



salesianos
CÁDIZ

Entornos de Desarrollo

UD6. Optimización y Documentación
(3. Documentación)

Índice

Introducción

Tipos de Comentarios

Cuando hay que poner un Comentario

Javadoc

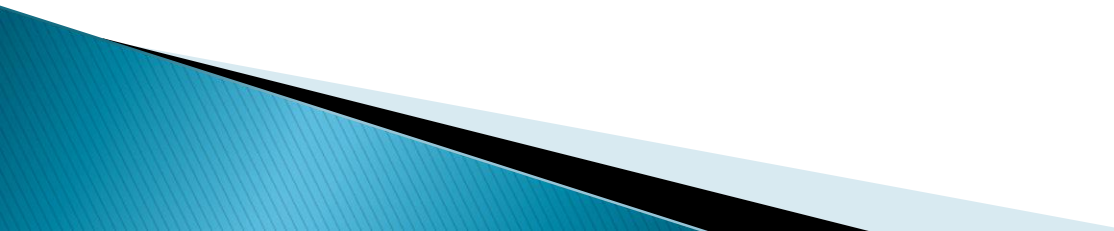
Introducción

- ▶ **Necesidad que se aprecia cuando hay errores que reparar o hay que extender el programa con nuevas capacidades o adaptarlo a un nuevo escenario.**
- ▶ **Reglas que no se deben olvidar nunca:**
 - **Todos los programas tienen errores** y descubrirlos sólo es cuestión de tiempo y de que el programa tenga éxito y se utilice frecuentemente
 - **Todos los programas sufren modificaciones** a lo largo de su vida, al menos todos aquellos que tienen éxito
- ▶ **Todo programa que tenga éxito será modificado en el futuro, bien por el programador original, bien por otro programador que le sustituya.**
- ▶ **¿Para qué sirve documentar el código?**
- ▶ **Para explicar su funcionamiento, de forma que cualquier persona que lea el comentario pueda entender la finalidad del código.**

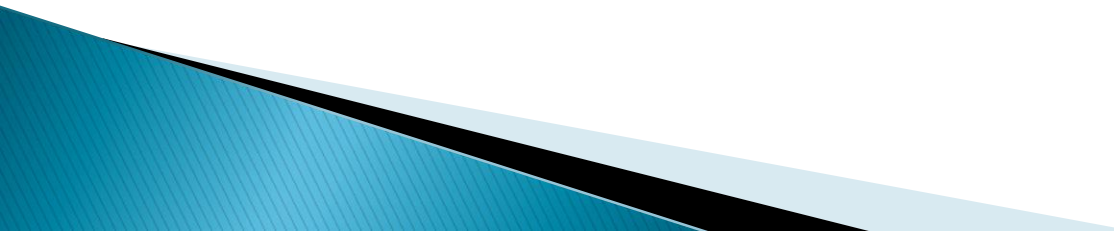
Introducción

- ▶ **¿A quiénes interesa el código fuente?**
 - Autores del propio código
 - Otros desarrolladores del proyecto
 - Clientes de la API del proyecto
- ▶ **¿Por qué documentar el código fuente?**
 - Mantenimiento
 - Reutilización
- ▶ **Documentar código** → añadir suficiente información para explicar lo que hace el código, de forma que las personas entiendan qué están haciendo y por qué.

Introducción

- ▶ Más vale que **sobren comentarios** a que **falten**. Pero con cautela, ya que cuando un programa se modifica, a la vez, deben modificarse los comentarios para que no queden obsoletos y acaben refiriéndose a un algoritmo que ya no existe.
- 

¿Qué hay que documentar?

- ▶ Añadir explicaciones a todo lo que no es evidente.
 - ▶ ¿de qué se encarga una clase? ¿un paquete?
 - ▶ ¿qué hace un método?
 - ▶ ¿cuál es el uso esperado de un método?
 - ▶ ¿para qué se usa una variable?
 - ▶ ¿cuál es el uso esperado de una variable?
 - ▶ ¿qué algoritmo estamos usando? ¿de dónde lo hemos sacado?
 - ▶ ¿qué limitaciones tiene el algoritmo?
 - ▶ ¿qué se debería mejorar ... si hubiera tiempo?
- 

Tipos de comentarios

- ▶ **javadoc**
- ▶ Se utiliza para generar documentación externa.

```
/**
 *
 * Descripción principal (texto / HTML)
 *
 *
 * Tags (texto / HTML)
 *
 */
```

- ▶ **Una línea**
- ▶ Caracteres "//" y terminan con la línea. Se utiliza para documentar código que no necesitamos que aparezca en la documentación externa (que genere javadoc).
- ▶ **Tipo C**
- ▶ Caracteres "/*", se pueden prolongar a lo largo de varias líneas (que probablemente comiencen con el carácter "/*") y terminan con los caracteres "*/". Mantenimiento de código obsoleto “por si acaso”.

Javadoc: documentación de APIs

- ▶ Generar documentación de una API a mano es tedioso y propenso a errores
- ▶ El paquete de desarrollo Java incluye una herramienta, **javadoc**, para generar un conjunto de páginas web a partir de los ficheros de código.
- ▶ Contrato entre el programador de la clase/método y el usuario, siendo la forma de referencia más rápida para ver que **funcionalidades ofrece una clase, paquete o interfaz**.
- ▶ Javadoc realiza algunos comentarios, de los que exige una sintaxis especial. Deben comenzar por `"/**"` y terminar por `"*/"`, incluyendo una descripción y algunas etiquetas especiales

Javadoc: documentación de APIs

- ▶ Es una utilidad de Oracle para la generación de documentación de APIs en formato HTML a partir de código fuente Java. Javadoc es el estándar de la industria para documentar clases de Java.
- ▶ Para generar APIs con Javadoc han de usarse etiquetas (tags) de HTML o ciertas palabras reservadas precedidas por el carácter "@". La lista completa de las tags con su correspondiente uso puede verse en oracle.com
- ▶ Estas etiquetas se escriben al principio de cada clase, miembro o método, dependiendo de qué objeto se desee describir, mediante un comentario iniciado con `"/**` y acabado con `*/`.

Javadoc: documentación de APIs

Java™ Platform
Standard Ed. 6

[All Classes](#)

Packages
[java.applet](#)
[java.awt](#)

All Classes
[AbstractAction](#)
[AbstractAnnotationValueVisitor6](#)
[AbstractBorder](#)
[AbstractButton](#)
[AbstractCellEditor](#)
[AbstractCollection](#)
[AbstractColorChooserPanel](#)
[AbstractDocument](#)
[AbstractDocument.AttributeCom](#)
[AbstractDocument.Content](#)
[AbstractDocument.ElementEdit](#)
[AbstractElementVisitor6](#)
[AbstractExecutorService](#)
[AbstractInterruptibleChannel](#)
[AbstractLayoutCache](#)
[AbstractLayoutCache.NodeDimen](#)
[AbstractList](#)
[AbstractListModel](#)
[AbstractMap](#)
[AbstractMap.SimpleEntry](#)

[Overview](#) [Package](#) [Class](#) [Use](#) [Tree](#) [Deprecated](#) [Index](#) [Help](#)

PREV NEXT

FRAMES NO FRAMES

Java™ Platform, Standard Edition 6
API Specification

This document is the API specification for version 6 of the Java™ Platform, Standard Edition.

See:
[Description](#)

Packages	
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.

Javadoc: documentación de APIs

```
/**  
 * Parte descriptiva.  
 * Que puede consistir de varias frases o párrafos.  
 *  
 * @etiqueta texto específico de la etiqueta  
 */
```

- ▶ Hay dos tipos de tags:
 - **Block tags:** Ubicados después de la descripción principal. @tag
 - **Inline tags:** Ubicados en la descripción principal o en las descripciones de los *block tags*. {@tag}

Javadoc: documentación de APIs

► Documentación de clases e interfaces

@author	nombre del autor
@version	identificación de la versión y fecha
@see	referencia a otras clases y métodos

► @see reference: Añade enlaces de referencia a otras partes de la documentación.

```
/**
 * @see "Patrones de Diseño: Un enfoque práctico
 * @see <a href="http://www.w3.org/WAI/"> Web Accesibility Initiative </a>
 * @see String#equals (Object) equals
 */

public void metodo ()
{

}
```

Javadoc: documentación de APIs

► Documentación de constructores y métodos

@param	nombre del parámetro	descripción de su significado y uso
@return		descripción de lo que se devuelve
@exception	nombre de la excepción	excepciones que pueden lanzarse
@throws	nombre de la excepción	excepciones que pueden lanzarse
@deprecated		Indica que el uso del elemento está desaconsejado

```
/**Elimina de esta lista todos los elementos cuyo índice está entre los dos
 * índices que se le pasan como parámetros
 * @param inicio índice del primer elemento a eliminar
 * @param fin índice del último elemento a eliminar
 */
public void EliminarElementos(int inicio,int fin)
{
}
}
```

Javadoc: documentación de APIs

► Documentación de constructores y métodos

```
/**
 * Tests if this vector has no components.
 *
 * @return <code>true</code> if and only if this vector has
 *         no components, that is, its size is zero;
 *         <code>false</code> otherwise.
 */
public boolean isEmpty() {
    return elementCount == 0;
}
```

```
/**
 * Parses the string argument as a signed decimal
 * <code>long</code>. The characters in the string must all be
 * decimal digits, except that...
 *
 * @param    s a <code>String</code> containing the <code>long</code>
 *             representation to be parsed
 * @return    the <code>long</code> represented by the argument in
 *             decimal.
 * @exception NumberFormatException if the string does not contain a
 *             parsable <code>long</code>.
 */
public static long parseLong(String s)
    throws NumberFormatException {
    ....
}
```

Javadoc: documentación de APIs

- ▶ Documentación de constructores y métodos
- ▶ {@link package.class#member label}

```
/**
 * Returns the component at the specified index.
 *
 * This method is identical in functionality to the {@link #get(int)}
 * method (which is part of the {@link List} interface).
 *
 * @param index an index into this vector
 * @return the component at the specified index
 * @throws ArrayIndexOutOfBoundsException if the index is out of range
 *         (<code>index < 0 || index >= size()</code>)
 */
public Object elementAt(int index) {
    ...
}
```


Javadoc: documentación de APIs

- ▶ **Documentación de constructores y métodos**
- ▶ `{@inheritDoc}`: Copiado de documentación
- ▶ **Implícito (automático)**. La herramienta de generación de la documentación toma la descripción o el *tag* de la clase o interfaz de nivel superior.
- ▶ **Explícito**: Copia la documentación del elemento de nivel superior inmediato.

```
/**
 * (superclase) ...
 *
 * @throws NullPointerException if <code>dst</code> is <code>>null</code>
 */
public void getChars(int srcBegin, int srcEnd, char dst[], int dstBegin) {
    ....
}
```

```
/**
 * (subclase) ...
 *
 * @throws NullPointerException {@inheritDoc}
 */
public void getChars(int srcBegin, int srcEnd, char dst[], int dstBegin) {
```