

## ÍNDICE

1. INTRODUCCIÓN .....	2
2. USO DE COMENTARIOS.....	2
3. ESTRUCTURA DE LAS HERRAMIENTAS DE CONTROL DE VERSIONES. ....	3

## 1. Introducción

El proceso de **documentación de código**, es uno de los aspectos más importantes de la labor de un programador. Documentar el código sirve para explicar su funcionamiento, punto por punto, de forma que cualquier persona que lea el comentario, pueda entender la finalidad del código.

La labor de documentación es fundamental para su mantenimiento posterior, que, en muchos casos, es realizado por personas diferentes a las que intervinieron en su creación.

La documentación añade explicaciones de la función del código, de las características de un método, etc. Debe tratar de describir todo lo que no resulta evidente. Su objetivo no es repetir lo que hace el código, sino indicar **por qué** se hace.

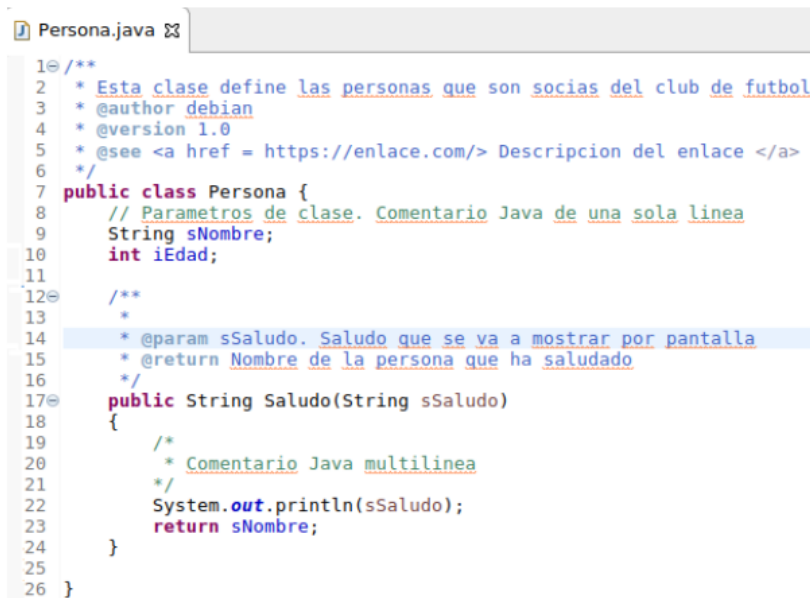
La documentación explicará cuál es la finalidad de una clase, de un paquete, qué hace un método, para qué sirve una variable, qué se espera del uso de una variable, qué algoritmo se usa, etc.

## 2. Uso de comentarios.

Uno de los elementos básicos para documentar código son los comentarios. Un comentario es una anotación que se realiza en el código, pero que el compilador va a ignorar. Sirve para indicar a los desarrolladores aspectos del código que pueden ser útiles.

En el caso del lenguaje Java, C# y C++, los comentarios se implementan de forma similar. Cuando se trata de un comentario de una sola línea, se usan los caracteres `//` seguidos del comentario. Para comentarios multilínea, los caracteres a utilizar son `/*` y `*/`, quedaría: `/* comentario-multilínea */`.

Otro tipo de comentarios disponibles en Java son los utilizados por la herramienta JavaDoc, que se describe más adelante, y que se escriben empezando por `/**` y terminando con `*/`. Estos comentarios pueden ocupar varias líneas y deben seguir un formato definido.



```
1 1= /**
2 2  * Esta clase define las personas que son socias del club de futbol
3 3  * @author debian
4 4  * @version 1.0
5 5  * @see <a href = https://enlace.com/> Descripcion del enlace </a>
6 6  */
7 7 public class Persona {
8 8     // Parametros de clase. Comentario Java de una sola linea
9 9     String sNombre;
10 10    int iEdad;
11 11
12 12    /**
13 13     *
14 14     * @param sSaludo. Saludo que se va a mostrar por pantalla
15 15     * @return Nombre de la persona que ha saludado
16 16     */
17 17    public String Saludo(String sSaludo)
18 18    {
19 19        /*
20 20         * Comentario Java multilinea
21 21         */
22 22        System.out.println(sSaludo);
23 23        return sNombre;
24 24    }
25 25
26 26 }
```

Existen diferentes herramientas que permiten automatizar, completar y enriquecer nuestra documentación, entre otras están **JavaDoc**, **SchemeSpy** o **Doxygen**.

### 3. Estructura de las herramientas de control de versiones.

Ahora que está clara la importancia de documentar el código, ¿cómo hacerlo? Los comentarios del código Java se incluyen incrustados en el código fuente, por tanto, serán parte de la información que aparece en los ficheros con extensión .java.

El compilador java (javac), ignora toda línea que aparece etiquetada como comentario, bien sea por // - comentario java de una línea, /\* --- \*/ - comentario java multilínea o /\*\* ----- \*/ comentario JavaDoc.

Por su parte el programa generador de documentación JavaDoc, ignorará toda información introducida en los ficheros \*.java excepto aquella que haya sido enmarcada entre los caracteres /\*\* y \*/.

Existe una serie de etiquetas que fijan como se presentará la información en la documentación resultante JavaDoc. En la url <https://en.wikipedia.org/wiki/Javadoc> aparece una colección de palabras reservadas (etiquetas) definidas en Javadoc. A continuación, se muestran algunas de las más utilizadas.

Etiqueta y parámetros	Uso	Asociada a
@author <i>nombre</i>	Identifica al autor.	Clase, interfaz
@version <i>versio_no</i>	Número de versión del software.	Clase, interfaz
@since <i>fecha-inicio</i>	Identifica la fecha de inicio de la función.	Clase, interfaz, campo, método.
@see <i>referencia</i>	Enlace a otro elemento de la documentación.	Clase, interfaz, campo, método.
@param <i>nombre descripción</i>	Describe un parámetro de un método.	Método.
@return <i>descripción</i>	Describe el valor devuelto de un método.	Método.
@exception <i>clase descripción</i> @throws <i>clase descripción</i>	Describe una excepción que puede ser lanzada por un método.	Método.
@deprecated <i>descripción</i>	Describe un método obsoleto.	Clase, interfaz, campo, método.

Los entornos de programación que implementan Java, como Eclipse o Netbeans, incluyen herramientas que generan páginas HTML de documentación a partir de los comentarios JavaDoc incluidos en el código fuente.

El siguiente enlace muestra la página de soporte Javadoc de Oracle:

<https://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html>