PRÁCTICA JAVADOC EN ECLIPSE Y NETBEANS

IES VALLE DEL JERTE

PROFESOR: ABEL LÓPEZ RUIZ

CURSO ACADÉMICO: 2024-2025

MÓDULO: ENTORNOS DE DESARROLLO

GRADO SUPERIOR: DAM, DAW

1.INTRODUCCIÓN

Javadoc es una herramienta que proporciona Java para crear la documentación de un proyecto, para documentar todos los métodos de una clase y los atributos de los que dispone, para conseguir una documentación que lo pueda comprender con facilidad otro desarrollador que vaya a utilizar nuestras clases es aconsejable que se haga una documentación comprensible para todos.

Antes de generar la documentación mediante los IDE's Netbeans y Eclipse, se crearán una serie de comentarios con /* */ antes de los métodos añadiendo una descripción del método, el autor si se desea, los parámetros que se introducen y lo que devuelve el método, hay una serie de anotaciones que entiende Javadoc para realizar la documentación que se mostrará en la siguiente tabla:

Syntax	Description	Description
@author	Es el autor de la clase	@author name-text
{@code}	Muestra texto en formato de código sin que sea interpretado como código HTML	{@code text}
{@docRoot}	Indica la ruta relativa al directorio raíz del documento generado desde cualquier página generada.	{@docRoot}
@deprecated	Se pone indicando que esta API no debe usarse más	@deprecated deprecatedtext
@exception	Se indica cuando es vulnerable a lanzar una excepción, en seguida se ponen las clases de las excepciones posibles	@exception class-name description
{@inheritDoc}	Indica la herencia o implementación procedentora	
{@link}	Hace un enlace a la miembro indicado	{@link package.class#member label}
{@linkplain}	Lo mismo que en anterior excepto que la etiqueta del enlace se muestra en texto plano	{@linkplain package.class#member label}
@param	Añade parámetros con nombres específicos, seguido de su descripción.	@param parameter-name description
@return	Asigna un parámetro de retorno, seguido de su descripción	@return description
@see	Se añade para manejar referencias, o información relacionada	@see reference
@serial	Se utiliza para indicar campos o clases serializables.	@serial field-description include exclude
@serialData	Se utiliza para documentar métodos que generan una serialización	@serialData data-description
@serialField	Se utiliza para documentar objetos serializados	@serialField field-name field-type field-description
@since	Se añade en el encabezado para especificar desde cuándo fue creado	@since release
@throws	Es sinónimo con la etiqueta @exception	@throws class-name description
{@value}	Cuando es usado sin argumento se usa para especificar un campo estático en otro caso se usa para mostrar el valor constante.	{@value package.class#field}
@version	Se añade en el subtítulo con la versión especificada.	@version version-text

En otras versiones más antiguas de Javadoc había menos etiquetas disponibles pero mayormente se usarán las etiquetas @param y @return, pero se pueden usar todas las que se deseen para conseguir que sea más entendible para otros desarrolladores. Un ejemplo de ello de cómo se hace en el código fuente es lo que se muestra en la siguiente imagen:

```
     * Method to execute the request to the server
     * @param message Object to send to the specified protocol
     * @param host destination host
     * @param port destination port number
     * @return Message returned by the server
     * @since JSockets 1.0.0
     */
    public byte [] executeRequest(Object message, String host, Integer port)
    {
        byte [] resultado = null;

        try
        {
            connectToTheServer(host, port);

            out = new ObjectOutputStream(client.getOutputStream());
            out.flush();
            in = new ObjectInputStream(client.getInputStream());

            sendData(message);

            resultado = readData();
        }
        catch (EOFException excepcionEOF)
```

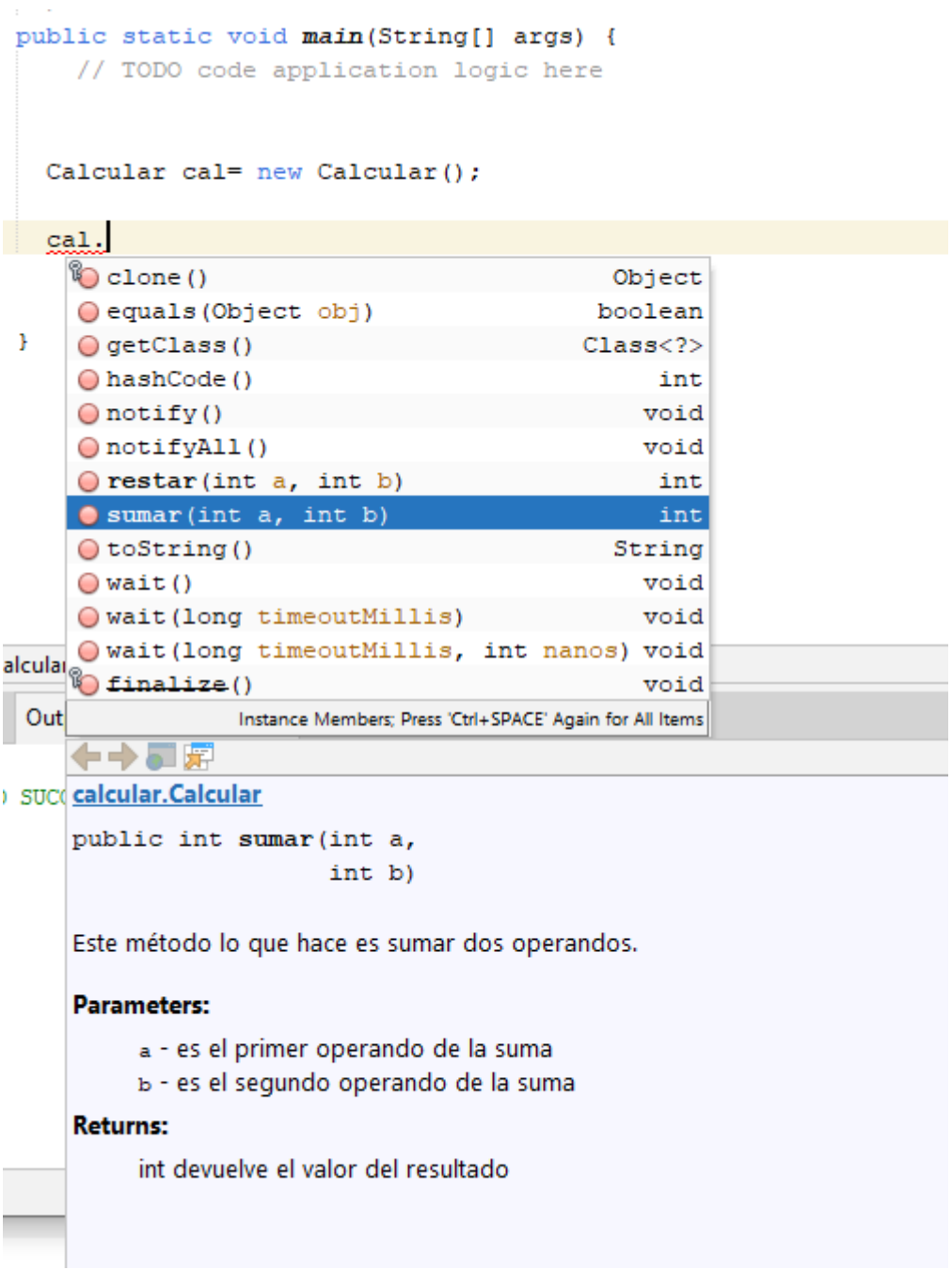
2. JAVADOC CON NETBEANS

En este apartado se aprenderá a generar la documentación de un proyecto en Netbeans mediante Javadoc.

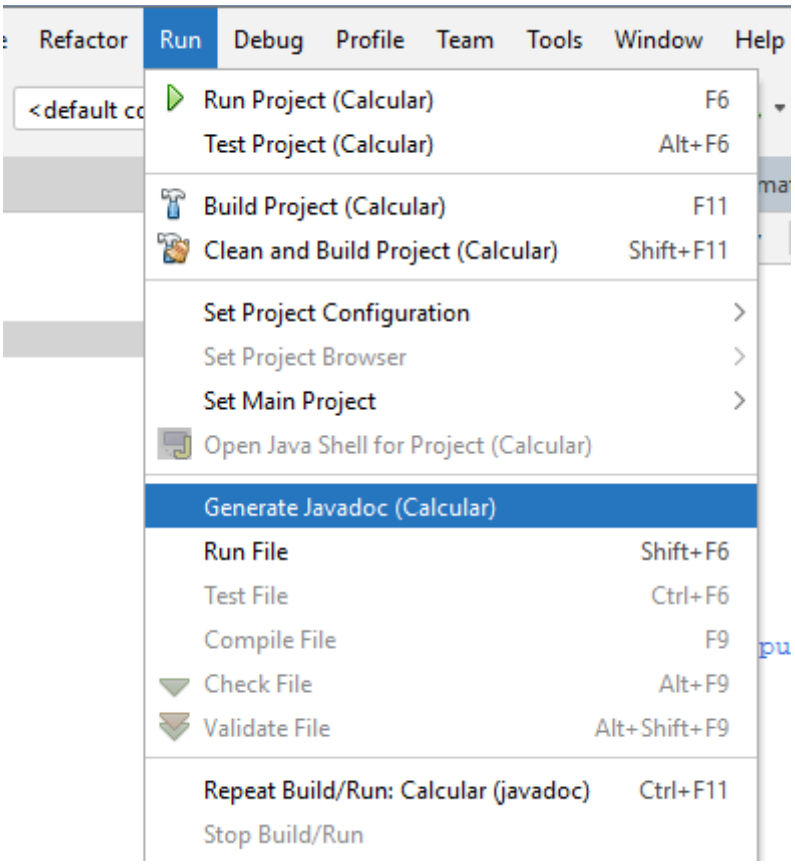
Para ello, se tomará un proyecto de ejemplo, en este caso un ejemplo sencillo como una calculadora y un ejemplo de los comentarios se muestra en la siguiente imagen:

```
public class Calcular {


    /**
    * Este método lo que hace es sumar dos operandos.
    * @autor abel
    * @param a es el primer operando de la suma
    * @param b es el segundo operando de la suma
    * @return int devuelve el valor del resultado
    **/
    public int sumar(int a, int b)
    {

        int suma;

        suma=a+b;

        return suma;


    }
```

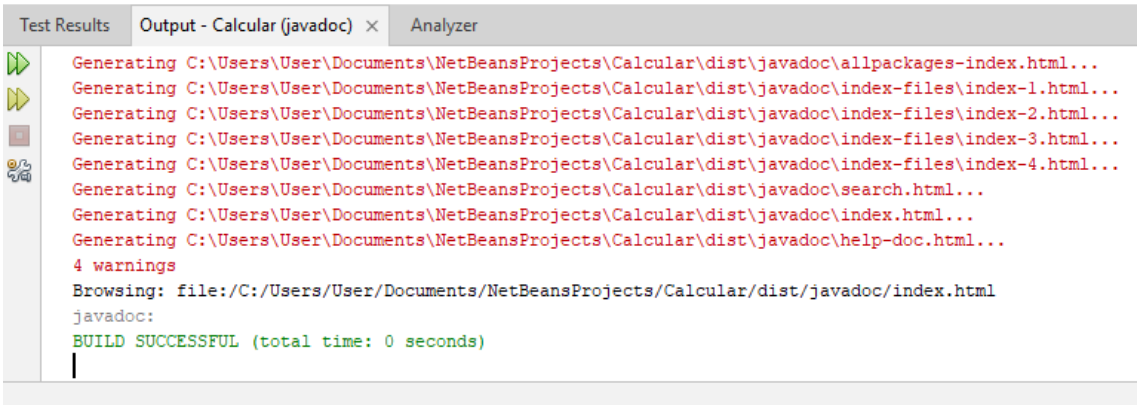
Si ahora se llama al método en el main se puede comprobar que aparece la documentación que se ha puesto:



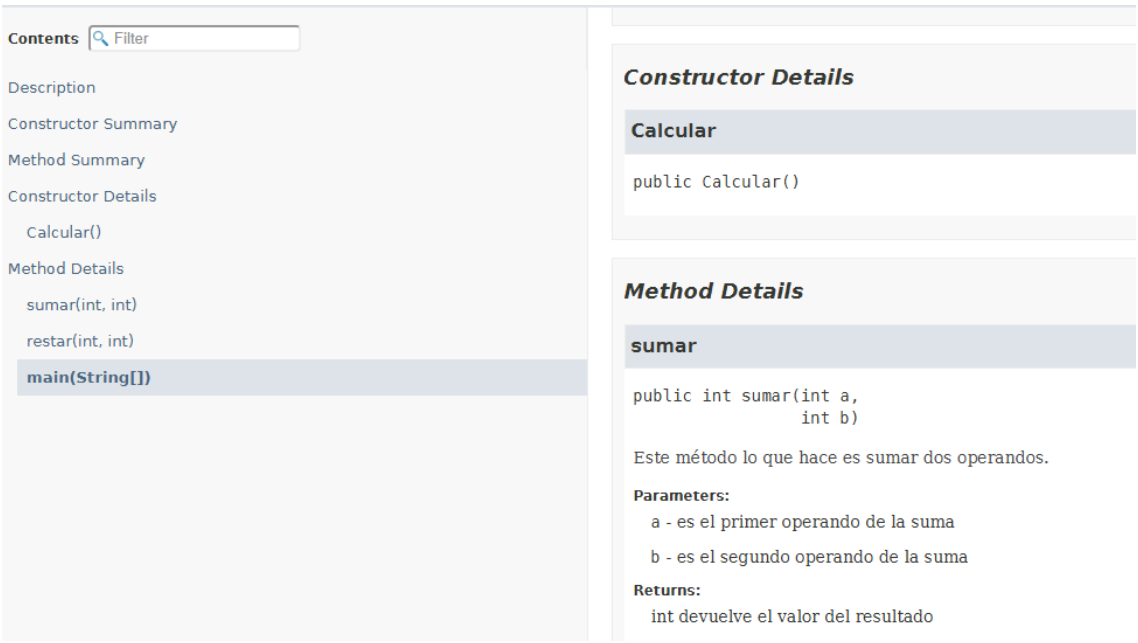
Para general el javadoc, me generará la documentación en html, se va al siguiente menú en Netbeans:



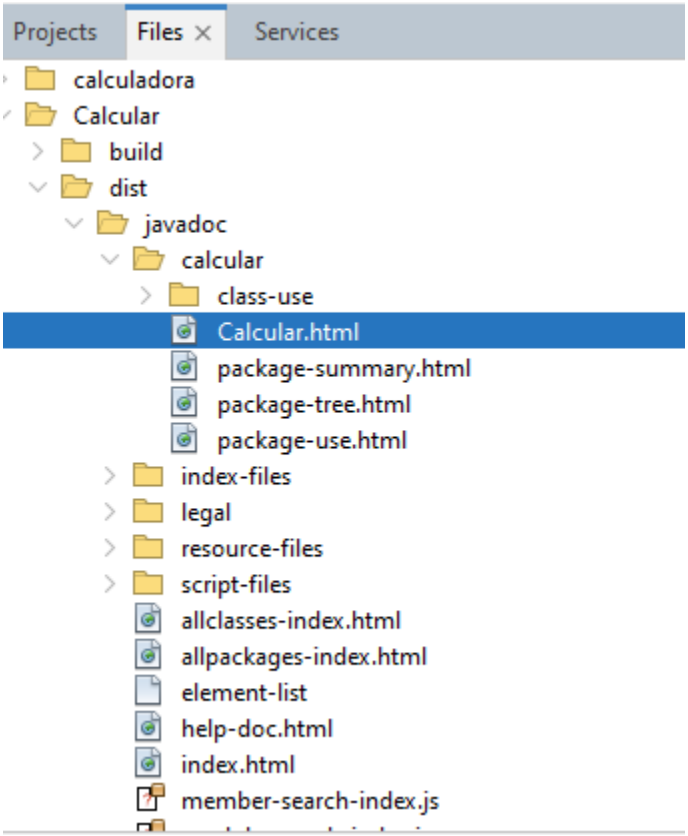
En la ventana Output de salida aparecerá los pasos que se han realizado para generar el Javadoc:



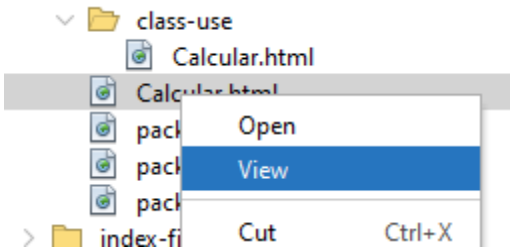
Para poder ver la documentación se abre una ventana en el navegador en la que aparece la documentación generada de mi proyecto (en el navegador se puede ver la ruta donde se ha guardado la documentación de mi proyecto):



De todas formas, la documentación se guarda por defecto en la carpeta dist (distribución) del proyecto, y ahí se puede encontrar el html.



Para poder verlo en el navegador, se puede hacer entonces:



3. JAVADOC CON ECLIPSE

Se procede de manera análoga al caso anterior:

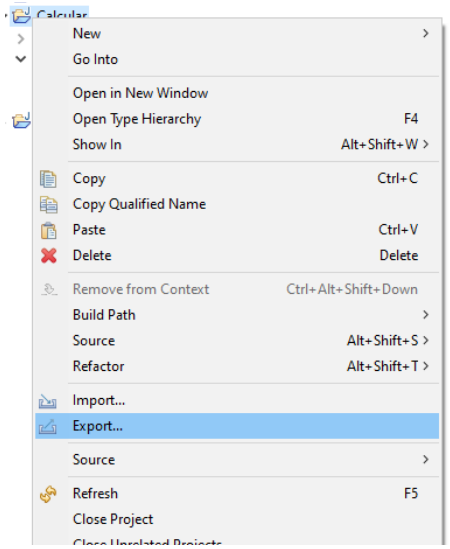
```
public class Calcular {

    /**
     *
     *
     * Es un Método que calcula la suma de dos números
     * @param a es el primer operando de la suma
     * @param b es el segundo operando de la suma
     * @return int devuelve el resultado de la suma
     **/


    public int sumar(int a, int b)
    {

        int suma;

        suma=a+b;

        return suma;


    }
```

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    Calcular cal= new Calcular();

    cal.
}
```

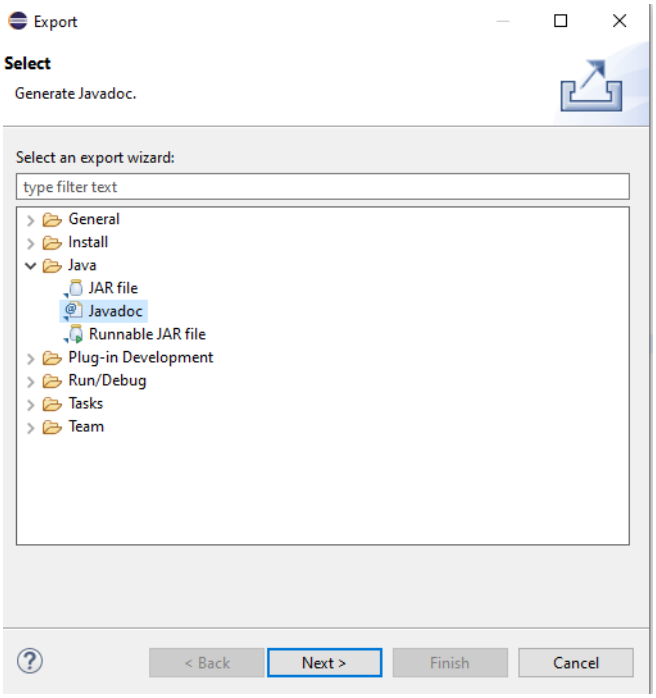
clone() : Object - Object
equals(Object obj) : boolean - Object
finalize() : void - Object
getClass() : Class<?> - Object
hashCode() : int - Object
notify() : void - Object
notifyAll() : void - Object
sumar(int a, int b) : int - Calcular
toString() : String - Object
wait() : void - Object
wait(long timeoutMillis) : void - Object
wait(long timeoutMillis, int nanos) : void - Object
Press 'Ctrl+Space' to show Template Proposals

Es un Método que calcula la suma de dos números
Parameters:
a es el primer operando de la suma
b es el segundo operando de la suma
Returns:
int devuelve el resultado de la suma

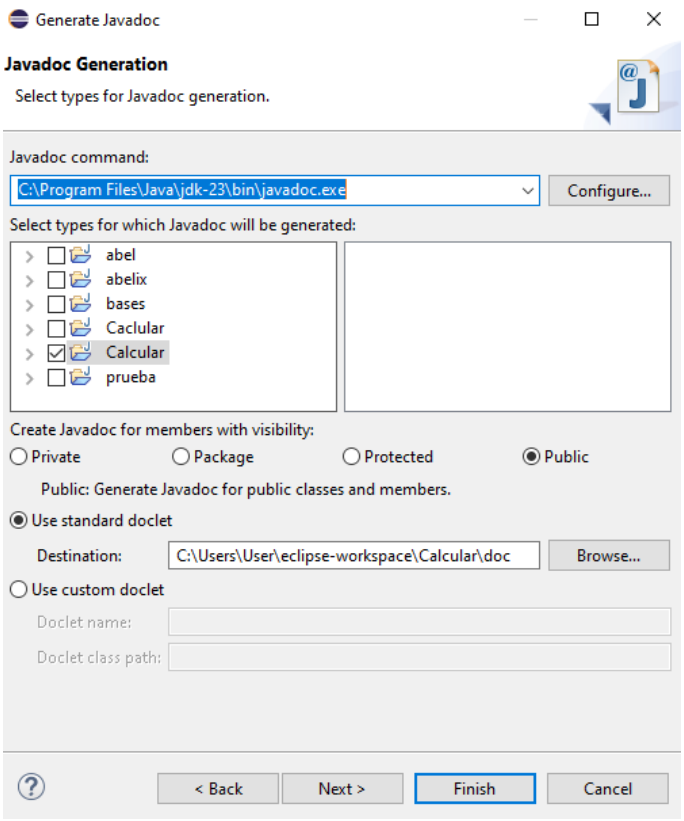
Para generar el Javadoc en Eclipse se procederá de forma distinta que en el caso de Netbeans, primeramente se pincha sobre el proyecto, click derecho y luego en el menú a exportar:



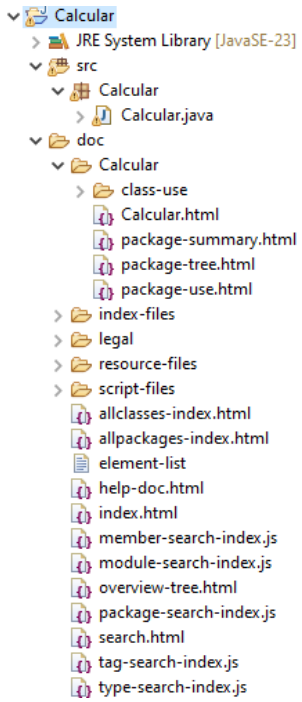
Después se pincha en Java y en Javadoc, como se muestra en la siguiente imagen:



Se pulsa a next y se elige la carpeta destino donde se va a guardar la documentación y el proyecto al que quiero generarle la documentación:

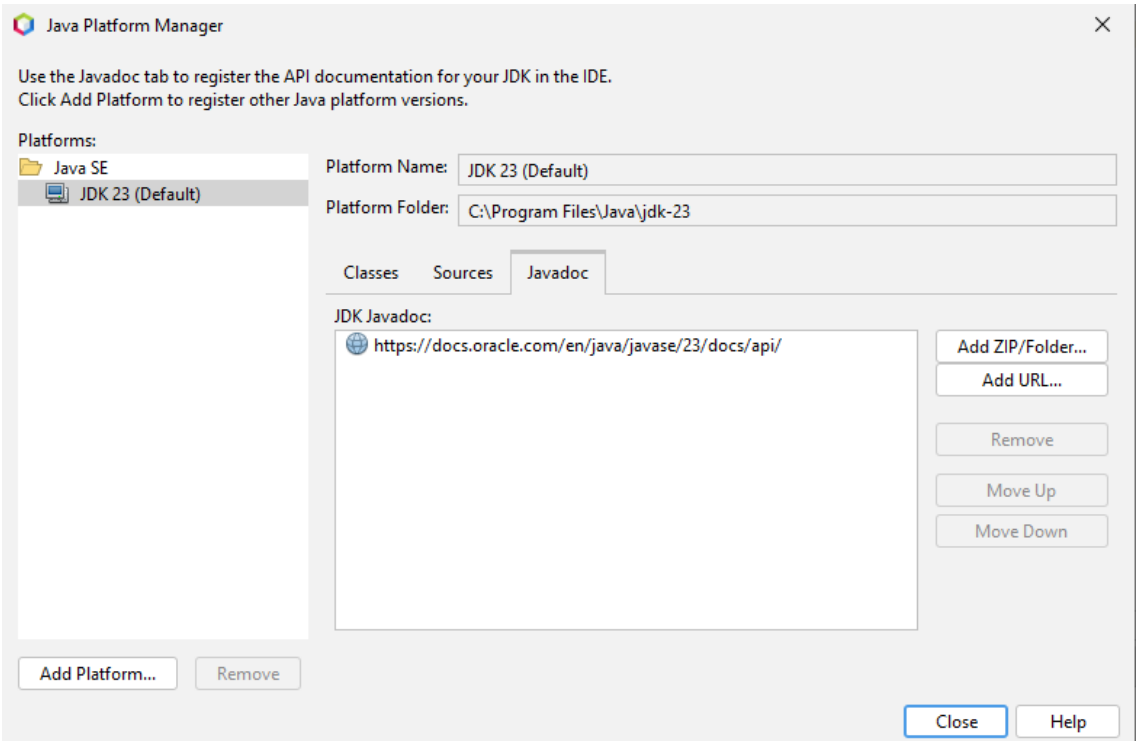


En el caso de Eclipse como se ha puesto que la carpeta de documentación se guarde en el proyecto, aparecerá en el menú de proyectos:



Para poder verlo en el navegador se puede hacer de la siguiente forma:

En Netbeans, en el menú tools y luego Java Platform, y en la pestaña Javadoc se puede elegir una carpeta donde se guarde la documentación en un zip:



4. EJERCICIO DE JAVADOC

Se le propone al alumno realizar la generación de la documentación del siguiente proyecto tanto en Netbeans tanto como en Eclipse:

```
package caracteres;


/**
 *
 * @author User
 */
public class Caracteres {



    public int Contadore(String nombre)
    {
        int total=0;

        int i=0;

        while (i<nombre.length())
        {
            if ( (nombre.charAt(i)=='e') || (nombre.charAt(i)=='E'))
            {

                total++;

            }

        i++;

        }
```

```
    return total;

}

public int contvocales(String nombre)
{

int total=0;

    int i=0;

    while (i<nombre.length())
    {
        if ( (nombre.charAt(i)=='e') || (nombre.charAt(i)=='E') || (nombre.charAt(i)=='a') || (nombre.charAt(i)=='A') || (nombre.charAt(i)=='i') || (nombre.charAt(i)=='I') || (nombre.charAt(i)=='U') || (nombre.charAt(i)=='u') || (nombre.charAt(i)=='O') || (nombre.charAt(i)=='o'))
        {

            total++;

        }

    i++;
```

```
    }



    return total;



}

public int contCon(String nombre)
{

    int total=0;



    total=nombre.length()-contvocales(nombre);

    return total;


}
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        Caracteres c= new Caracteres();

        int n= c.Contadore("La casa es de uno nuevo");

        System.out.println(n);

    }

}
```