DOCUMENTACIÓN DEL CÓDIGO CON JAVADOC

QUÉ ES UN COMENTARIO

COMENTAR VS DOCUMENTAR

- ¿Es lo mismo?
- Documentar es el objetivo, comentar es la herramienta.
- Con los comentarios, entre otras herramientas, conseguimos documentar una aplicación.
- Otras herramientas: diagramas, gráficos, esquemas, otros textos, …

COMENTARIO DE CÓDIGO

- Añadir *metainformación* a nuestro código fuente que ayude a entender:
  - Qué es lo que hace
  - Por qué motivo se hace así

COMENTARIO DE CÓDIGO

- En Java
  - Comentario en una línea: `// Esto es un comentario`
  - Comentario multilínea:

```
/*
    Esto es un comentario
    en más de una línea
*/
```

COMENTARIO DE CÓDIGO

- En Java
  - Comentario Javadoc:

```
/**
 *  Esto es un comentario Javadoc
 *  @param Un parámetro
 *  @return Valor de retorno
 */
```

DOCUMENTACIÓN DEL CÓDIGO CON JAVADOC

CUÁNDO ES NECESARIO COMENTAR

NECESIDAD DE COMENTAR

- Es algo que provoca controversia entre los expertos.
- Lo que sí queda claro es que es beneficioso:
  - Entender la utilidad de los diferentes módulos de una aplicación.
  - Conocer las decisiones de diseño de código, sobre todo aquellas que no son obvias.

ALGUNAS MÁXIMAS EN PROGRAMACIÓN

- Todos los programas tienen errores y descubrirlos sólo es cuestión de tiempo y de que el programa tenga éxito y se utilice frecuentemente. (Mantenimiento correctivo)
- Todos los programas sufren modificaciones a lo largo de su vida, al menos todos aquellos que tienen éxito. (Mantenimiento evolutivo)

Si tu programa no merece ser documentado,

probablemente no merezca ser ejecutado

J. Nagler

Si tienes necesidad de documentar tu código,

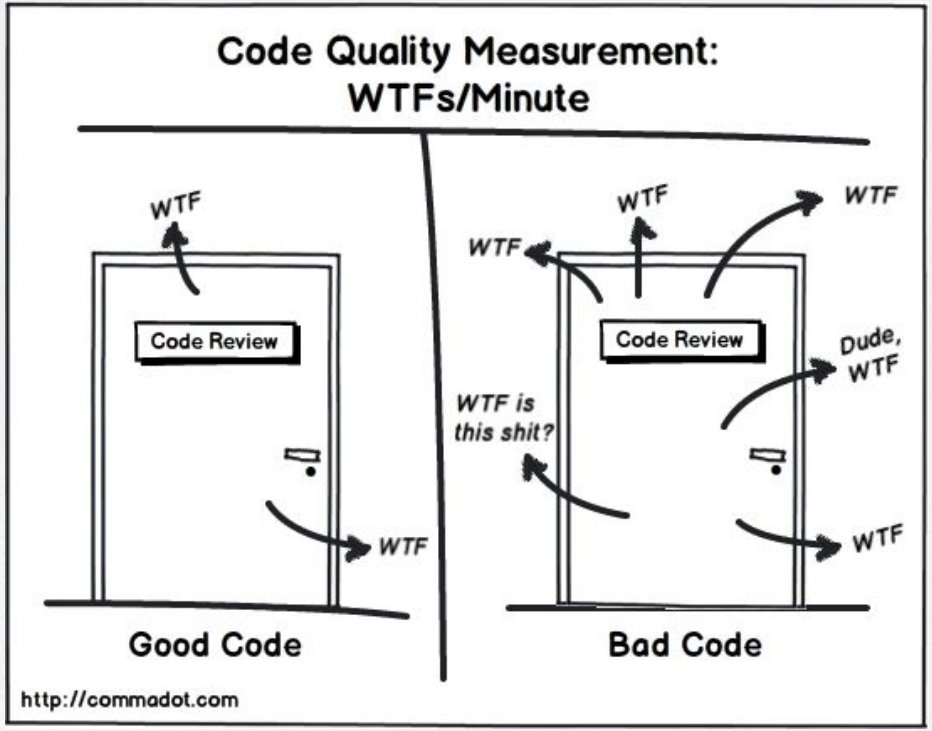
probablemente no seas buen programador

Brian Kernighan

Los comentarios redundantes son solo lugares para recopilar mentiras y desinformación

Robert C. Martin
https://www.goodreads.com/author/quotes/45372.Robert_C_Martin

GOOD CODE VS BAD CODE



CÓDIGO AUTOCOMENTADO (CLEAN CODE)

- *Clean Code*. Libro de Robert C. Martin
- El primer nivel de documentación es un buen uso del lenguaje de programación.
- Los comentarios solo están justificados cuando no somos capaces de expresarnos con el código.
- Hay comentarios que no aportan nada.

CÓDIGO AUTOCOMENTADO (CLEAN CODE)

```
1  /** * This function reset the given matrix to 0; */
2  public void clean(int[][] m, int i, int j){
3      for(int a;a < i;a++){
4          for(int b;b < j;b++){
5              m[a][b] = 0;
6          }
7      }
8  }
```

CÓDIGO AUTOCOMENTADO (CLEAN CODE)

```
10    public void resetMatrix(int[][] matrix, int rows_size, int columns_size){
11        for(int current_row ; current_row < rows_size ; current_row++){
12            for(int current_col ; current_col < columns_size ; current_col++){
13                matrix[current_row][current_col] = 0;
14            }
15        }
16    }
```

CÓDIGO AUTOCOMENTADO (CLEAN CODE)

- Nombra adecuadamente las variables, los métodos y las clases.
- Unidades de código pequeñas.
- Principio de Responsabilidad Única (SOLID)
- Controla el número de argumentos (3 como mucho)

CÓDIGO AUTOCOMENTADO (CLEAN CODE)

- Evita el *copy&paste*. Don’t repeat yourself.
- Coding Style.
- Usa excepciones.
- Testea tu código
- Regla del *boy scout*.

COMENTARIOS EN CLEAN CODE

- Comentarios legales, de derechos de autor.
- Retorno de una función
  - Se puede eliminar si el nombre de la función lo incluye
- Cuando usamos librerías de terceros
  - No podemos modificar el nombre de sus funciones

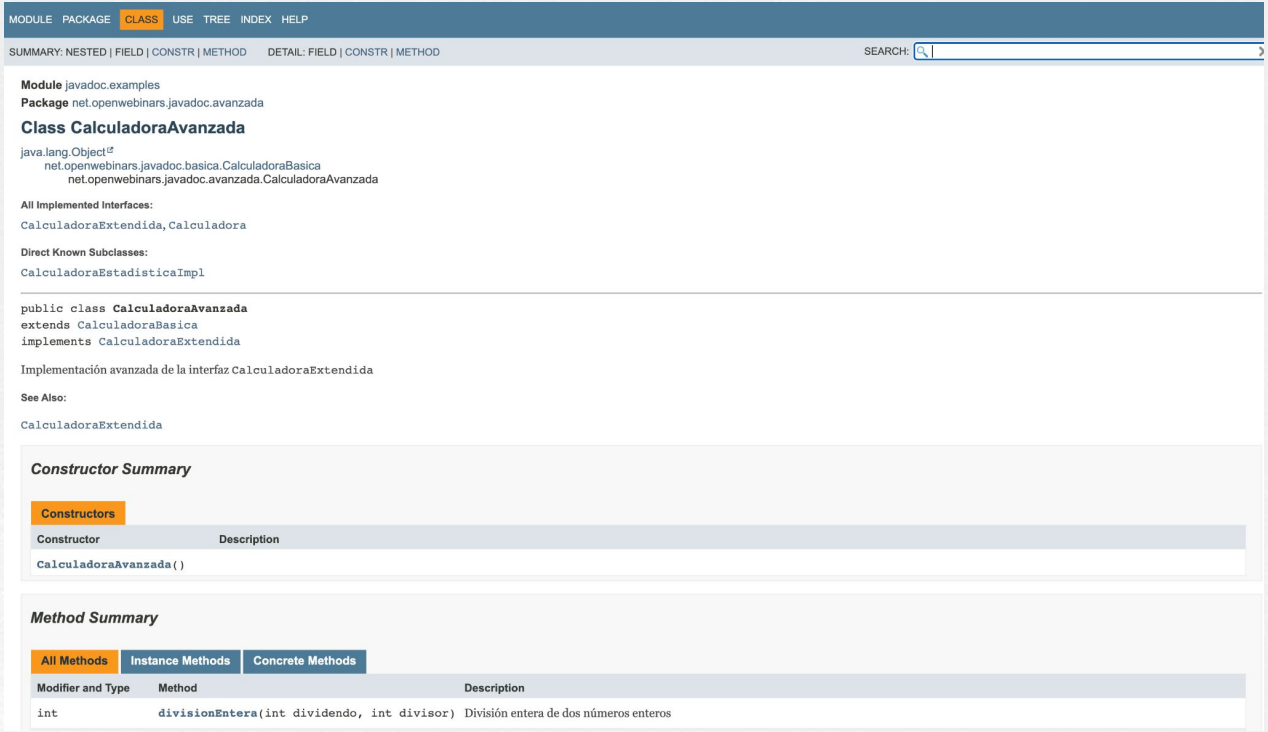
COMENTARIOS EN CLEAN CODE

- Comentarios *TODO*
  - No deben ser una excusa para dejar código incompleto.
- Comentarios para una API pública
  - Si nuestro código va a ser usado como librería de tercero.
  - Tipo Javadoc

DOCUMENTACIÓN DEL CÓDIGO CON JAVADOC

JAVADOC Y SUS ANOTACIONES

¿QUÉ ESPERAMOS OBTENER?



MODULE PACKAGE CLASS USE TREE INDEX HELP

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

Module javadoc.examples
Package net.openwebinars.javadoc.avanzada

Class CalculadoraAvanzada

java.lang.Object
net.openwebinars.javadoc.basica.CalculadoraBasica
net.openwebinars.javadoc.avanzada.CalculadoraAvanzada

All Implemented Interfaces:

`CalculadoraExtendida`, `Calculadora`

Direct Known Subclasses:

`CalculadoraEstadisticaImpl`

```
public class CalculadoraAvanzada
extends CalculadoraBasica
implements CalculadoraExtendida
```

Implementación avanzada de la interfaz `CalculadoraExtendida`

See Also:

`CalculadoraExtendida`

Constructor Summary

Constructors

Constructor	Description
`CalculadoraAvanzada()`	

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
int	`divisionEntera(int dividendo, int divisor)`	División entera de dos números enteros

FORMATO DE UN COMENTARIO DE DOCUMENTACIÓN

- Situado entre /** y */ y con “estructura HTML”.
- Precede la declaración de una interfaz, clase, atributo o método.
- Puede tener dos partes
  - Un texto (con etiquetas HTML)
  - Una serie de **etiquetas**

ETIQUETAS BÁSICAS

- *@author*: autor o autores del bloque de código

```
public interface CalculadoraExtendida
extends Calculadora
```

Extensión de la calculadora básica con otras operaciones

Author:

Luis Miguel López

ETIQUETAS BÁSICAS

- *@version*: versión actual del bloque de código
- *@since*: desde qué versión está disponible

Since:

1.0

Version:

1.1

Author:

Luis Miguel López Magaña

ETIQUETAS BÁSICAS

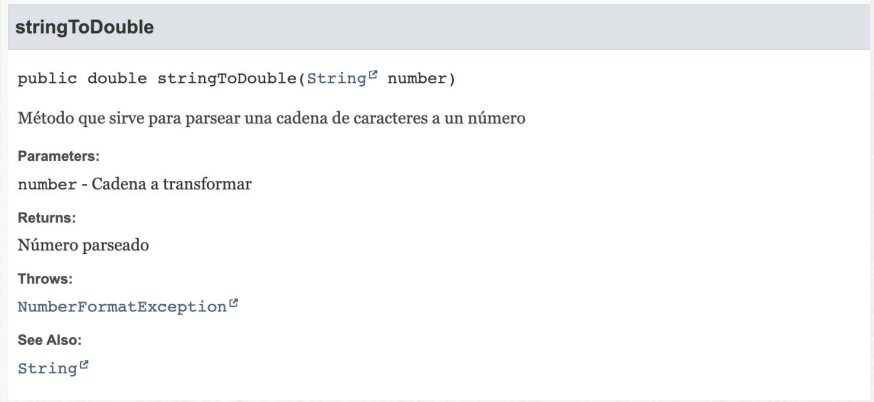
- *@see*: añade un comentario *See Also*. Muy versátil.
- Puede haber más de una por bloque de código.
- Puede incluir:
  - Una interfaz, clase, método, … del proyecto o de otro módulo
  - Un enlace HTML.
  - Texto (título de un libro)

ETIQUETAS BÁSICAS

- *@see* java.lang.String
- *@see* java.lang.String The String class
- *@see* String
- *@see* String#equals(Object)
- *@see* Openwebinars.net
- *@see* Clean Code, by Robert C. Martin.

ETIQUETAS BÁSICAS

- *@throws*, *@exception*: Indica que el bloque de código puede lanzar una excepción específica.
- Puede haber más de una por bloque de código.



stringToDouble

`public double stringToDouble(String number)`

Método que sirve para parsear una cadena de caracteres a un número

Parameters:

`number` - Cadena a transformar

Returns:

Número parseado

Throws:

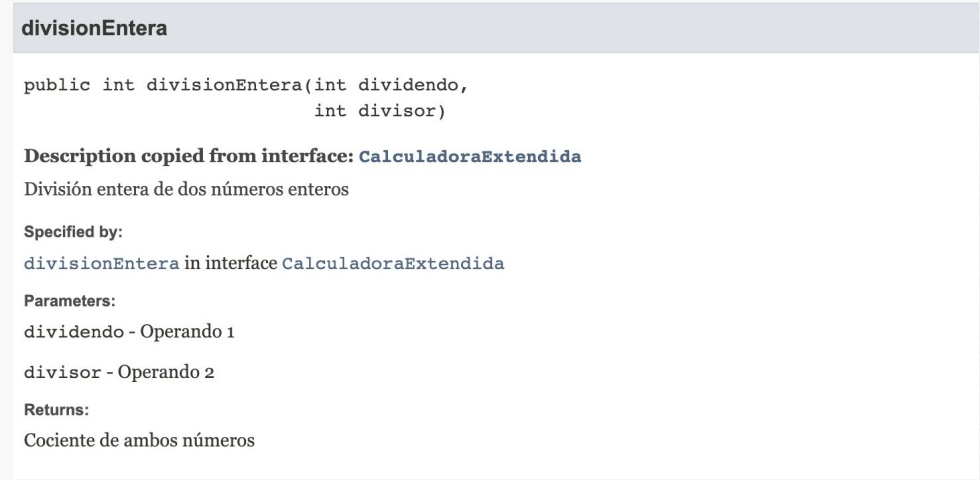
`NumberFormatException`

See Also:

`String`

ETIQUETAS BÁSICAS

- *@param*: argumento que recibe un método.
- ***Debe*** haber una por argumento del método.



divisionEntera

```
public int divisionEntera(int dividendo,
                          int divisor)
```

Description copied from interface: `CalculadoraExtendida`

División entera de dos números enteros

Specified by:

`divisionEntera` in interface `CalculadoraExtendida`

Parameters:

`dividendo` - Operando 1

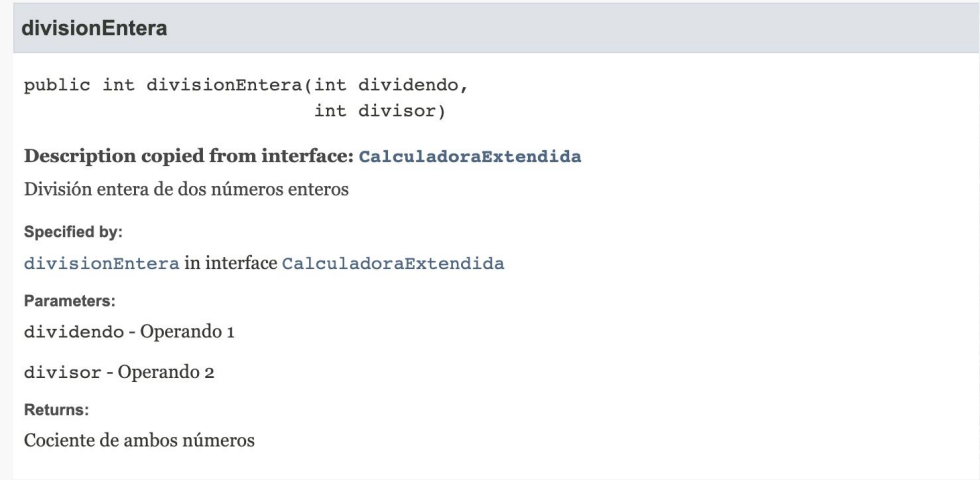
`divisor` - Operando 2

Returns:

Cociente de ambos números

ETIQUETAS BÁSICAS

- *@return:* valor de retorno del método
- ***Puede*** haber una por método.



divisionEntera

```
public int divisionEntera(int dividendo,
                          int divisor)
```

Description copied from interface: `CalculadoraExtendida`

División entera de dos números enteros

Specified by:

`divisionEntera` in interface `CalculadoraExtendida`

Parameters:

`dividendo` - Operando 1

`divisor` - Operando 2

Returns:

Cociente de ambos números

OTRAS ETIQUETAS

- Menos usuales
- @serial, @serialField, @serialData
- Se usan en serialización de objetos.
- Más información en https://docs.oracle.com/javase/7/docs/platform/serialization/spec/serial-arch.html#5251

PAQUETES Y MÓDULOS

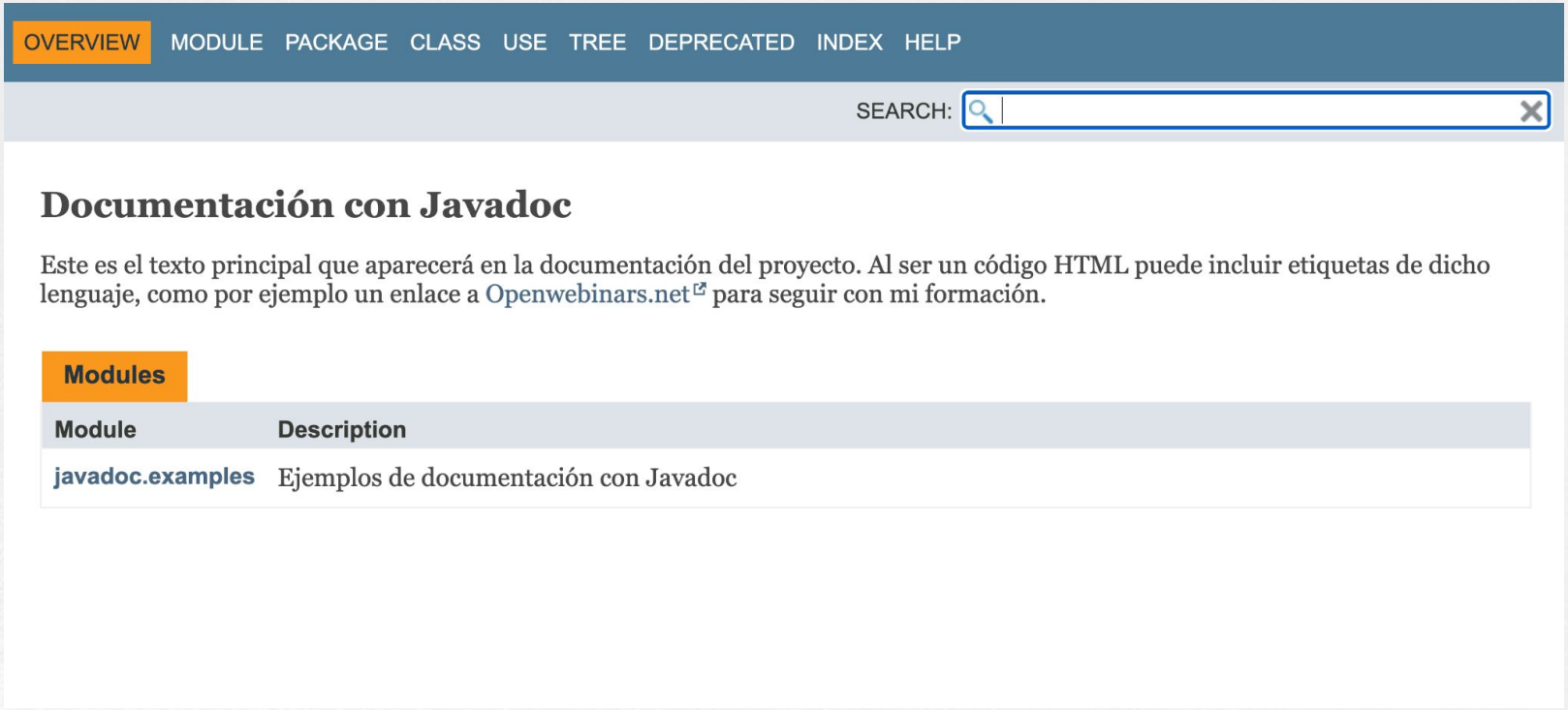
- Para poder documentarlos se generan dos ficheros “adicionales”.
  - package-info.java
  - module-info.java

PAQUETES Y MÓDULOS

OVERVIEW

- Página de bienvenida
- Incluye un texto html y la lista de módulos / paquetes.
- HTML extraído de un documento (overview.html)

OVERVIEW


OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
SEARCH:
Documentación con Javadoc
Este es el texto principal que aparecerá en la documentación del proyecto. Al ser un código HTML puede incluir etiquetas de dicho lenguaje, como por ejemplo un enlace a Openwebinars.net para seguir con mi formación.
Modules
Module
Description
javadoc.examples
Ejemplos de documentación con Javadoc

ÁMBITO DE USO DE LAS ETIQUETAS

Tag	Overview	Module	Package	Class	Method	Field
@see	✅	✅	✅	✅	✅	✅
@link	✅	✅	✅	✅	✅	✅
@since	✅	✅	✅	✅	✅	✅
@deprecated				✅	✅	✅
@author		✅	✅	✅	✅	✅
@version		✅	✅	✅	✅	✅
@param					✅	
@return					✅	
@throws (@exception)					✅	

ALGUNAS RECOMENDACIONES

- Usar <*code*> para palabras clave y nombres (clases, paquetes, métodos, atributos, …).
- Usar @link con mesura.
- Omitir paréntesis para hacer referencia a la forma general de un método.

ALGUNAS RECOMENDACIONES

- Usar frases cortas en lugar de largos párrafos.
- Usar la 3ª persona, no la 2ª.
- Preferiblemente, la descripción de los métodos debe empezar con un verbo.
- Las descripciones de clase/interfaz/campo pueden omitir el sujeto y simplemente indicar el objeto.

MÁS INFO

- https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

DOCUMENTACIÓN DEL CÓDIGO CON JAVADOC

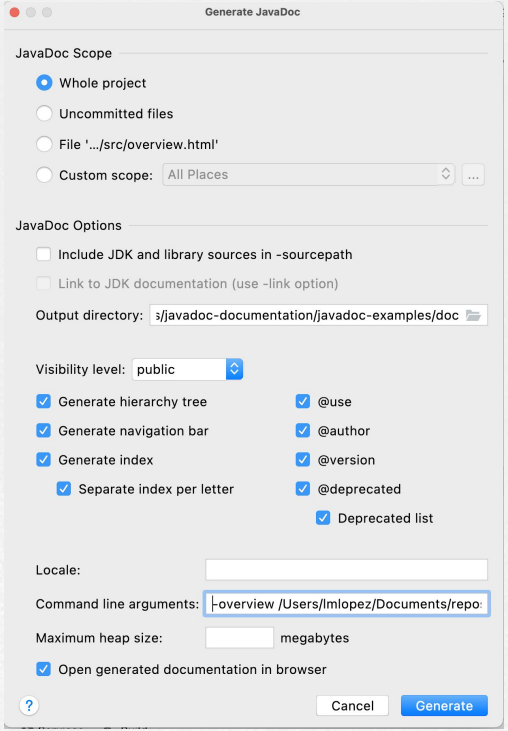
CÓMO GENERAR EL RESULTADO DE NUESTRA DOCUMENTACIÓN

OpenWebinars

GENERAR JAVADOC EN INTELLIJ IDEA

- Tools > Generated JavaDoc
  - *Scope*: Whole Project
  - *Visibility Level*: public
  - Marcar todas las opciones: generate, separate, @XXXX
  - *args*: -overview /ruta/absoluta/al/ovierview.html

GENERAR JAVADOC EN INTELLIJ IDEA



GENERAR JAVADOC EN ECLIPSE

- Export > Java > Generate Javadoc
- Asistente
  - Visibilidad: public
  - Doclet por defecto
  - Todas las opciones básicas y etiquetas
  - Seleccionar ruta a overview.html

GENERAR JAVADOC EN ECLIPSE