye la descripción del uso de cada parte del sistema, su interfaz y funcionalidades.

Esta parte ha de ser detallada, fácil de seguir y leer. Ha de incluir imágenes, indicaciones de la interfaz y paso a paso para realizar cada operación.

Es recomendable incluir ejemplos realistas y no solo descripciones.

Es vital dividir de forma clara la información por secciones acordes a las funcionalidades o partes del sistema, de tal modo que sea fácil buscarlas y que el lector sepa a qué parte se refiere.

Algunos manuales dividen esto en dos partes:

1. **Manual de introducción:** La descripción de las funcionalidades básicas del sistema, sin entrar en detalles, simplemente para empezar a utilizar el sistema.
2. **Manual de referencia:** La descripción total y detallada del sistema, en caso de necesitar conocer a fondo alguna de las partes a utilizar.

2.1.5. Posibles errores y soluciones

Aunque no obligatorio, es recomendable incluir una sección con posibles errores y como solventarlos.

Estos errores pueden ir, por ejemplo, desde tener que habilitar ciertas utilidades antes de usarlas, posibles pasos que el usuario ha pasado por alto o funcionalidades que han de ser autorizadas previamente por un perfil de rango superior.

Esta sección puede incluirse directamente en el manual de uso, dentro del apartado correspondiente a cada funcionalidad.

2.1.6. Preguntas frecuentes

Esta sección no es obligatoria, pero si muy útil, pues incluye posibles dudas que podrían surgirle al usuario y su respuesta.

2.1.7. Anexos y bibliografía

Si fuese necesario, se incluirían todos aquellos documentos adicionales, bien dentro del mismo documento u otro, ya sea propios o externos.

Estos anexos también pueden estar en otro formato, papel, digital, disco, web, enlace, etc.

En este apartado también se incluye la bibliografía consultada, citada y/o necesaria.

2.2. Manual técnico

El manual técnico describe el sistema de forma interna y va dirigido al equipo que va a mantenerlo y actualizarlo.

Este manual ha de describir hasta el último detalle del sistema, su código, la jerarquía de archivos, el proceso para implantarlo, iniciarlo y cerrarlo, los mensajes de error e informes que pueda generar, etc.

El manual técnico incluye los diagramas de clase, de uso, los diagramas de la base de datos, etc. Todo lo que forme parte del sistema ha de incluirse en este manual.

Parte de este manual puede ser generado automáticamente por el propio software utilizado para crear el sistema, sobre todo cosas como los diagramas.

Dependiendo del ámbito del sistema, este manual podrá incluir informes previos, análisis, estudios, normativa, etc.

3. Documentación interna

La documentación interna es aquella que va incluida en el propio código.

3.1. Comentarios

Los comentarios son el tipo de documentación interna más básica y utilizada. Es el propio programador el que suele incluirla a medida que va programando para facilitar el entendimiento de los bloques de código, sobre todo aquellos extensos o más complicados, así como pequeñas decisiones tomadas.

3.2. Generadores de documentación

Hay múltiples utilidades que permiten generar documentación a partir de la documentación interna.

La mayoría comparten la forma en la que extraen esa documentación y el formato en el que el programador ha de describir el código para que el generador cree el documento.

4. Javadoc

Javadoc es un generador de documentación creado por Oracle (antes Sun Microsystems) para Java.

Los IDEs para Java disponen del generador incluido y son capaces de generar dicha documentación como documentos HTML.

Su estándar es igual o muy similar al de otros generadores, por lo que es extrapolable a otros lenguajes.

El formato Javadoc permite describir las clases, atributos y métodos, tanto para generar la documentación como para mostrar dicha información al programador mientras programa.

Además, Java permite exportar esta documentación como un archivo que puede ser importado por los IDEs junto al código facilitado, permitiendo anexionarlo al código del sistema para ser usado por terceros.

4.1. Inclusión y formato

Su uso es muy sencillo, tan solo es necesario estipular un comentario Javadoc antes de lo que se quiere describir. Estos comentarios empiezan por /** y terminan por */ y cada línea intermedia empieza por *. La primera y última línea solo contienen dichos caracteres.

En su interior se describe brevemente qué es lo que realiza, sus parámetros, que devuelve, etc. Toda aquella información relevante para comprender el funcionamiento.

La mayoría de los IDEs, al introducir la apertura y un salto de línea, autocompleta con cierta información sacada de la clase o método.

```
/**
* Clase que efectúa la lógica de la piscifactoría.
* @author Adrián
*/
public class Piscifactoria {
```

En este ejemplo puede verse una clase comentada con su autor.

```
/** Nombre de la piscifactoría **/
private String name;
```

En este otro, la descripción de un atributo. En este caso, se puede utilizar un comentario de una única línea, reservado solo para los atributos.

```java
/**
 * Constructor parametrizado
 *
 * @param num El numero de peces permitidos.
 * @param name El nombre del invernadero
 */
public Piscifactoria(int num, String name) {
    this.name = name;
    this.tank = new Pececillo[num];
    this.maxFood = num * 2;
}
```

Las descripciones más completas suelen ser las de los métodos, pues son los principales bloques de código que conforman el sistema.

Toda clase, método y atributo ha de estar comentado, independientemente de si es público o no.

Además de texto, los comentarios Javadoc permiten código HTML.

4.2. Etiquetas

Para completar la información textual, se utilizan una serie de etiquetas que el generador utilizará para dar formato a la descripción.

Estas etiquetas empiezan por @ el nombre de la etiqueta, un espacio y el texto a incluir.

Hay etiquetas exclusivas de ciertas estructuras, unas para clases, otras para métodos y otras generales.

4.2.1. Autor @author

La etiqueta ***@author nombre*** especifica el nombre del autor de la clase o interfaz.

Los IDEs la añaden por defecto, pues es una etiqueta obligatoria de toda clase e interfaz.

```java
/**
* Clase que efectúa la lógica de la piscifactoría.
* @author Adrián
* @version 1.0.0
*/
public class Piscifactoria {
```

En este ejemplo se puede ver el uso de la etiqueta *@author* y *@version* en una interfaz.

4.2.2. Versión @version

La etiqueta ***@version numero*** estipula la versión de la clase o la interfaz. Se utiliza para mantener un control de las diferentes versiones de las clases e interfaces y los cambios en el sistema.

El estándar indica que es obligatoria para toda clase e interfaz, pero ha de incluirse manualmente en muchos IDEs.

```java
/**
* Interfaz que representa un pez.
* @author Adrián
* @version 1.0.0
*/
public interface IPez {
```

En este ejemplo se puede ver el uso de la etiqueta *@author* y *@version* en una clase.

4.2.3. Parámetros @param

La etiqueta ***@param nombre descripción*** permite describir cada parámetro del método.

Su uso es obligatorio en todo método y los IDEs los añaden automáticamente por cada parámetro cuando se añade el Javadoc del mismo.

```java
/**
 * Constructor parametrizado
 *
 * @param num El numero de peces permitidos. Tiene que ser mayor de 0.
 * @param name El nombre del invernadero.
 */
public Piscifactoria(int num, String name) {
```

Como puede observarse, se añaden en el mismo orden en el que aparecen.

La explicación ha de ser concisa pero breve, indicando que representa cada parámetro, si tienen alguna restricción o valores por defecto.

4.2.4. Devolución @return

La etiqueta ***@return descripción*** permite describir lo que devuelve el método.

Su uso es obligatorio en todo método que devuelva algún valor y los IDEs los añaden automáticamente cuando se añade el Javadoc del mismo.

```java
/**
* Devuelve si esta vacío.
*
* @return True si está vacío, false en caso contrario.
*/
public boolean isEmpty()
```

Como se puede apreciar en el ejemplo, aunque la descripción del método ya especifique que devuelve, es necesario estipular la etiqueta *@return* por muy redundante u obvia que parezca.

Si el método tiene algún valor devuelto distinto, por ejemplo un *null* o un valor arbitrario para algún caso concreto, ha de indicarse.

Aunque no hay una norma, para las descripciones largas se suele usar la descripción del método y en el *@return* se suele abreviar.

4.2.5. Excepción @exception

Si el método es susceptible de devolver una excepción, ha de incluirse la etiqueta ***@exception excepción descripción*** estipulando la excepción lanzada y cuándo y porqué la lanza.

```java
/**
* Devuelve el elemento solicitado.
*
* @param index El indice a buscar.
* @return El elemento solicitado.
* @exception IndexOutOfBoundsException Si el indice está fuera del rango permitido.
*/
public String getElem(int index) throws IndexOutOfBoundsException
```

4.2.6. Referencia @see

La etiqueta ***@see paquete.clase#metodo texto*** permite referenciar a un método, clase o paquete, ya sea interno o externo.

El Javadoc resultante generará un enlace a lo referenciado.

No debe abusarse de su uso, solo cuando sea estrictamente necesario.

4.2.7. Desde @since

La etiqueta ***@since versión*** permite estipular desde qué versión existe lo que referencia.

Esto permite gestionar los cambios realizados en una clase.

4.2.8. Obsoleto @deprecated

La etiqueta ***@deprecated texto*** permite notificar que el método, clase o interfaz está obsoleto y no debe usarse.

El texto se suele utilizar para indicar qué usar en su lugar.

```java
/**
 * Añade un elemento
 * @param name El nombre del elemento a añadir.
 *
 * @deprecated Usar el método con dos parámetros.
 */
public void addElement(String name) {
```

Como puede observarse en el ejemplo, los IDEs marcan los métodos obsoletos tachándolos y advirtiendo al programador.

4.3. Convenciones y buenas prácticas

Existen una serie de buenas prácticas y normas de facto para escribir Javadoc, así como para describir en general.

- Usar la tercera persona del singular.
- Describir de forma clara y breve, pero sin dejarse nada.
- No describir el funcionamiento interno o la implementación, solo qué hace y cómo usarlo a no ser que sea relevante para su uso.
- Empezar las descripciones de los métodos con un tiempo verbal.
- Omitir los sujetos, sobre todo en las clases y atributos.
- Usar este en vez de el/la. Por ejemplo, este método.
- Añadir más que el nombre a las descripciones de los parámetros, aunque sea autoexplicativo.

- Usar las etiquetas HTML <code></code> para englobar aquellas referencias a variables, clases, paquetes o cualquier otro código.
- Evitar las abreviaturas.

4.4. Generación de documentación

La mayoría de los IDE disponen del generador integrado, por lo que la generación se realiza mediante los menús del propio programa.

Muchos analizan todo el Javadoc antes de generar la documentación, indicando si falta algo o hay algún tipo de error. Es habitual, por ejemplo, que obliguen a incluirlo en todos los métodos.

Para cada IDE es necesario seguir la guía o manual específico, pues el proceso difiere de uno a otro.

4.5. Generación de archivo

Hay IDEs que permiten general la documentación solo en formato HTML mediante un conjunto de archivos y carpetas, otros permiten generarla y comprimirla en *zip* o *rar* y otros, general un *jar* de Javadoc.

Si el IDE no lo permite desde el menú contextual, la generación del archivo *jar* es muy sencilla, solo es necesario comprimir la estructura de carpetas en *rar* y luego cambiar la extensión por *jar*.