Documentación en Java, Javadoc.

En este documento se expondrá la documentación en Java, empleando para ello la utilidad Javadoc.

https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html

¿Qué es la documentación Javadoc?

Lo primero que debemos entender es el concepto de documentar.

Documentar el código de un programa es añadir suficiente información como para explicar adecuadamente lo que hace, de manera que las personas que accedan a este programa sean capaces de comprender el código.

Cuando un código no está documentado, o n lo está adecuadamente, entenderlo puede ser un proceso bastante largo y complicado, e implica una gran pérdida de tiempo. En un entorno de desarrollo real, donde los trabajadores que mantienen el código no tienen porque ser los mismos que lo han desarrollado (lo mismo para aquellas personas que realizan evolutivos), es extremadamente importante mantener el código comentado adecuadamente.

La documentación del código se hace a través de la inserción de comentarios dentro del propio fichero de código fuente (facilita mantener sincronizado el código y su documentación) y siempre debemos llevar a cabo el proceso de documentación al mismo tiempo que escribimos el código, hacerlo de otra forma es prácticamente garantía de que el código quedará o bien directamente sin comentar o no debidamente comentado.

Para facilitarnos esta tarea, Oracle incluye la herramienta Javadoc, que trabaja sobre los llamados comentarios Javadoc, que deben de tener una sintaxis concreta y a partir de estos genera un conjunto de páginas web.

Javadoc 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. La mayoría de los IDEs los generan automáticamente.

¿Cómo debemos comentar nuestros programas?

Debemos añadir explicaciones a todo aquello que no sea evidente. Asegurándonos que explicamos:

- De que se encarga cada clase Java.
- El uso y función de cada variable.
- La función y uso esperado de cada método.
- Las posibles mejoras a introducir en el programa en un futuro.
- Etc.

¿Qué formato deben de tener nuestros comentarios Javadoc?

- Deben aparecer inmediatamente ante de los elementos a comentar. Es decir, si estamos comentando una variable, el comentario correspondiente estará en la línea anterior, si estamos comentando un método o una clase pues más de los mismo.
- Deben empezar con los caracteres /** y terminar con los caracteres */
- La mayoría de entornos de desarrollo suelen resaltar este tipo de comentarios con el color azul.
- La primera frase de la documentación debe de ser un resumen que contenga una descripción completa y concisa de la entidad comentada. Si añadimos más frases, serán comentarios entrando más al detalle.
- La ubicación le define a javadoc qué representa el comentario: si está incluido justo antes de la declaración de clase se considerará un comentario de clase, y si está incluido justo antes de la signatura de un constructor o método se considerará un comentario de ese constructor o método.
- Si estamos añadiendo un comentario para documentar una clase, debemos incluir: **Nombre de la clase, descripción general, número de versión, nombre de autores.**
- Si estamos añadiendo un comentario para documentar un constructor u otro método: nombre del constructor o método, tipo de retorno, nombres y tipos de parámetros si los hay, descripción general, descripción de parámetros (si los hay), descripción del valor que devuelve.

Además, **todo proyecto debería tener un archivo Leeme o Readme.** En el readme.txt sería adecuado incluir al menos: título del proyecto, descripción, versión, cómo arrancar el proyecto, autores e instrucciones para los usuarios.

Para alimentar javadoc se usan ciertas palabras reservadas (tags o etiquetas) precedidas por el carácter "@", dentro de los símbolos de comentario javadoc. Si no existe al menos una línea que comience con @ no se reconocerá el comentario para la documentación de la clase.

Existen más etiquetas, puedes buscarlas por tu cuenta en la documentación de Oracle, aunque las más empleadas son las siguientes:

Tag	Descripción	Uso	Versión
@author	Nombre del desarrollador.	nombre_autor	1.0
@version	Versión del método o clase.	versión	1.0
@param	Definición de un parámetro de un método, es requerido para todos los parámetros del método.	nombre_parametro descripción	1.0
@return	Informa de lo que devuelve el método, no se puede usar en constructores o métodos "void".	descripción	1.0
@throws	Excepción lanzada por el método, posee un sinónimo de nombre @exception	nombre_clase descripción	1.2
@see	Asocia con otro método o clase.	referencia (#método(); clase#método(); paquete.clase; paquete.clase#método()).	1.0
@since	Especifica la versión del producto	indicativo numerico	1.2

@serial	Describe el significado del campo y sus valores aceptables. Otras formas válidas son @serialField y @serialData	campo_descripcion	1.2
@deprecated	Indica que el método o clase es antigua y que no se recomienda su uso porque posiblemente desaparecerá en versiones posteriores.	descripción	1.0

Las etiquetas @author y @version se usan para documentar clases e interfaces. Por tanto no son válidas en cabecera de constructores ni métodos. La etiqueta @param se usa para documentar constructores y métodos. La etiqueta @return se usa solo en métodos de tipo función. Dentro de los comentarios se admiten etiquetas HTML, por ejemplo con @see se puede referenciar una página web como link para recomendar su visita de cara a ampliar información.

A continuación, y como ejemplo para que empieces a comentar tus programas, se muestra una clase Empleado completamente comentada:

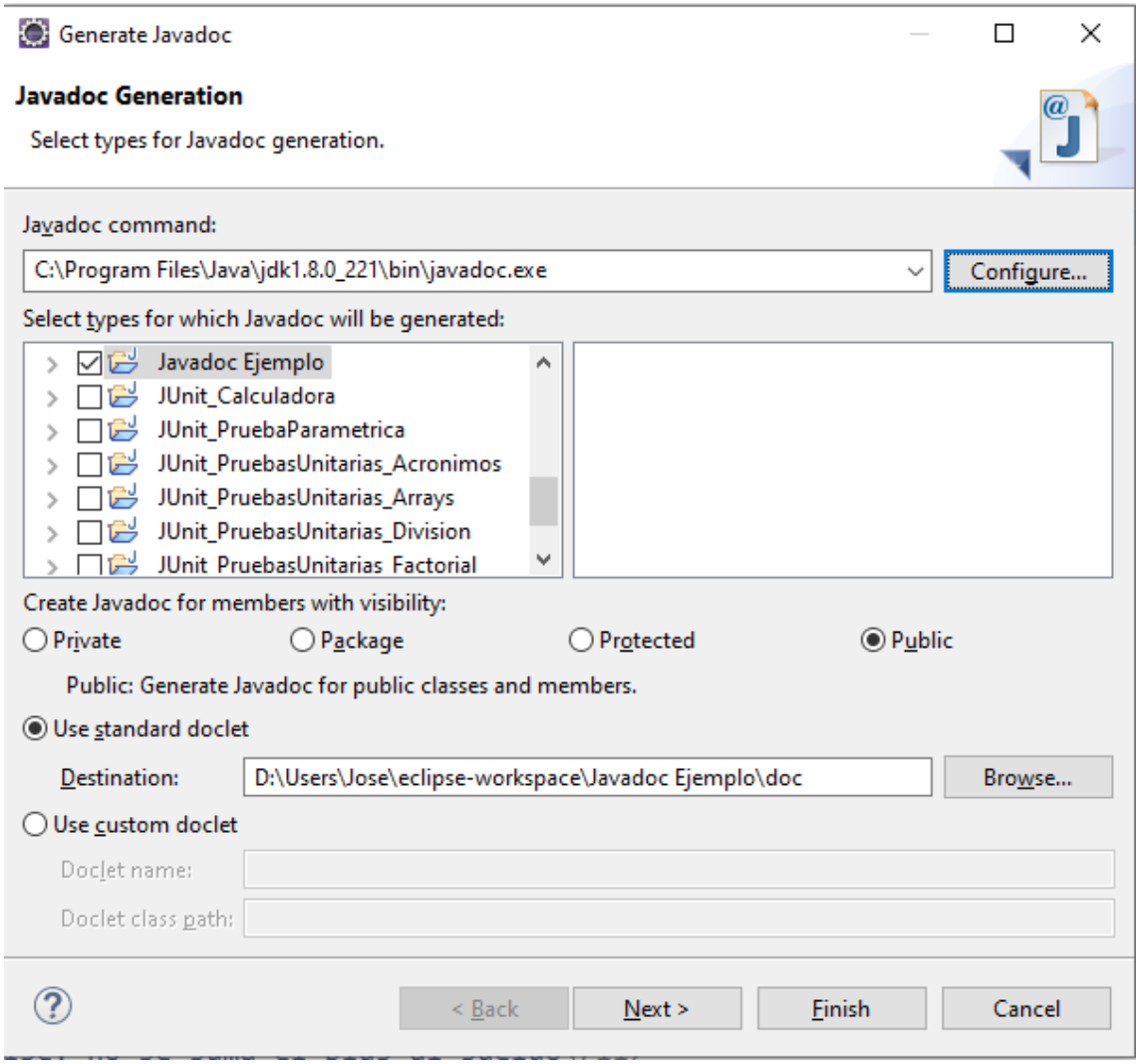
```
/**
 * Clase Empleado
 *
 * Contiene informacion de cada empleado
 *
 * @author Jose Rey Cid
 * @version 1.0
 */
public class Empleado {

    //Atributos

    /**
     * Nombre del empleado
     */
    private String nombre;
    /**
     * Apellido del empleado
     */
    private String apellido;
    /**
     * Edad del empleado
     */
```

```
    private int edad;
    /**
     * Salario del empleado
     */
    private double salario;

    //Metodos publicos

    /**
     * Suma un plus al salario del empleado si el empleado tiene mas
de 40 años
     * @param sueldoPlus
     * @return <ul>
     *          <li>true: se suma el plus al sueldo</li>
     *          <li>false: no se suma el plus al sueldo</li>
     *          </ul>
     */
    public boolean plus (double sueldoPlus){
        boolean aumento=false;
        if (edad>40 && compruebaNombre()){
            salario+=sueldoPlus;
            aumento=true;
        }
        return aumento;
    }

    //Metodos privados
    /**
     * Comprueba que el nombre no este vacio
     * @return <ul>
     *  <li>true: el nombre es una cadena vacia</li>
     *  <li>false: el nombre no es una cadena vacia</li>
     *  </ul>
     */
    private boolean compruebaNombre(){
        if(nombre.equals("")){
            return false;
        }
        return true;
    }

    //Constructores
    /**
     * Constructor por defecto
     */
    public Empleado(){
        this ("", "", 0, 0);
    }

    /**
     * Constructor con 4 parametros
     * @param nombre nombre del empleado
     * @param apellido nombre del empleado
     * @param edad edad del empleado
     * @param salario salario del empleado
     */
    public Empleado(String nombre, String apellido, int edad, double
salario){
        this.nombre=nombre;
```

```
        this.apellido=apellido;
        this.edad=edad;
        this.salario=salario;
    }
}
```

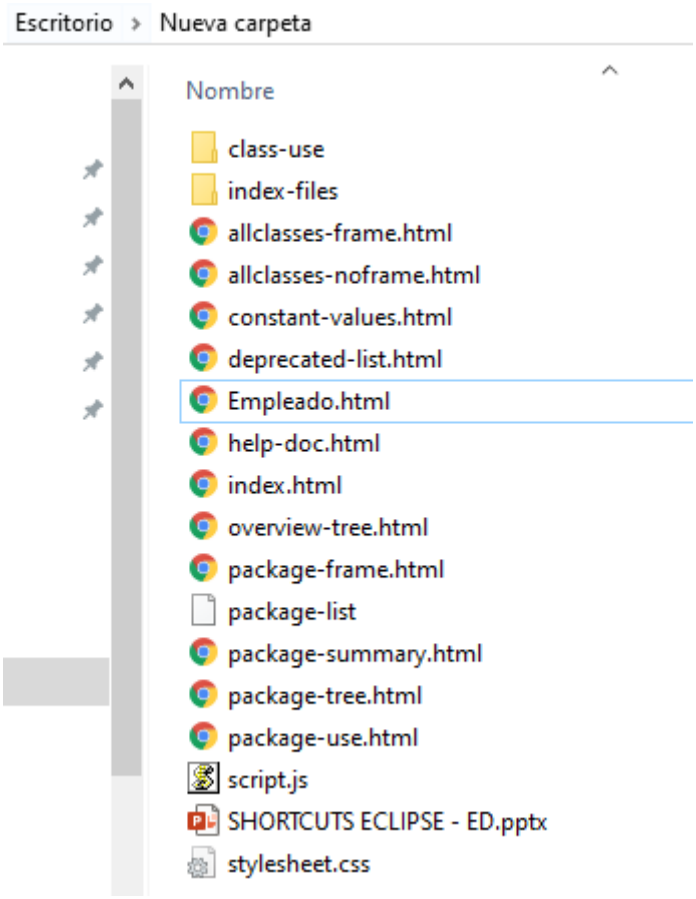
Ejercicio: Documenta y entrega en el Aula Virtual de la asignatura, el proyecto de sistema de gestión universitario que realizaste en la Unidad 3.

Una vez tengamos el código debidamente comentado, solo faltaría generar la documentación Javadoc. Para eso, pinchamos sobre el proyecto que queremos comentar y usamos la opción Proyect -> Generate Javadoc, que nos abrirá una ventana como la siguiente:

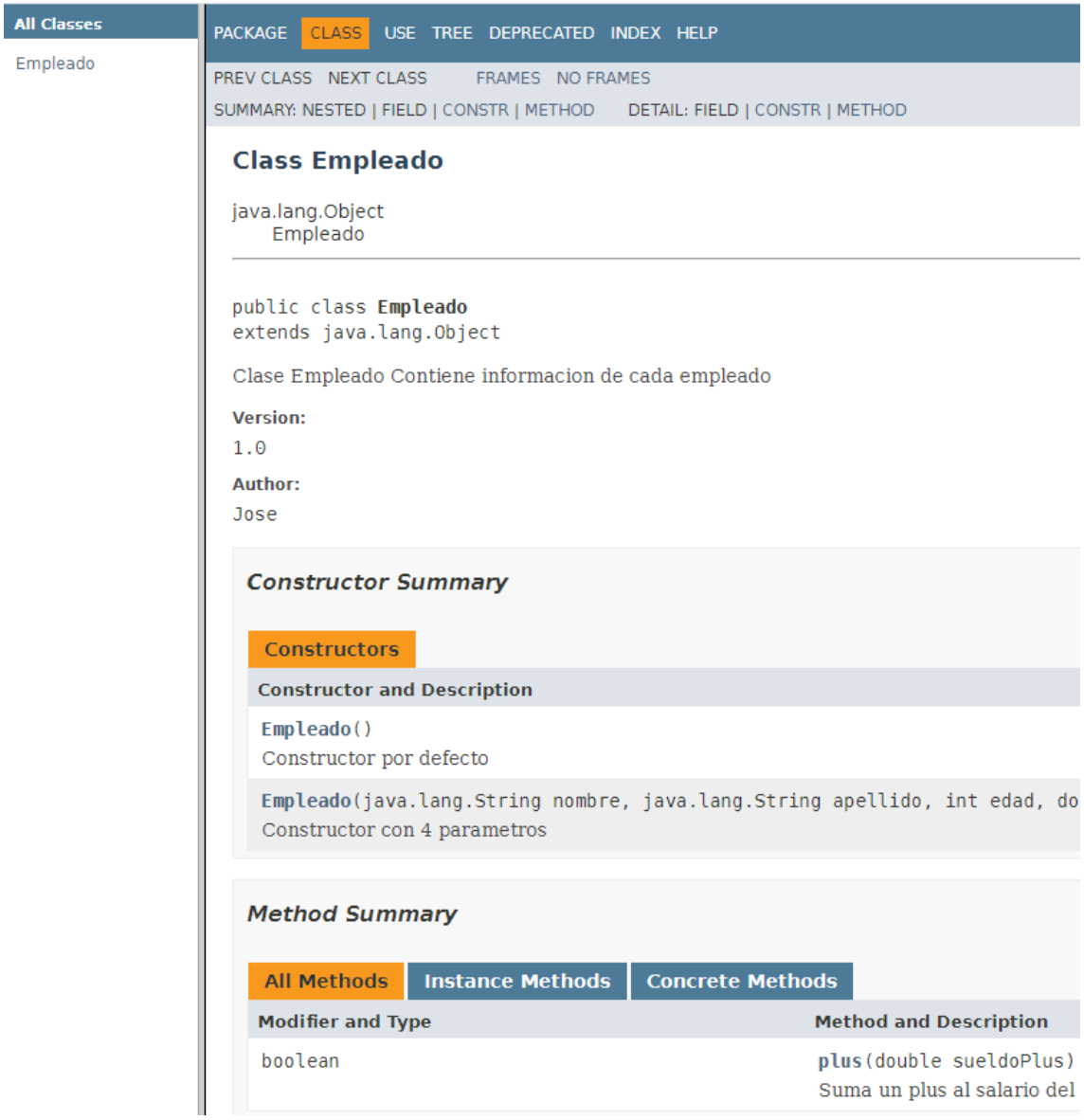


En ella indicamos la ubicación de nuestro javadoc.exe, que se encontrará en la capeta bin de nuestro JDK y elegiremos el destino donde queremos que se genere la página web.

Debido a la multitud de archivos que se generan, es conveniente crear una carpeta propia para la documentación:



Ya solo nos falta ver el resultado final abriendo el index.html:



All Classes

Empleado

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Empleado

java.lang.Object
Empleado

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

Clase Empleado Contiene informacion de cada empleado

Version:

1.0

Author:

Jose

Constructor Summary

Constructors

Constructor and Description
Empleado() Constructor por defecto
Empleado(java.lang.String nombre, java.lang.String apellido, int edad, do Constructor con 4 parametros

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
boolean	**plus**(double sueldoPlus) Suma un plus al salario del