DOCUMENTACIÓN

3.1 INTRODUCCIÓN

- Por documentación se entiende el texto escrito que acompaña a los proyectos de cualquier tipo, en nuestro caso de software.
- La documentación es un requisito importante en un proyecto comercial, el cliente va a solicitar que se documenten las distintas fases del proyecto.
  - 1. Documentación de especificaciones. Este documento tiene como objeto asegurar que tanto el desarrollador como el cliente tienen la misma idea sobre las funcionalidades del sistema. Es muy importante ya que si no el desarrollo software no será aceptable. Este documento deberá tener: u
    - Introducción → Objetivos y fines del software
    - Descripción detalla del problema, incluyendo el software y hardware necesario
    - Descripción funcional con la descripción de cada función requerida en el sistema (incluyendo diagramas)
    - Y otro tipo de documentación descripción del comportamiento o criterios de validación.

- 2. Documentación del diseño. En la fase de diseño se decide la estructura de datos a utiliza, la forma en la que se van a implementar las distintas estructuras, el contenido de las clases, sus métodos y sus atributos, los objetos a utilizar, funciones, etc.
- 3. Documentación del código fuente. Durante la fase de implementación, cuando se está programando, es necesario comentar convenientemente cada una de las partes que tiene el programa. Estos comentarios se incluyen en el código fuente con el objeto de clarificar y explicar cada elemento del programa: Se deben comentar las clases, métodos, variables y en definitiva todo elemento que se considere importante.
- 4. Documentación de usuario final. Es la documentación que se entrega al usuario tanto especializado como no especializado y se describirá en la misma como utilizar las aplicaciones del proyecto.

- En este apartado nos vamos a centrar en la documentación del código fuente.
- En un proyecto de software, es muy importante la documentación, no solo para el usuario final, sino para los propios desarrolladores, tanto para uno mismo como para otros desarrolladores que tengan que trabajar en el presente o futuro en el proyecto.
- En un programa bien documentado es mucho más fácil reparar errores y añadirle nuevas funcionalidades para adaptarlo a nuevos escenarios que si carece de documentación.
- Hay dos reglas que no se deben olvidar nunca: Los programas tienen errores y descubrirlos solo es cuestión de tiempo y todos los programas sufren modificaciones a lo largo de su vida, al menos los que tienen éxito.

- Aunque es cierto que la refactorización ayuda en el entendimiento del código, siempre nos será más comprensible una buena documentación, no solo porque está escrito en nuestro propio lenguaje, sino porque puede contener notas aclaratorias que el código no nos podría decir directamente.
- A la hora de documentar interesa que se explica lo que hace una clase o un método y por qué y para qué se hace. Así se indicará entre otras cosas, de qué se encarga una clase, un paquete, un método o una variable.
- Para documentar proyectos existen muchas herramientas, normalmente cada lengujae dispone de su propia herramienta: PHPDoc, phpDocumentor, Javadoc o JSDoc (Javadoc para JavaScript). Nos centraremos principalmente en Javadoc.

3.2 USO DE JAVADOC

- El primer paso en una buena documentación es el uso apropiado de comentarios. El código necesita ser comentado porque no es suficientemente descriptivo por sí solo.
- Cada lenguaje tiene sus propios modos de establecer comentarios.
- Los comentarios pueden ser comentarios de líneas o comentarios de bloque. Estos últimos nos ahorran tener que comentar las líneas una a una cuando queremos comentar toda una porción de código.
- Además de los dos tipos de comentarios indicados, podemos añadir un tercer tipo de comentario: el comentario de documentación. Este tipo de comentario aporta información sobre las clases y métodos de nuestro código

- Javadoc es la utilidad de Java para extraer y generar documentación directamente del código en formato HTML.
- Para que la documentación sea en verdad útil debemos escribir los comentarios del código de acuerdo a las recomendaciones de Javadoc.
- La documentación y el código se van a incluir dentro del mismo fichero.
- Los tipos de comentario para generar la documentación son:
  - Comentarios de línea: Comienzan con los caracteres "//" y terminan con la línea
  - Comentarios tipo C: Comienzan con los caracteres "/*" y terminan con los caracteres "*/". Pueden agrupar varias línea

- Comentarios Javadoc:
  - Estos comentarios se colocar entre los delimitadores /** … */, agrupan varias líneas, y cada línea irá precedida por un * , y lo más importante es que estos deben colocarse antes de la declaración de una clase, un campo, un método o un constructor.
  - Dentro de estos delimitadores se podrán escribir etiquetas HTML.
  - Los comentarios Javadoc están formados por dos partes, una descripción seguida de un bloque de tags.
  - Vamos a crear un proyecto nuevo (JavadocPrueba) y dentro de ese proyecto crearemos una nueva clase que se va a llamar Prueba.

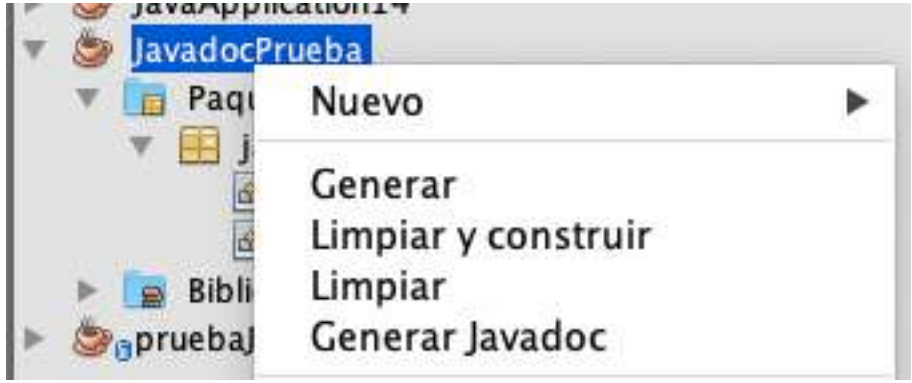
- Al crear NetBeans ya nos deja preparada la misma para introducir el Javadoc correspondiente.

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package javadocprueba;

/**
 *
 * @author guillermopalazoncano
 */
public class Prueba {

}
```

- Vamos a rellenar este javadoc con el siguiente código

```java
/**
 * <h2> Esto es un ejemplo de documentación </h2>
 * Puedo añadir etiquetas HTML, para <b>mejorar</b> la presentación.
 * Por ejemplo añado unas viñetas
 * <ul>
 *  <li> Inserción de registros </li>
 *  <li> Borrado y modificación de registros </li>
 * </ul>
 * @author guillermopalazoncano
 * @version v1.0
 */
public class Prueba {

}
```

- Para ver el javadoc que se generaría podemos pulsar con el botón derecho sobre el nombre del proyecto y se elegiría la opción “Generar Javadoc”



- Esto nos genera una estructura en forma de html en nuestra carpeta dist/javadoc (dentro de la carpeta del proyecto) con varios ficheros y entre ellos uno de ellos será el fichero index.html.
- Directamente Netbeans nos abre dicho fichero en un navegador con lo que podemos consultar la estructura creada en forma de html.

- Si pulsamos en nuestra clase creada (Prueba) podemos ver los comentarios que hemos insertado en la misma.

javadocprueba

Class Prueba

java.lang.Object
    javadocprueba.Prueba

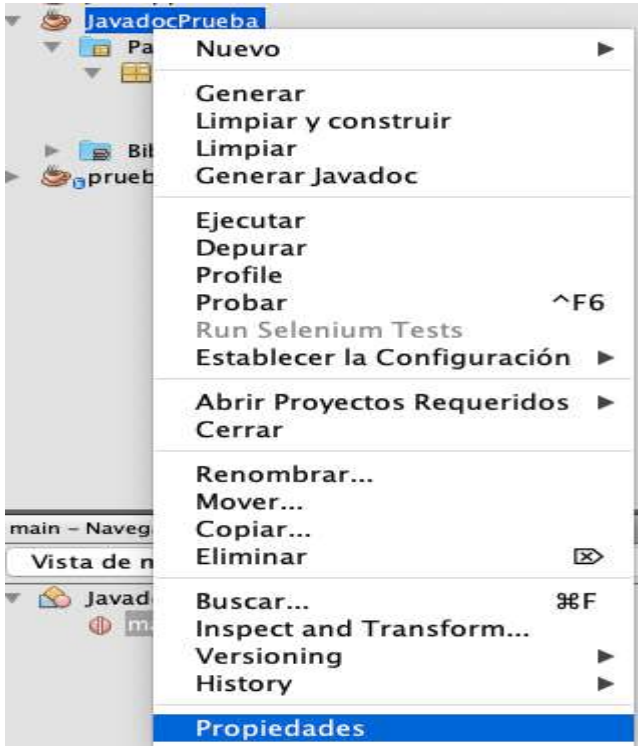
```
public class Prueba
extends java.lang.Object
```

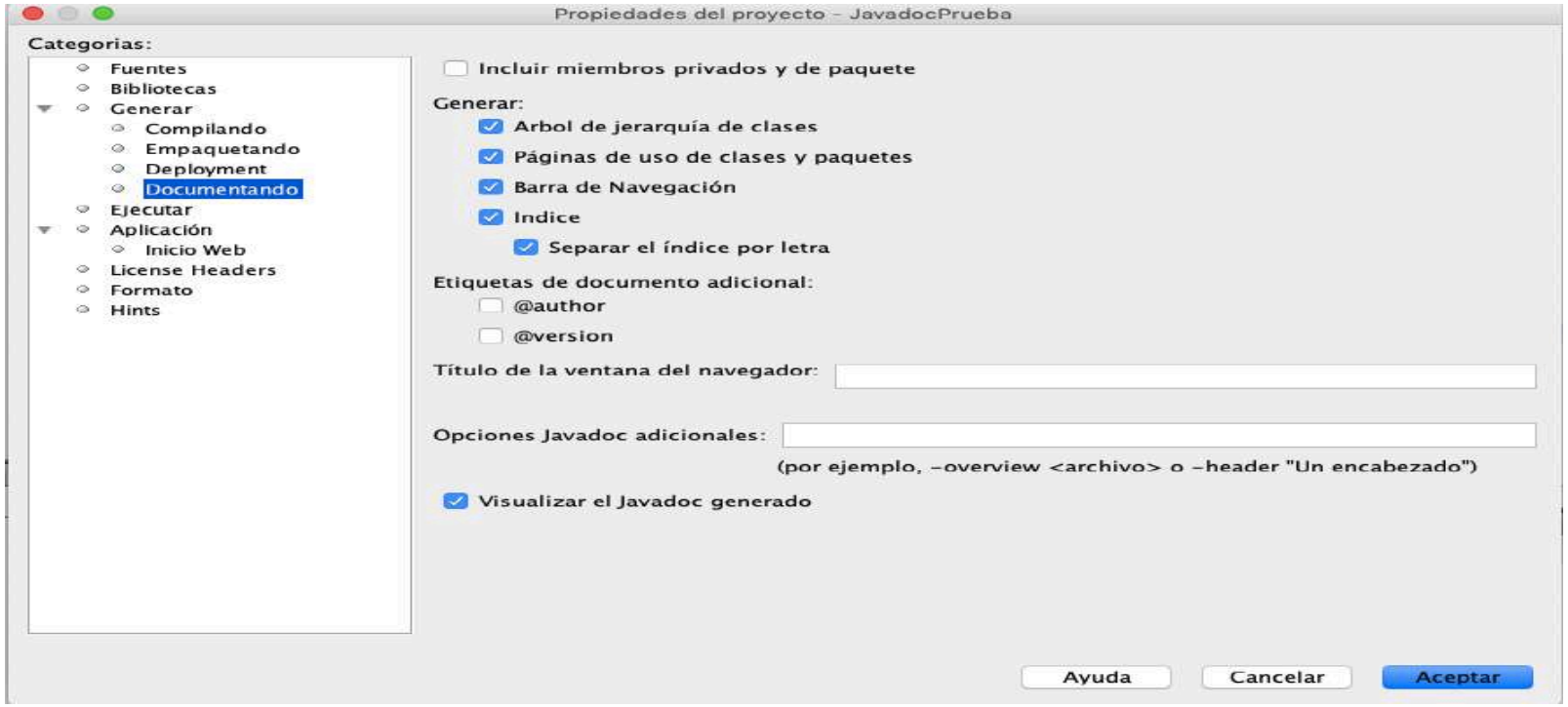
Esto es un ejemplo de documentación

Puedo añadir etiquetas HTML, para **mejorar** la presentación. Por ejemplo añado unas viñetas

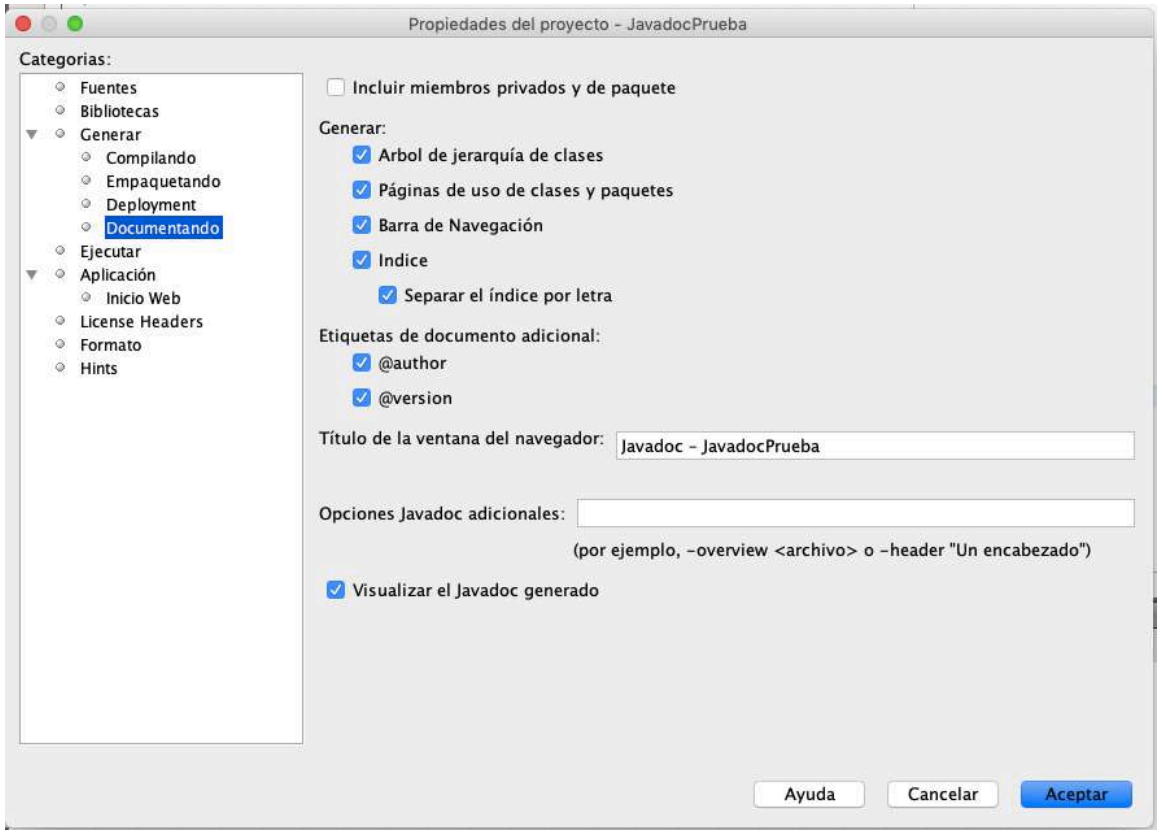
- Inserción de registros
- Borrado y modificación de registros

- Podemos revisar que habíamos añadido unos tags (etiquetas) @author y @version que sin embargo no aparecen en nuestro Javadoc creado.
- Esto es debido a que por defecto no se añaden al mismo. Pero nosotros podemos configurar en las propiedades de nuestro proyecto, en la categoría de Generar -> Documentando ciertas propiedades y entre ellas podemos indicar para que sí se incorporen la próxima vez que se genere.




Propiedades del proyecto - JavadocPrueba
Categorias:
Fuentes
Bibliotecas
Generar
Compilando
Empaquetando
Deployment
Documentando
Ejecutar
Aplicación
Inicio Web
License Headers
Formato
Hints
Incluir miembros privados y de paquete
Generar:
Arbol de jerarquía de clases
Páginas de uso de clases y paquetes
Barra de Navegación
Indice
Separar el índice por letra
Etiquetas de documento adicional:
@author
@version
Título de la ventana del navegador:
Opciones Javadoc adicionales:
(por ejemplo, -overview <archivo> o -header "Un encabezado")
Visualizar el Javadoc generado
Ayuda
Cancelar
Aceptar

- Además de marcar las opciones de @author y @version, vamos a indicar también un título para la ventana del navegador. Por ejemplo, podemos indicar "Javadoc – JavadocPrueba"



- Podemos comprobar que ha cambiado el título de la ventana y además ya aparece esta información en la página generada.

javadocprueba

Class Prueba

java.lang.Object
    javadocprueba.Prueba

```
public class Prueba
extends java.lang.Object
```

Esto es un ejemplo de documentación

Puedo añadir etiquetas HTML, para **mejorar** la presentación. Por ejemplo añado unas viñetas

- Inserción de registros
- Borrado y modificación de registros

Version:

v1.0

Author:

guillermopalazoncano

3.2.1 ETIQUETAS DE DOCUMENTACIÓN

- Se pueden usar varios tags o etiques para documentar ciertos aspectos concretos como la versión de la clase, el autor, los parámetros utilizados, o los valores devueltos.
- Las etiquetas de Javadoc van precedidas por @ siendo las más usadas las siguientes:
  - 1. @author. Autor de la clase. Sólo para las clases
  - 2. @versión. Versión de la clase. Sólo para las clases.
  - 3. @see. Referencia a otra clase, ya sea del API, del mismo proyecto o de otro. Algunas veces se utiliza para enlazar a páginas web.
    - Referencia (#método(); clase#método(); paquete.clase; paquete.clase#método()).

- 4. @param. Descripción de parámetro. Una etiqueta por cada parámetro.
- 5. @return. Informa de lo que devuelve el método, no se puede usar en constructores o métodos "void".
- 6. @throws. Descripción de la excepción que puede propagar. Habrá una etiqueta throws por cada tipo de excepción.
- 7. @deprecated. Marca el método como obsoleto. Solo se mantiene por compatibilidad.
- 8. @since. Indica el número de versión desde la que existe el método.

3.2.2 EJEMPLO JAVADOC EN NETBEANS

- Vamos a crear una nueva clase Empleado en la que vamos a poner en práctica casi la totalidad de las etiquetas explicadas.
- Como documentación de la clase indicaremos la siguiente:

```
/**
 * <h2> Clase Empleado, se utiliza para crear y leer empleados de una Base de Datos </h2>
 *
 * Busca información de Javadoc en <a href="https://www.google.es/">GOOGLE</a>
 * @see <a href="https://www.google.es/">Google</a>
 * @version 1.0
 * @author guillermopalazoncano
 * @since 1.0
 */
public class Empleado {
```

- La clase tendrá una serie de variables que también vamos a documentar

```java
public class Empleado {
    /**
     * Atributo nombre del empleado. Tipo cadena
     */
    private String nombre;
    /**
     * Atributo apellido del empleado. Tipo cadena
     */
    private String apellido;
    /**
     * Salario del empleado. Tipo double
     */
    private double salario;
```

- Podemos ir generando el Javadoc del proyecto, para ver cómo se va quedando esta nueva clase que hemos añadido (Empleado).

javadocprueba

Class Empleado

java.lang.Object
    javadocprueba.Empleado

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

Clase Empleado, se utiliza para crear y leer empleados de una Base de Datos

Busca información de Javadoc en GOOGLE

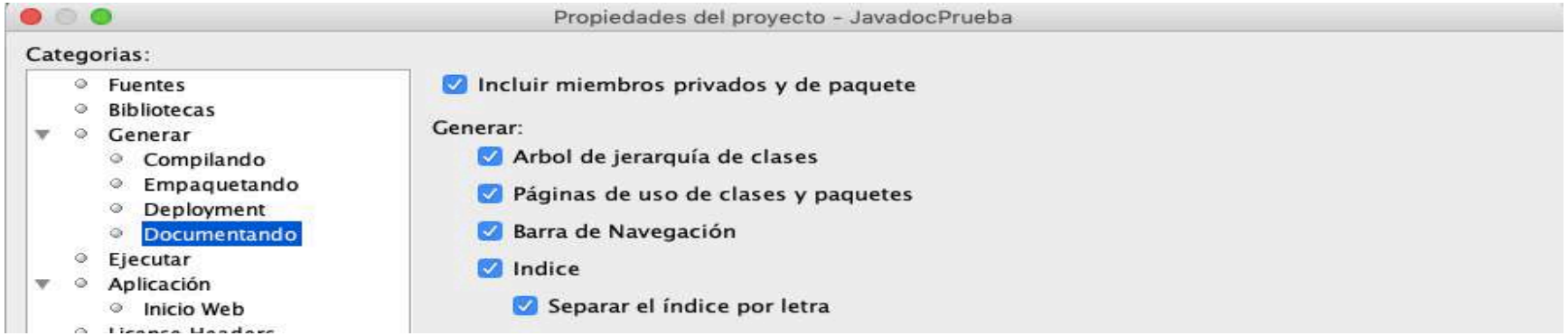
Since:
1.0

Version:
1.0

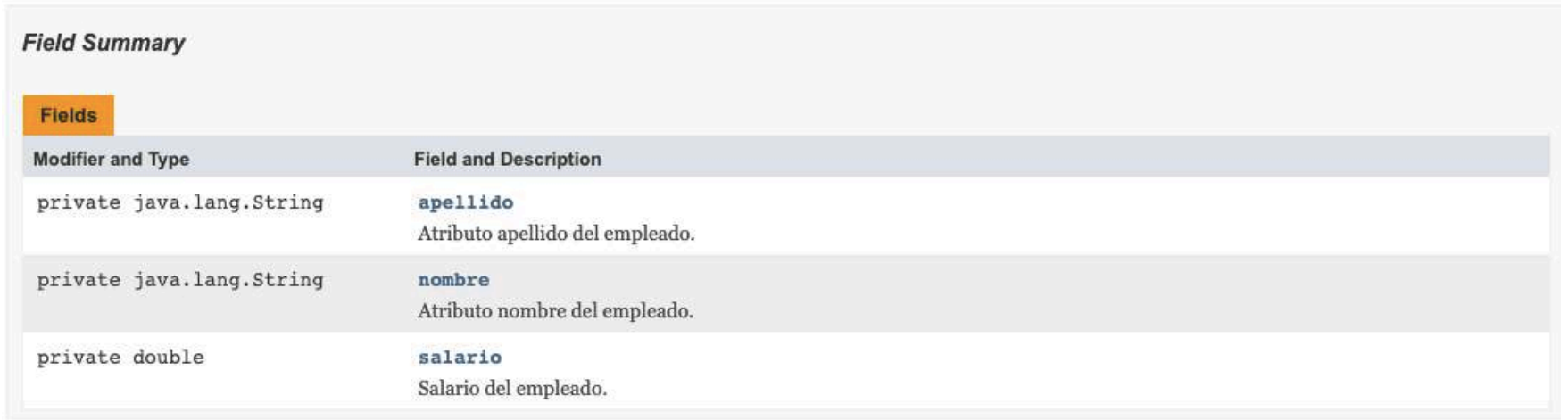
Author:
guillermopalazoncano

See Also:
Google

- Podemos comprobar que no se ha generado ninguna de la información indicada para los atributos.
- Esto es debido a que en las propiedad teníamos desmarcada la opción de Incluir miembros privados y de paquete. Marcamos dicha opción y volvemos a generar el Javadoc.



- Ahora ya sí se genera información para los atributos de la clase. Además tenemos dos opciones: Field Summary y Field Details donde se verá más información.



Field Summary

Fields

Modifier and Type	Field and Description
`private java.lang.String`	**`apellido`** Atributo apellido del empleado.
`private java.lang.String`	**`nombre`** Atributo nombre del empleado.
`private double`	**`salario`** Salario del empleado.

Field Detail

nombre

```
private java.lang.String nombre
```

Atributo nombre del empleado. Tipo cadena

apellido

```
private java.lang.String apellido
```

Atributo apellido del empleado. Tipo cadena

salario

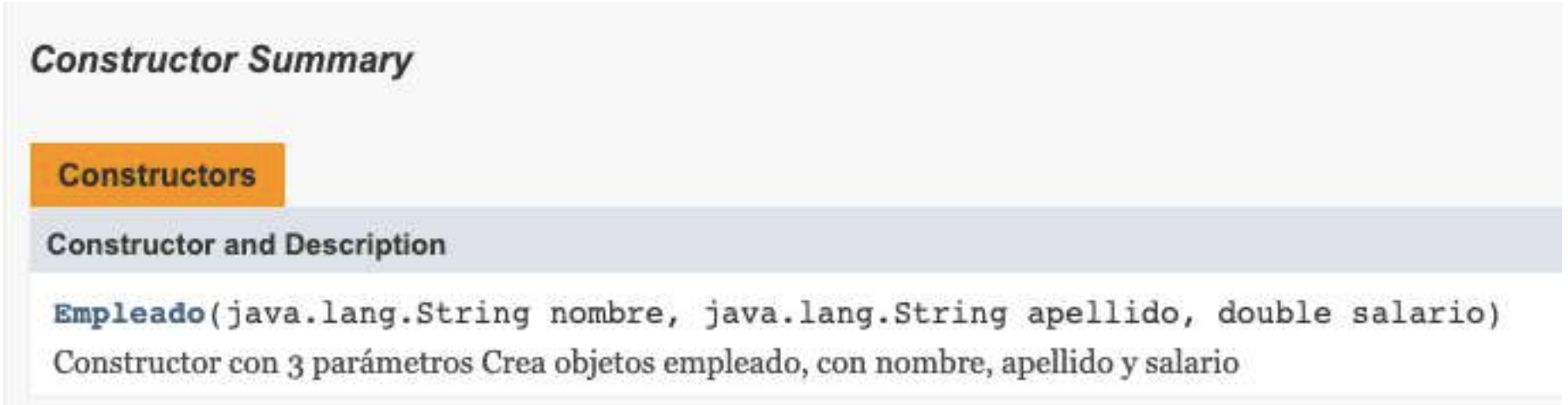
```
private double salario
```

Salario del empleado. Tipo double

- Lo siguiente que vamos a hacer es incorporar los comentarios de documentación para el constructor de la clase. Posteriormente generaremos el Javadoc

```
/**
 * Constructor con 3 parámetros
 * Crea objetos empleado, con nombre, apellido y salario
 * @param nombre Nombre del empleado
 * @param apellido Apellido del empleado
 * @param salario Salario del empleado
 */
public Empleado (String nombre, String apellido, double salario){
    this.nombre = nombre;
    this.apellido = apellido;
    this.salario = salario;
}
```

- Al igual que con Field, tenemos dos tablas de información sobre el constructor, Constructor Summary y Constructor Detail (donde ya aparece la información metida en las etiquetas).



Constructor Summary

Constructors

Constructor and Description
Empleado(java.lang.String nombre, java.lang.String apellido, double salario) Constructor con 3 parámetros Crea objetos empleado, con nombre, apellido y salario

Constructor Detail

Empleado

```
public Empleado(java.lang.String nombre,
                java.lang.String apellido,
                double salario)
```

Constructor con 3 parámetros Crea objetos empleado, con nombre, apellido y salario

Parameters:

`nombre` - Nombre del empleado

`apellido` - Apellido del empleado

`salario` - Salario del empleado

- Añadimos un método público y generamos el JavaDoc para revisar este método creado.

```java
/**
 * Este método sube el salario al empleado
 * @see Empleado
 * @param subida Double que se sumará al salario que ya tiene
 */
public void subidasalario(double subida){
    salario = salario + subida;
}
```

- Tenemos también un Method Summary y finalmente un Method Detail (Desde el Method Summary si pulsamos sobre el nombre del Método nos lleva a su opción de Method Detail respectivamente).

Method Summary

All Methods	Instance Methods	Concrete Methods

Modifier and Type	Method and Description
`void`	**`subidasalario`**`(double subida)` Este método sube el salario al empleado

Method Detail

subidasalario

```
public void subidasalario(double subida)
```

Este método sube el salario al empleado

Parameters:

`subida - Double que se sumará al salario que ya tiene`

See Also:

`Empleado`

- A continuación añado ahora un método privado. Y generamos el Javadoc correspondiente.

```
/**
 * Comprueba que el nombre no este vacío
 * @return <ul>
 *           <li> true: el nombre es una cadena vacía </li>
 *           <li> false: el nombre no es una cadena vacía </li>
 *         </ul>
 */
private boolean comprobar(){
    if (nombre.equals("")){
        return false;
    } else {
        return true;
    }
}
```

Method Summary

All Methods	Instance Methods	Concrete Methods

Modifier and Type	Method and Description
`private boolean`	**`comprobar()`** Comprueba que el nombre no este vacío
`void`	**`subidasalario`**`(double subida)` Este método sube el salario al empleado

Method Detail

subidasalario

```
public void subidasalario(double subida)
```

Este método sube el salario al empleado

Parameters:

`subida` – Double que se sumará al salario que ya tiene

See Also:

`Empleado`

comprobar

```
private boolean comprobar()
```

Comprueba que el nombre no este vacío

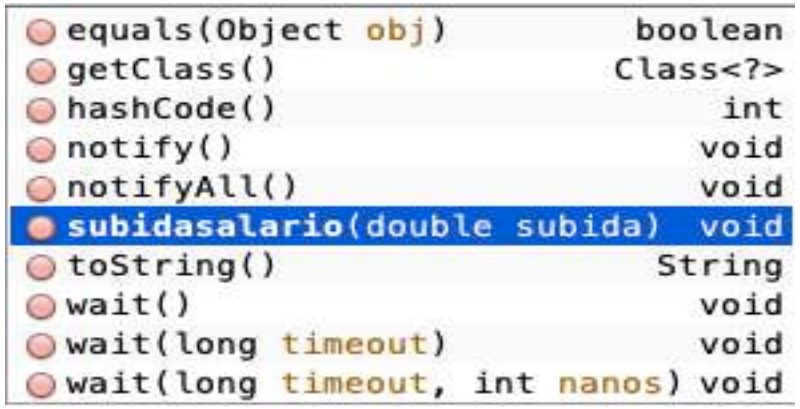
Returns:

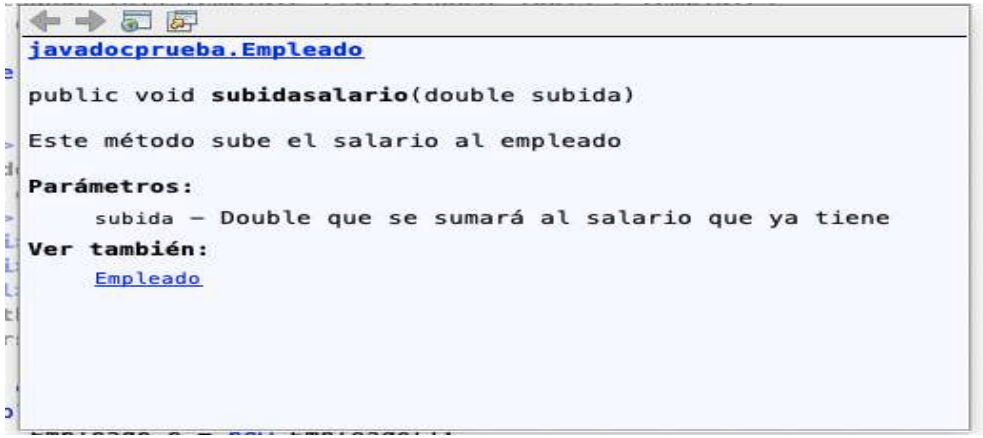
- `true`: el nombre es una cadena vacía
- `false`: el nombre no es una cadena vacía

- Una de las cosas que también tiene de positivo añadir nuestros comentarios en la creación de métodos, es que estos se podrán consultar desde otra clase que utilice dicha clase (y dichos métodos).
- Por ejemplo, desde nuestra clase Prueba, creo un nuevo método que se llama creaEmpleado

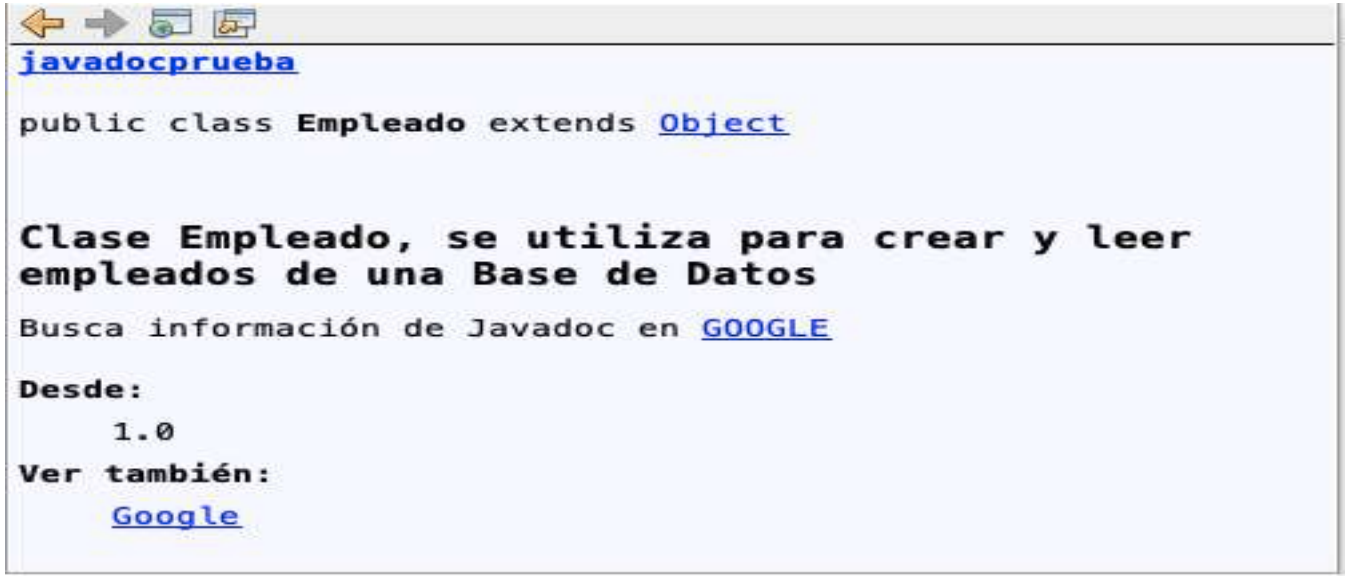
```
public void creaEmpleado(){
    Empleado e = new Empleado("Guillermo","Palazon",10000);
    e.
}
```

- Al poner el “e.” Netbeans nos va a sugerir una serie de métodos y entre ellos los métodos públicos que tiene dicha clase.
- Si me sitúo encima del método subidasalario nos mostrará toda la información indicada en nuestro Javadoc

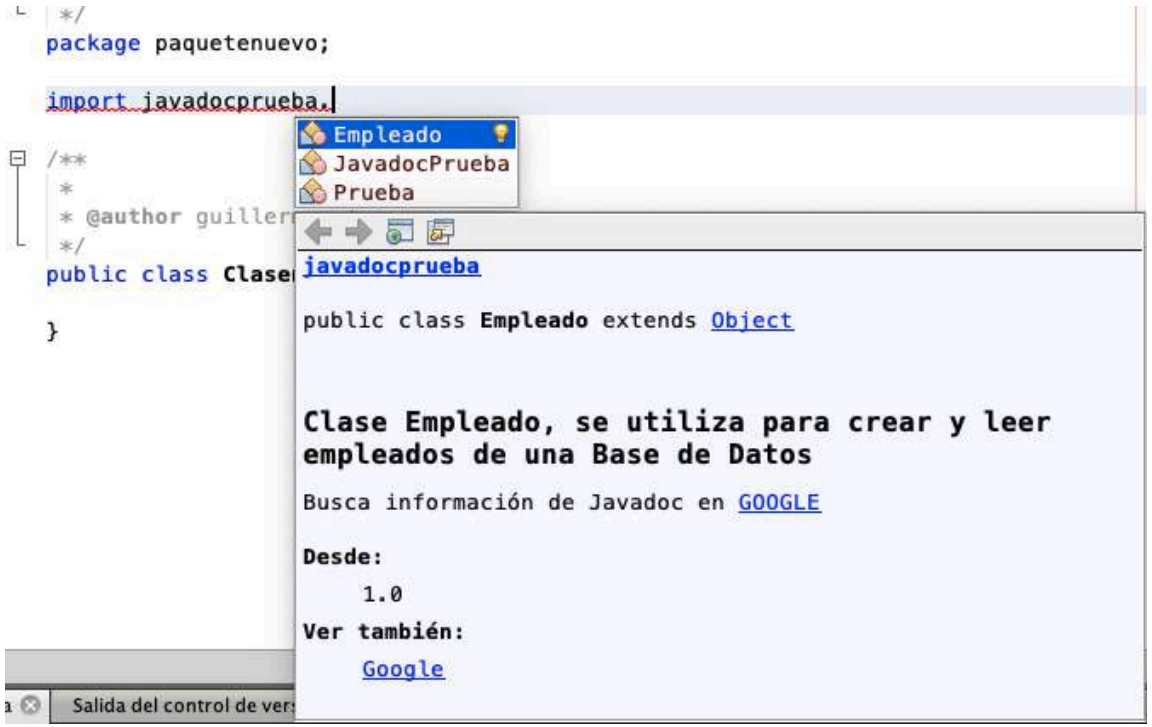




- Si pulsamos sobre la clase (donde pone javadocprueba.Empleado) nos llevaría también a la información que hemos insertado sobre la clase en sí.



- Esta información también sale disponible a la hora de importar una clase (de esta manera, puedo saber viendo su descripción, autor o cualquier información) si es la clase que quiero o no importar.
- Creo para ello un nuevo paquete (paquetenuevo) y dentro del mismo una nueva clase (Clasenueva).
- Si ahora dentro de esa clase pongo un import .javadocprueba, una de las clases que me va a ofrecer es Empleado y puedo consultar la información que he puesto antes mediante captura de pantalla.


*/
package paquetenuevo;
import javadocprueba.
Empleado
JavadocPrueba
Prueba
/**
*
* @author guiller
*/
public class Clase
}
javadocprueba
public class Empleado extends Object
Clase Empleado, se utiliza para crear y leer empleados de una Base de Datos
Busca información de Javadoc en GOOGLE
Desde:
1.0
Ver también:
Google
Salida del control de ver

- **RECUERDA:**
- Los comentarios de documentación Javadoc deben colocarse **ANTES** de las declaraciones de las clases, de los métodos, o de los atributos.