

JavaDoc

Documentación

```
/**  
 * Easy  
 * Javadoc  
 */
```

INDEX

1. Proyecto de Java	3
a. Código Java	3
b. Comentarios al código	3
2. Creación del JavaDoc	4
3. Explicación del JavaDoc	6

1. Proyecto de Java

a. Código Java

Para este proyecto de Java he creado 3 archivos .java para poder generar un el JavaDoc y realizar la práctica:

- Clase Persona
- Clase Estudiante
- Clase Main

Link al código:

<https://github.com/xzhou12/m08-despl-app-web/tree/UF1/UF1/Practica%203%20JavaDoc/eli pse-workpage/src>

b. Comentarios al código

Para generar el JavaDoc con los diferentes parámetros/etiquetas tendremos que añadir comentarios en la parte superior de cada función y cada clase:

Las diferentes etiquetas que podremos usar son las siguientes:

- **@author** : Nombre del desarrollador.

```
/**
 * Esta clase representa a un estudiante, que es una persona y tiene una universidad asociada.
 * @author Xiaobin Zhou
 */
public class Estudiante extends Persona {
```

- **@version** : Versión del método o clase.
- **@param** : Definición de un parámetro de un método, es requerido para todos los parámetros del método.

```
/**
 * Constructor de la clase Estudiante.
 *
 * @param nombre El nombre del estudiante.
 * @param edad La edad del estudiante.
 * @param universidad La universidad a la que asiste el estudiante.
 */
public Estudiante(String nombre, int edad, String universidad) {
    super(nombre, edad);
    this.universidad = universidad;
}
```

- **@return** : Informa de lo que devuelve el método, no se puede usar en constructores o métodos "void".

```

/**
 * Obtiene el nombre de la persona.
 *
 * @return El nombre de la persona.
 */
public String getNombre() {
    return nombre;
}

```

- **@throws** : Excepción lanzada por el método, posee un sinónimo de nombre @exception.

```

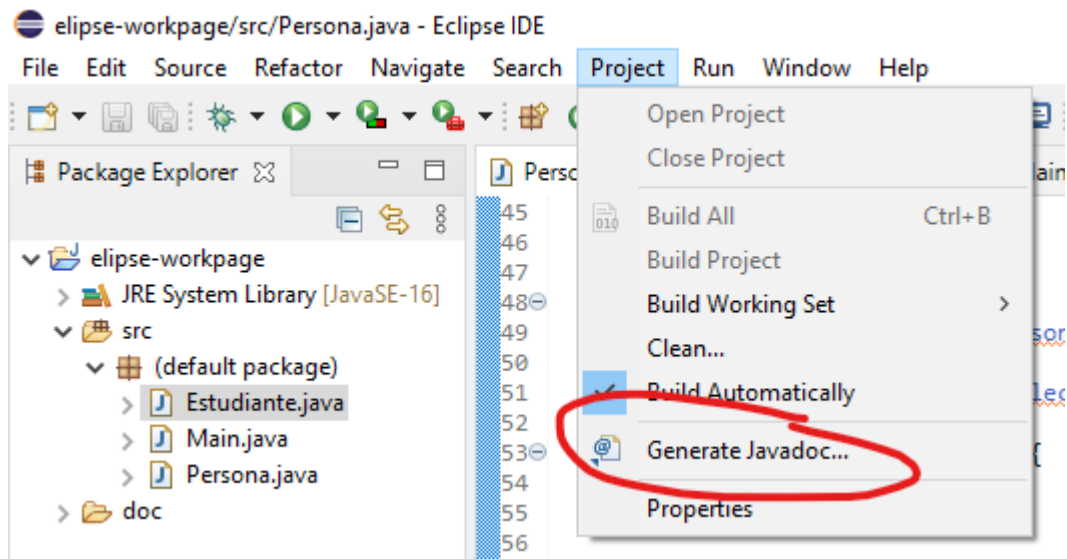
/**
 * Revisa si la edad de la persona es válida.
 *
 * @throws Exception Si la edad es un número negativo.
 */
public void revisarEdad() throws Exception {
    if (edad < 0) {
        throw new Exception("La edad no puede ser un número negativo.");
    } else {
        System.out.println("La edad es válida.");
    }
}

```

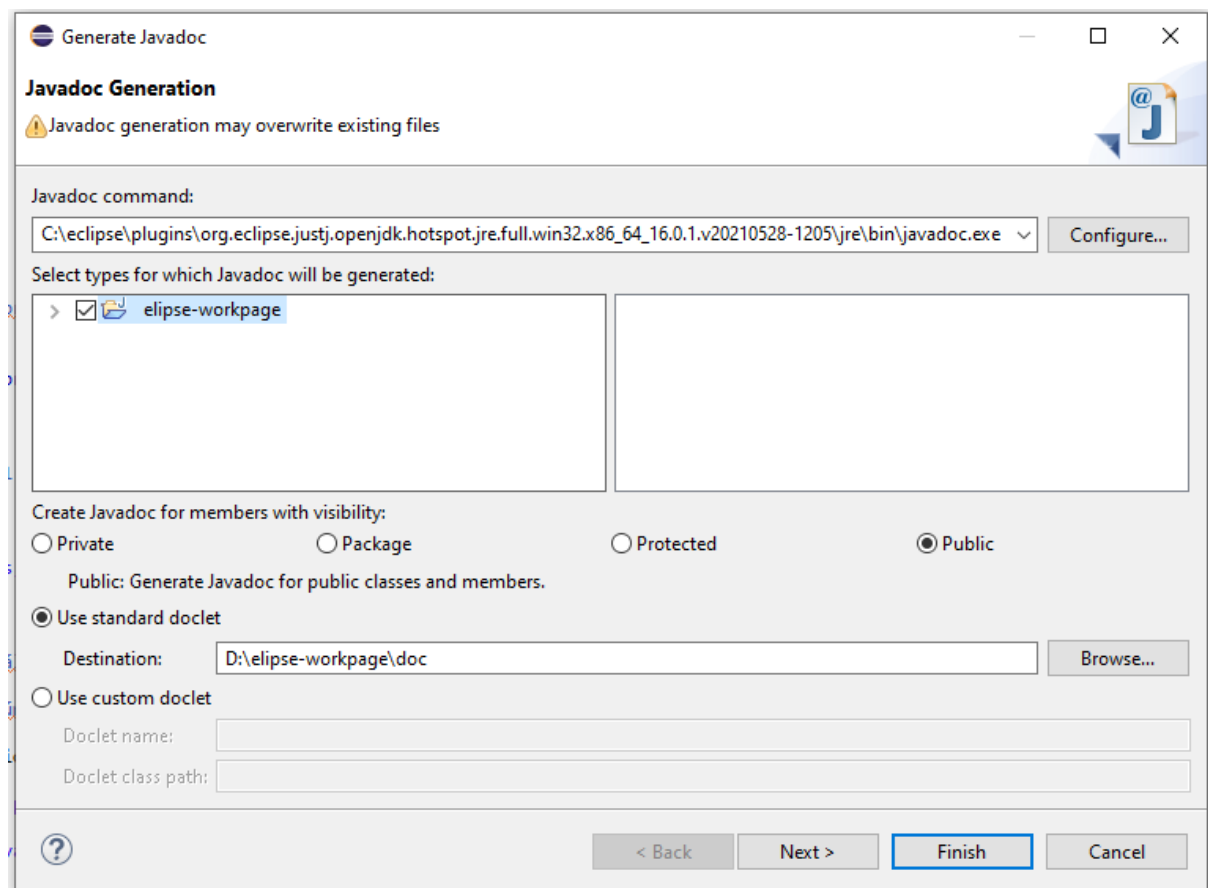
- **@see** : Asocia con otro método o clase.
- **@serial** : Describe el significado del campo y sus valores aceptables. Otras formas válidas son @serialField y @serialData.
- **@deprecated** : Indica que el método o clase es antigua y que no se recomienda su uso porque posiblemente desaparecerá en versiones posteriores.

2. Creación del JavaDoc

Una vez que tenemos creado el proyecto de Java, "EclipseIDE" nos da una opción para generar el JavaDoc:

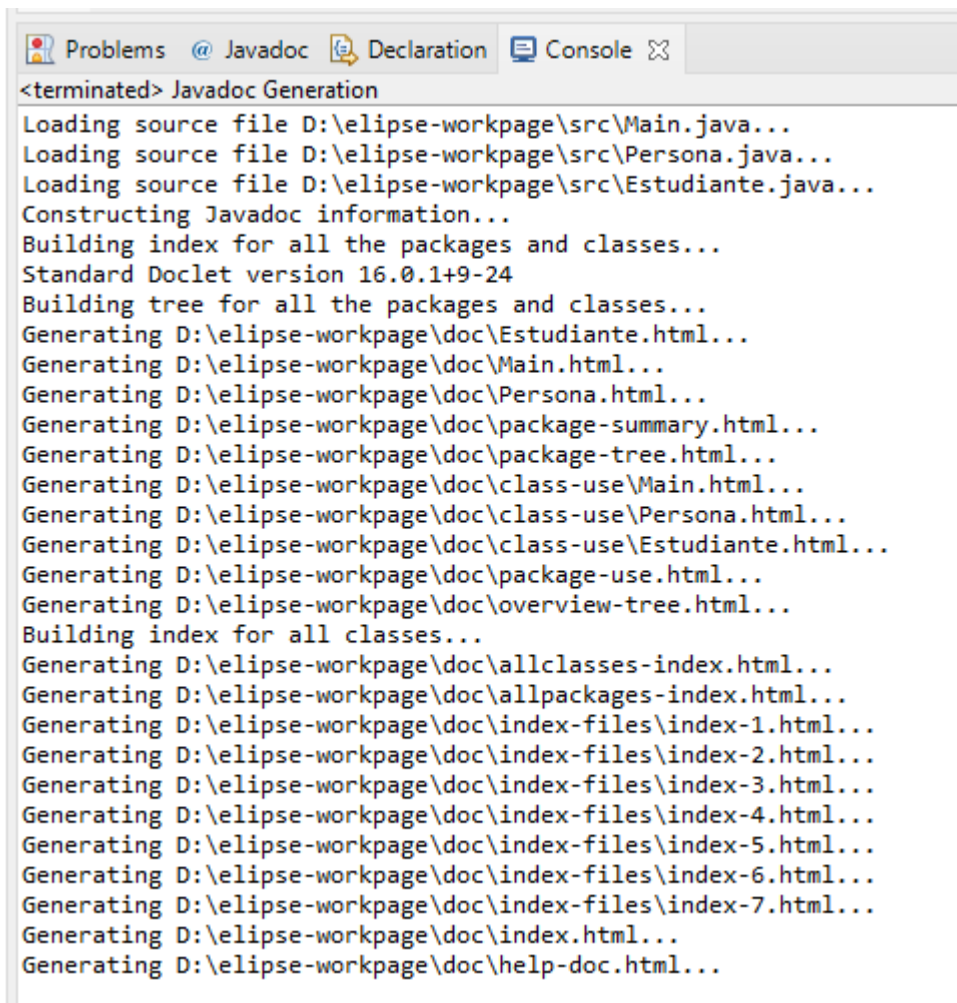


A continuación, nos saldrá esta pestaña, que básicamente es un asistente para crear y generar el Javadoc:



Le indicaremos qué proyecto queremos generar en Javadoc y el destino, es decir, donde quiere guardar los archivos de Javadoc.

Una vez que tenemos elegidos donde queremos guardarlo, le daremos a "Finish" y en la consola del IDE, nos aparecerá esto:



```
<terminated> Javadoc Generation
Loading source file D:\elipse-workpage\src\Main.java...
Loading source file D:\elipse-workpage\src\Persona.java...
Loading source file D:\elipse-workpage\src\Estudiante.java...
Constructing Javadoc information...
Building index for all the packages and classes...
Standard Doclet version 16.0.1+9-24
Building tree for all the packages and classes...
Generating D:\elipse-workpage\doc\Estudiante.html...
Generating D:\elipse-workpage\doc\Main.html...
Generating D:\elipse-workpage\doc\Persona.html...
Generating D:\elipse-workpage\doc\package-summary.html...
Generating D:\elipse-workpage\doc\package-tree.html...
Generating D:\elipse-workpage\doc\class-use\Main.html...
Generating D:\elipse-workpage\doc\class-use\Persona.html...
Generating D:\elipse-workpage\doc\class-use\Estudiante.html...
Generating D:\elipse-workpage\doc\package-use.html...
Generating D:\elipse-workpage\doc\overview-tree.html...
Building index for all classes...
Generating D:\elipse-workpage\doc\allclasses-index.html...
Generating D:\elipse-workpage\doc\allpackages-index.html...
Generating D:\elipse-workpage\doc\index-files\index-1.html...
Generating D:\elipse-workpage\doc\index-files\index-2.html...
Generating D:\elipse-workpage\doc\index-files\index-3.html...
Generating D:\elipse-workpage\doc\index-files\index-4.html...
Generating D:\elipse-workpage\doc\index-files\index-5.html...
Generating D:\elipse-workpage\doc\index-files\index-6.html...
Generating D:\elipse-workpage\doc\index-files\index-7.html...
Generating D:\elipse-workpage\doc\index.html...
Generating D:\elipse-workpage\doc\help-doc.html...
```


























Nos indicará que el Javadoc ya está generado y listo para ver.

3. Explicación del Javadoc

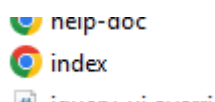
Link del Javadoc:

<https://github.com/xzhou12/m08-despl-app-web/tree/UF1/UF1/Practica%203%20JavaDoc>

Una vez creado el Javadoc, nos creará estos diferentes archivos:

<div> <div> <div></div> <div>Este equipo > Dades (D:) > elipse-workpage > doc ></div> </div> <div> <div></div> <div></div> </div> <div> <div></div> </div> </div>				
	Nombre	Fecha de modificación	Tipo	Tamaño
do	 class-use	16/10/2023 19:07	Carpeta de archivos	
	 index-files	16/10/2023 19:07	Carpeta de archivos	
	 resources	16/10/2023 19:07	Carpeta de archivos	
	 script-dir	16/10/2023 19:07	Carpeta de archivos	
tos	 allclasses-index	16/10/2023 19:07	Chrome HTML Do...	4 KB
	 allpackages-index	16/10/2023 19:07	Chrome HTML Do...	3 KB
	 element-list	16/10/2023 19:07	Archivo	1 KB
	 Estudiante	16/10/2023 19:07	Chrome HTML Do...	12 KB
tos	 help-doc	16/10/2023 19:07	Chrome HTML Do...	7 KB
	 index	16/10/2023 19:07	Chrome HTML Do...	1 KB
	 jquery-ui.overrides	16/10/2023 19:07	Archivo de origen ...	2 KB
	 Main	16/10/2023 19:07	Chrome HTML Do...	10 KB
D	 member-search-index	16/10/2023 19:07	Archivo de origen ...	1 KB
	 module-search-index	16/10/2023 19:07	Archivo de origen ...	1 KB
	 overview-tree	16/10/2023 19:07	Chrome HTML Do...	3 KB
	 package-search-index	16/10/2023 19:07	Archivo de origen ...	1 KB
il (C:)	 package-summary	16/10/2023 19:07	Chrome HTML Do...	4 KB
	 package-tree	16/10/2023 19:07	Chrome HTML Do...	3 KB
	 package-use	16/10/2023 19:07	Chrome HTML Do...	3 KB
	 Persona	16/10/2023 19:07	Chrome HTML Do...	18 KB
	 script	16/10/2023 19:07	Archivo de origen ...	6 KB
	 search	16/10/2023 19:07	Archivo de origen ...	14 KB
	 stylesheet	16/10/2023 19:07	Archivo de origen ...	19 KB
	 tag-search-index	16/10/2023 19:07	Archivo de origen ...	1 KB
	 type-search-index	16/10/2023 19:07	Archivo de origen ...	1 KB

Para poder visualizar el JavaDoc, tendremos que entrar en el “index.html”



Y nos aparecerá este diseño:

Class Main
 java.lang.Object[Ⓜ]
 Main

public class **Main**
 extends Object[Ⓜ]

Esta es la clase principal que contiene el método main para probar las clases Persona y Estudiante.

Author:
 Xiaobin Zhou

Constructor Summary

Constructor	Description
Main()	

Method Summary

Modifier and Type	Method	Description
static void	main (String [Ⓜ] [] args)	Punto de entrada para la aplicación.

Methods inherited from class java.lang.Object[Ⓜ]
 equals[Ⓜ], getClass[Ⓜ], hashCode[Ⓜ], notify[Ⓜ], notifyAll[Ⓜ], toString[Ⓜ], wait[Ⓜ], wait[Ⓜ], wait[Ⓜ]

Constructor Details

Main

public Main()

Con el JavaDoc, nos enseñará como está estructurado el proyecto de Java, nos mostrará información sobre las funciones, las variables, los parámetros, los returns, ...

También nos muestra la estructura que sigue los objetos en el proyecto:

Hierarchy For Package <Unnamed>

Class Hierarchy

- java.lang.Object[Ⓜ]
 - **Main**
 - **Persona**
 - **Estudiante**

Con esta información podemos deducir, que la clase “Estudiante” depende de la clase “Persona”, es decir, que hereda las variables de Persona.

En cada página de cada clase, nos muestra un pequeño resumen de cada método/función que tiene el objeto.

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
void	<code>birthday()</code>	Incrementa la edad de la persona en 1 y muestra un mensaje de cumpleaños.
int	<code>getEdad()</code>	Obtiene la edad de la persona.
<code>String</code>	<code>getNombre()</code>	Obtiene el nombre de la persona.
void	<code>revisarEdad()</code>	Revisa si la edad de la persona es válida.
<code>String</code>	<code>saludar()</code>	Saluda a la persona.
void	<code>setEdad(int edad)</code>	Establece la edad de la persona.
void	<code>setNombre(String nombre)</code>	Establece el nombre de la persona.
Methods inherited from class <code>java.lang.Object</code>		
<code>equals</code> , <code>getClass</code> , <code>hashCode</code> , <code>notify</code> , <code>notifyAll</code> , <code>toString</code> , <code>wait</code> , <code>wait</code> , <code>wait</code>		

También podemos ver las diferentes funciones que tiene cada objeto con la diferente información que le hemos introducido en el proyecto de Java anteriormente.

Method Details

getNombre

```
public StringⓂ getNombre()
```

Obtiene el nombre de la persona.

Returns:

El nombre de la persona.

setNombre

```
public void setNombre(StringⓂ nombre)
```

Establece el nombre de la persona.

Parameters:

nombre - El nombre a establecer.

getEdad

```
public int getEdad()
```

Obtiene la edad de la persona.

Returns:

La edad de la persona.