Javadoc. Generación automática de documentación

Cuando programamos una clase, debemos generar documentación lo suficientemente detallada sobre ella como para que otros programadores sean capaces de usarla. No debe existir necesidad de leer o estudiar su implementación, lo mismo que nosotros para usar una clase del API[i] no leemos ni estudiamos su código fuente.

Javadoc es una herramienta (perteneciente al JDK) para la generación de documentación en formato HTML (HyperText Markup Language, lenguaje de marcas de hipertexto) a partir de código fuente Java. Esta herramienta analiza los comentarios y extrae información sobre las clases y los métodos.

¿Qué se debe incluir al documentar una clase?

a) Nombre de la clase, descripción general, número de versión, nombre de autores.

b) Documentación de cada constructor o método (especialmente los públicos) incluyendo: nombre del constructor o método, tipo de dato regresado, nombres y tipos de parámetros si los hay, descripción general, descripción de parámetros (si los hay), descripción del valor que devuelve.

Para generar la documentación de un proyecto automáticamente hemos de seguir unas normas a la hora de realizar los comentarios dentro del mismo. Si las hemos seguido, en NetBeans usamos.

Run --> Generate Javadoc (NombreProyecto)

Nos abre la documentación del proyecto en un navegador web. El navegador que utilizará es el indicado en:

Tools → Options → General → Web Browser

Podemos modificarlo seleccionando en 'web browser' el que deseamos, Apply → OK

También en la carpeta dist → Javadoc encontramos los archivos creados.

Al generar el Javadoc en Netbeans muestra:

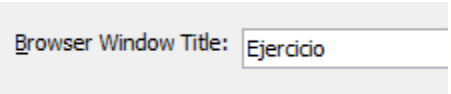
Warning: Leaving out empty argument '-windowtitle'

Para evitar que salga se hace clic con el botón derecho sobre el nombre del proyecto:

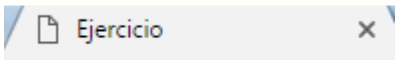
Properties → Build → Documenting → Browser Window Title

Se escribe el título que se desee.

Ejemplo:



En Chrome se mostrará como:



Para que Javadoc sea capaz de generar documentación automáticamente han de seguirse los siguientes pasos:

a) La documentación para Javadoc debe comenzar con /** y terminar con */

```
/**
 *Este es un comentario para javadoc
*/
```

b) La ubicación del comentario le define a Javadoc qué representa: si está justo antes de la declaración de la clase es un comentario de clase, y si está justo antes de un constructor o método se considerará un comentario de ese constructor o método. Los comentarios que se escriben en el cuerpo de un método no sirven para generar documentación. Un error común es intercalar una instrucción import entre el comentario y la declaración de la clase, en este caso Javadoc ignora el comentario y no genera la documentación.

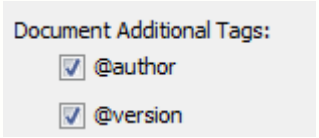
c) Javadoc funciona por medio de marcas (tags) o palabras reservadas que le ayudan al API a reconocer las partes que queremos documentar y crear los archivos HTML. Estas marcas van precedidas por el carácter "@", dentro de los símbolos de comentario. Si no existe al menos una línea que comience con @, entonces no se considerará el comentario para la documentación de Javadoc.

En la siguiente tabla se muestran algunas de las palabras reservadas (tags), aunque hay más

TAG	DESCRIPCIÓN
@author	Nombre del desarrollador.
@deprecated	Indica que el método o clase es obsoleto (propio de versiones anteriores) y que no se recomienda su uso.
@param	Definición de un parámetro de un método, es requerido para todos los parámetros del método.
@return	Informa de lo que devuelve el método, no se aplica en constructores o métodos "void".
@see	Indicar referencia cruzada. Otros métodos que se estén usando en el que se describe.
@version	Versión del método o clase.

Las etiquetas @author y @version se usan para documentar clases e interfaces. Por tanto no son válidas en la documentación de constructores ni métodos. No se muestran en la documentación generada por Javadoc, sólo se ven en el código del programa a menos que se seleccionen haciendo clic con el botón derecho sobre el nombre del proyecto y seleccionando:

Properties → Build → Documenting



La etiqueta @param se usa para documentar constructores y métodos. La etiqueta @return se usa sólo en métodos de tipo función.

Es importante que primero se escriba la descripción y al final del comentario se localicen los tags.

Los constructores deben llevar el calificativo public para que Javadoc los considere y muestre su documentación.

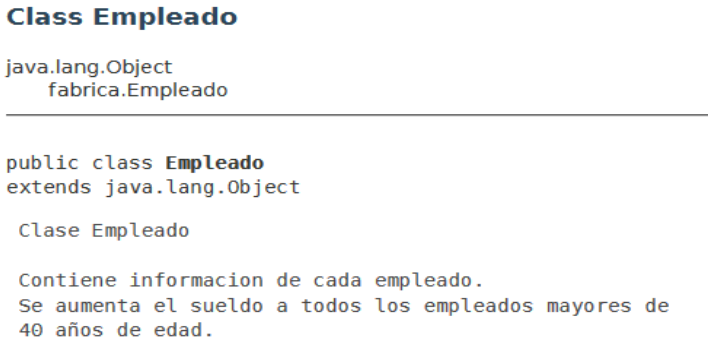
Se pueden utilizar elementos HTML en los comentarios de Javadoc. Nosotros sólo veremos el que nos permite escribir varias líneas en un comentario que describe la funcionalidad de una clase y los que utilizamos para crear listas.

Para escribir varias líneas utilizamos el elemento <pre>, para comenzar, y </pre> para terminar.

Ejemplo:

```
/**
 * <pre>
 * Clase Empleado
 *
 * Contiene información de cada empleado.
 * Se aumenta el sueldo a todos los empleados mayores de
 * 40 años de edad.
 * </pre>
*/
```

Se mostrará como:



Class Empleado

java.lang.Object
    fabrica.Empleado

---

public class **Empleado**
extends java.lang.Object

Clase Empleado

Contiene informacion de cada empleado.
Se aumenta el sueldo a todos los empleados mayores de
40 años de edad.

Si no utilizamos <pre> escribe el comentario en una sola línea, aunque utilicemos más de una al escribirlo.

Para crear listas tenemos dos opciones:

1. Listas numeradas. Se utiliza y y cada elemento de la lista estará encabezado por <li> que puede o no llevar la tag de cierre . Es conveniente que cada elemento de la lista esté en una línea. Cuando el navegador interpreta una lista ordenada, numera y sangra cada elemento en forma secuencial.

2. Listas con viñetas. Se utiliza <ul> y y cada elemento de la lista, también estará encabezado por la <li>. El resultado es que el navegador inserta viñetas delante de cada elemento.

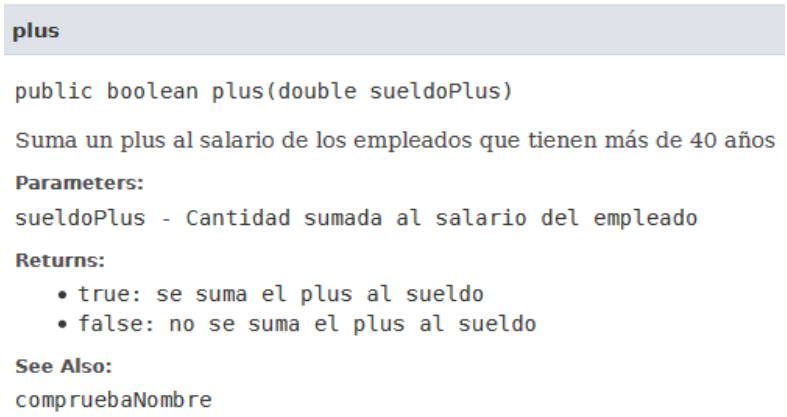
Las listas no se pueden especificar dentro de <pre> .. </pre>

Ejemplo:

```
/**
    * Suma un plus al salario de los empleados que tienen más de 40 años
    * @param sueldoPlus Cantidad sumada al salario del empleado
    * @see compruebaNombre
    * @return <ul>
    *         <li>true: se suma el plus al sueldo</li>
    *         <li>false: no se suma el plus al sueldo</li>
    *         </ul>
    */
```

Nota que primero se describe la acción del método y luego se encuentran los tags.

El resultado será:



plus

```
public boolean plus(double sueldoPlus)
```

Suma un plus al salario de los empleados que tienen más de 40 años

Parameters:

`sueldoPlus - Cantidad sumada al salario del empleado`

Returns:

- `true: se suma el plus al sueldo`
- `false: no se suma el plus al sueldo`

See Also:

`compruebaNombre`

[i] Al instalar el JDK se instalan ciertas clases que ofrecen la multinacional desarrolladora de Java (System y String, entre otras) y que están a disposición de todos los programadores listas para ser usadas. Estas clases junto a otros elementos forman lo que se denomina API (Application Programming Interface) de Java.