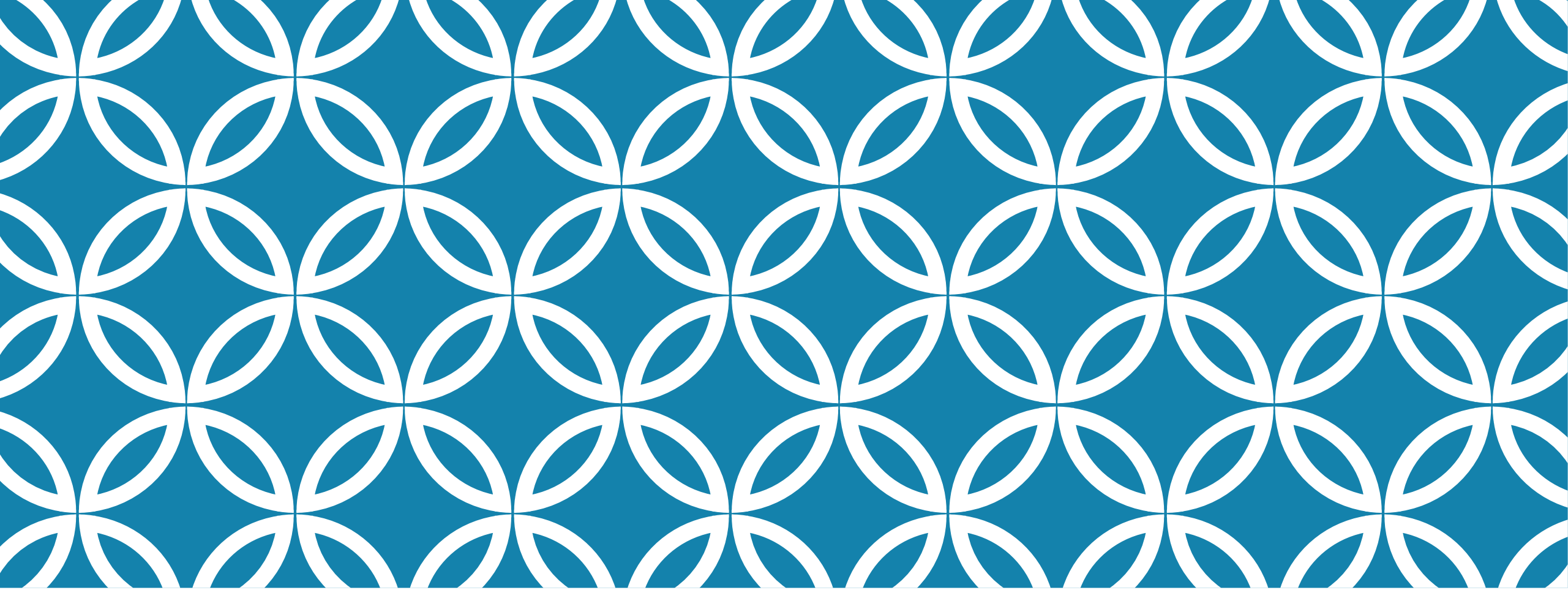


MÓDULO PROFESIONAL ENTORNOS DE DESARROLLO

UD 4.2 – Documentación

JAVADOC

Este tipo de utilidad de creación de documentación a partir de comentarios ha sido desarrollada por Oracle.

Javadoc permite la creación de documentación de API en formato HTML.

API

Del inglés Application Programming Interface (Interfaz de programación de aplicaciones). En la programación orientada a objetos, las API ofrecen funciones y procedimientos para ser utilizados por otras aplicaciones.

JAVADOC

Actualmente, es el estándar para crear comentarios de clases desarrolladas en Java. La sintaxis utilizada para este tipo de comentarios es empezar por /** y finalizar con */, donde se incorporará el carácter * para cada línea, tal como se muestra a continuación:

En el siguiente ejemplo se efectúa una propuesta de los comentarios tipo Javadoc para la clase factorial, clase que calcula el factorial de un número.

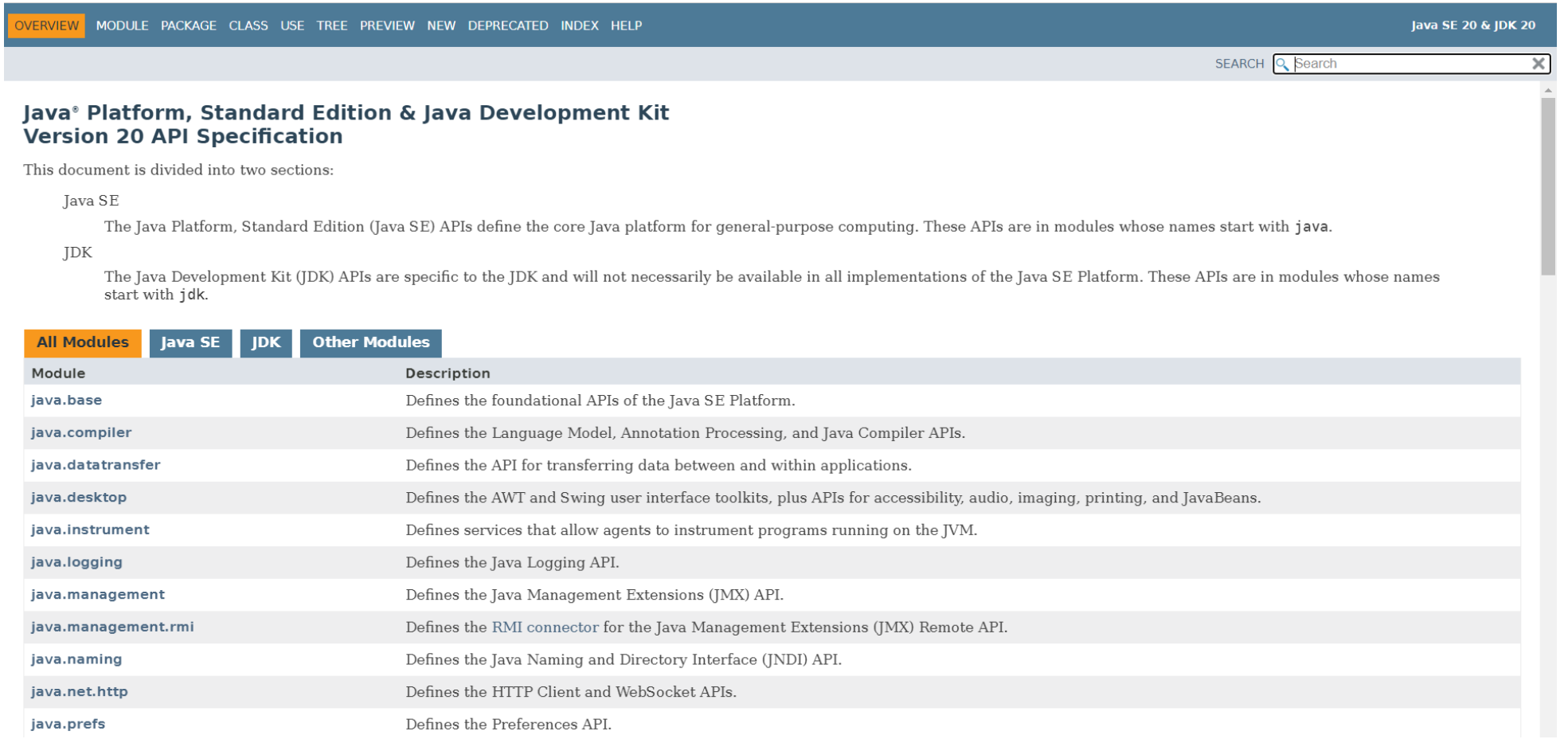
```
/ **
 * Clase que calcula el factorial de un número.
 * @Author IOC
 * @Version 2012
 * /
public class Factorial {

    / **
     * Calcula el factorial de n.
     * N! = N * (n-1) * (n-2) * (n-3) * ... * 1
     * @Param n es el número al que se calculará el factorial.
     * @Return n! es el resultado del factorial de n
     * /
    public static double factorial (double n) {

       if (n == 0)
            return 1;
       else
       {
            double resultado = n * factorial (n-1);
            return resultado;
       }
    }
}
```

JAVADOC

- La diferencia entre este tipo de comentarios y el resto de los comentarios es que los comentarios Javadoc sí tienen una estructura específica a seguir para ser escritos. En cambio, los comentarios internos dan completa libertad para ser implementados.
- Otra diferencia significativa es el objetivo de los comentarios. Los **comentarios internos** se hacen en todo el código fuente, mientras que los comentarios **Javadoc** están pensados para ser utilizados al principio de cada clase y de cada método.
- Finalmente, los comentarios Javadoc generan de forma automática la documentación técnica del software, en formato HTML.

JAVADOC: DOCUMENTACIÓN DE APIS

La documentación oficial de la API de Java ha sido generada también con Javadoc:



JAVADOC: DOCUMENTACIÓN DE APIS

La documentación Javadoc es una colección de páginas HTML de todas las clases, métodos, parámetros y retornos junto con la información y especificaciones que quiera incluir el desarrollador de la API que en el caso de las clases de JDK incluye abundantes e interesantes detalles de implementación a tener en cuenta al usar las clases.

JAVADOC: DOCUMENTACIÓN DE APIS

Se genera a partir del propio código fuente de las clases con los comentarios incluidos que siguen cierto formato precediendo la definición de las clases y métodos.

Al estar código y documentación en el propio archivo de código fuente, es más fácil mantener sincronizados el código y su documentación.

JAVADOC: DOCUMENTACIÓN DE APIS

La documentación en el código fuente se incluye en comentarios que preceden una clase o método, además, con anotaciones se pueden documentar los parámetros y el valor de retorno.

Se pueden incluir etiquetas HTML junto con algunas "**anotaciones Javadoc**".

Algunas anotaciones Javadoc incluidas en el JDK son las que veremos a continuación pero también se pueden desarrollar anotaciones propias (taglet/doclet) o personalizar los estilos de la documentación para cambiar el contenido, información incluida o adaptar los estilos según los colores de la organización.

JAVADOC: DOCUMENTACIÓN DE APIS

Los comentarios tipo Javadoc siguen unos estándares en su creación:

```
/**
* Texto con alguna descripción (posibilidad de usar HTML)
*
* Etiquetas javadoc (comienzan por @)
*
*/
```

JAVADOC: DOCUMENTACIÓN DE APIS

Hay dos tipos de **etiquetas:**

- En **bloque**
  - @tag texto
- En **línea**
  - {@tag texto}

```
/ **
 * Clase que calcula el factorial de un número.
 * @Author IOC
 * @Version 2012
 * /
public class Factorial {

    / **
     * Calcula el factorial de n.
     * N! = N * (n-1) * (n-2) * (n-3) * ... * 1
     * @Param n es el número al que se calculará el factorial.
     * @Return n! es el resultado del factorial de n
     * /
    public static double factorial (double n) {

       if (n == 0)
            return 1;
       else
       {
            double resultado = n * factorial (n-1);
            return resultado;
       }
    }
}
```

Una etiqueta en **bloque** debe aparecer al inicio de la línea de comentario. El texto asociado con una etiqueta de bloque puede aparecer en múltiples líneas hasta que comience otra etiqueta de bloque o se alcance el fin del comentario.

Una etiqueta en **línea** aparece en cualquier lugar donde haya texto de un comentario Javadoc. Ejemplo:

```
/**
 * An example of inline tag. It computes {@code n1 + n2}.
 */
```

JAVADOC: DOCUMENTACIÓN DE APIS

Las etiquetas de **Javadoc** van precedidas por @, estos son los mas usados:

ETIQUETA	DESCRIPCIÓN
@author	Autor de la clase. Solo para las clases.
@version	Versión de la clase. Solo para clases.
@see	Referencia a otra clase, ya sea del API, del mismo proyecto o de otro. Por ejemplo: @see cadena @see paquete.clase#miembro @see enlace
@param	Descripción parámetro. Una etiqueta por cada parámetro.
@return	Descripción de lo que devuelve. Solo si no es void. Podrá describir valores de retorno especiales según las condiciones que se den, dependiendo del tipo de dato
@throws	Descripción de la excepción que puede propagar. Habrá una etiqueta throws por cada tipo de excepción.
@deprecated	Marca el método como obsoleto. Solo se mantiene por compatibilidad.
@since	Indica el nº de versión desde la que existe el método.

JAVADOC: DOCUMENTACIÓN DE APIS

Documentación de clases e interfaces

@author	Nombre del autor de la clase o interfaz.
@version	Versión de la clase y fecha

```
/ **
 * Clase que calcula el factorial de un número.
 * @Author IOC
 * @Version 2012
 * /
public class Factorial {
```

JAVADOC: DOCUMENTACIÓN DE APIS

Documentación de constructores y métodos

@param	nombre del parámetro	descripción de su significado y uso
@return		descripción de lo que se devuelve
@exception	nombre de la excepción	excepciones que pueden lanzarse
@throws	nombre de la excepción	excepciones que pueden lanzarse
@deprecated		Indica que el uso del elemento está desaconsejado

JAVADOC: DOCUMENTACIÓN DE APIS

@param nombre descripción

Ejemplo de uso:

```
/**
 * Removes from this List all of the elements whose index is between
 * fromIndex, inclusive and toIndex, exclusive.  Shifts any succeeding
 * elements to the left (reduces their index).
 * This call shortens the ArrayList by (toIndex - fromIndex) elements.  (If
 * toIndex==fromIndex, this operation has no effect.)
 *
 * @param fromIndex index of first element to be removed
 * @param toIndex index after last element to be removed
 */
protected void removeRange(int fromIndex, int toIndex) {
    ...
}
```

JAVADOC: DOCUMENTACIÓN DE APIS

Documentación de constructores y métodos

@return descripción

Ejemplo de uso:

```
/**
 * Tests if this vector has no components.
 *
 * @return  <code>true</code> if and only if this vector has
 *          no components, that is, its size is zero;
 *          <code>false</code> otherwise.
 */
public boolean isEmpty() {
    return elementCount == 0;
}
```

JAVADOC: DOCUMENTACIÓN DE APIS

@throws tipo descripción

- Aplicable a constructores y métodos.
- Describe las posibles excepciones del constructor/método.
- Se usa un @throws por cada posible excepción.
- Si es de ayuda para el usuario, también se pueden documentar las unchecked exceptions.

```
/**
 * Parses the string argument as a signed decimal
 * <code>long</code>.  The characters in the string must all be
 * decimal digits, except that...
 *
 * @param      s a <code>String</code> containing the <code>long</code>
 *             representation to be parsed
 * @return     the <code>long</code> represented by the argument in
 *             decimal.
 * @exception  NumberFormatException  if the string does not contain a
 *             parsable <code>long</code>.
 */
public static long parseLong(String s)
    throws NumberFormatException {
    ....
```

@throws: a partir de Javadoc 1.2 **es un sinónimo de @exception.**

(@exception es por tanto la anotación original, en cualquier caso, mejor usar @throws ya que es la anotación usada por defecto actualmente en los principales IDEs)

JAVADOC: DOCUMENTACIÓN DE APIS

@see referencia

- Aplicable a clases, interfaces, constructores, métodos, atributos y paquetes
- Añade enlaces de referencia a otras partes de la documentación
- Variantes:
  - @see Clase
  - @see #Constructor ()
  - @see #metodo ()
  - @see string
  - @see label
  - @see paquete.Clase

```
package com.jdojo.utility;

/**
 * A dummy class.
 *
 * @see "Online Java Tutorial"
 * @see <a href="http://www.oracle.com">Oracle Website</a>
 * @see com.jdojo.utility.Calc Calculator
 * @see com.jdojo.utility.Calc#add(int, int) Add Method
 * @see com.jdojo.utility.Calc#multiply(int, int)
 */
public class Dummy {
}
```

JAVADOC: DOCUMENTACIÓN DE APIS

@deprecated descripción

```
/**
 * @deprecated explanation of why it was deprecated
 */
@Deprecated
public void deprecatedMethod() {
    ...
}
```

JAVADOC: DOCUMENTACIÓN DE APIS

- **{@inheritDoc}**: Hereda el comentario Javadoc de la clase o método superior en la jerarquía de clases.
- De forma implícita (automática), la herramienta de generación de la documentación toma la descripción o el tag de la clase o interfaz de nivel superior.

```
/**
 * (superclase) ...
 *
 * @throws NullPointerException if <code>dst</code> is <code>null</code>
 */
public void getChars(int srcBegin, int srcEnd, char dst[], int dstBegin) {
    ....
}
```

```
/**
 * (subclase) ...
 *
 * @throws NullPointerException {@inheritDoc}
 */
public void getChars(int srcBegin, int srcEnd, char dst[], int dstBegin) {
```

JAVADOC: DOCUMENTACIÓN DE APIS

Reglas elementales

- La primera frase de cada comentario Javadoc debe ser una frase resumen con una **descripción concisa y completa**, terminada en punto, y seguida de un espacio, tabulador o retorno de carro
- Usar la **etiqueta** <code> para palabras clave y nombres
- Preferible el uso de la **tercera persona**

```
* Devuelve el índice del primer elemento...
* Devolvemos el índice del primer elemento... ⇐ Evitar
```

- Empezar con un **verbo** la descripción de los métodos
- **Omitir el sujeto** cuando es obvio

```
* @param peer nombre del peer
* @param peer parámetro con el nombre del peer ⇐ Evitar
```

JAVADOC: DOCUMENTACIÓN DE APIS

La etiqueta HTML <code> te permite darle un formato especifico al texto como si fuera código de ordenador.

Esto ofrece un determinado tamaño y una distancia especifica determinada para el código del ordenador.

HTML

```
<code>Este texto fue formateado con la etiqueta code.</code>
```

Demo

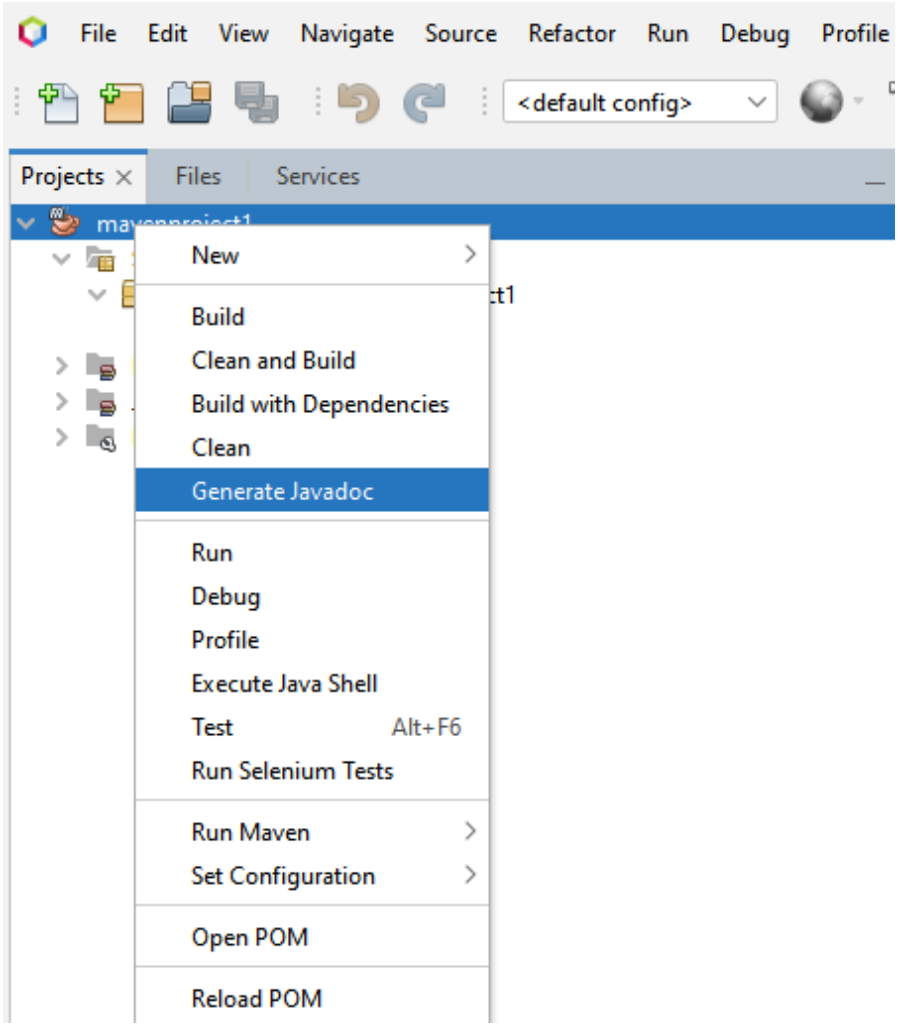
`Este texto fue formateado con la etiqueta code.`

```
/**
 * A utility class to perform basic calculations on numbers. All methods in this class are
 * <code>static</code>. It provides methods to perform addition, subtraction, multiplication,
 * and division.
 *
 * @author Kishori Sharan
 * @since Version 1.0
 */
public final class Calc {
    // More code goes here
}
```

UTILIZACIÓN DE JAVADOC CON NETBEANS

La mayoría de IDEs que permiten programar en Java ofrecen funcionalidades para poder crear documentación a partir de Javadoc. NetBeans no es una excepción y también ofrece esta posibilidad.

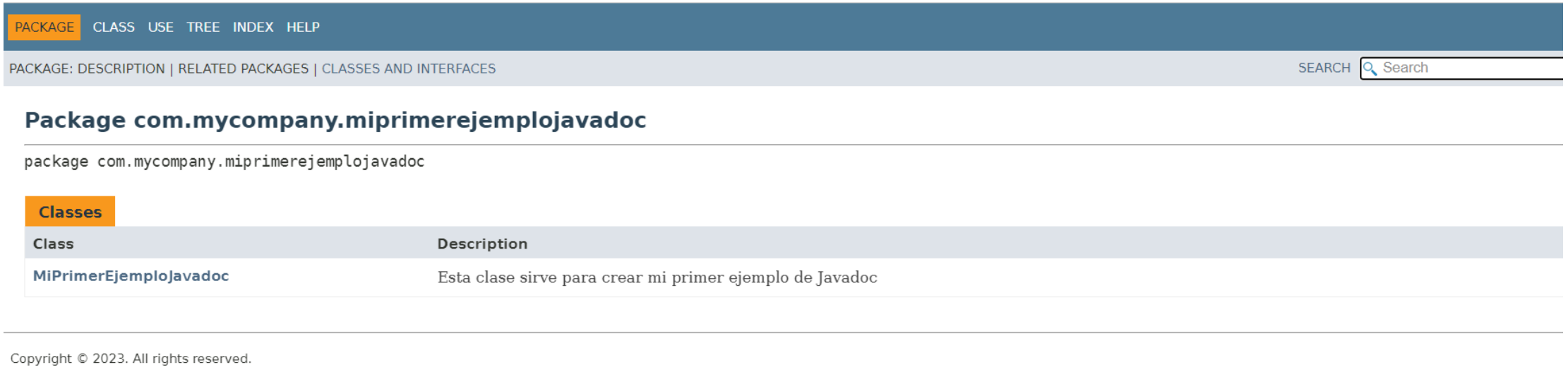
Una vez desarrollado un proyecto en Java y añadidos los comentarios con las estructuras y estándares Javadoc, se podrá generar la documentación de forma automática utilizando la funcionalidad "*Generate Javadoc*" haciendo clic con el botón derecho en el nombre del Proyecto.



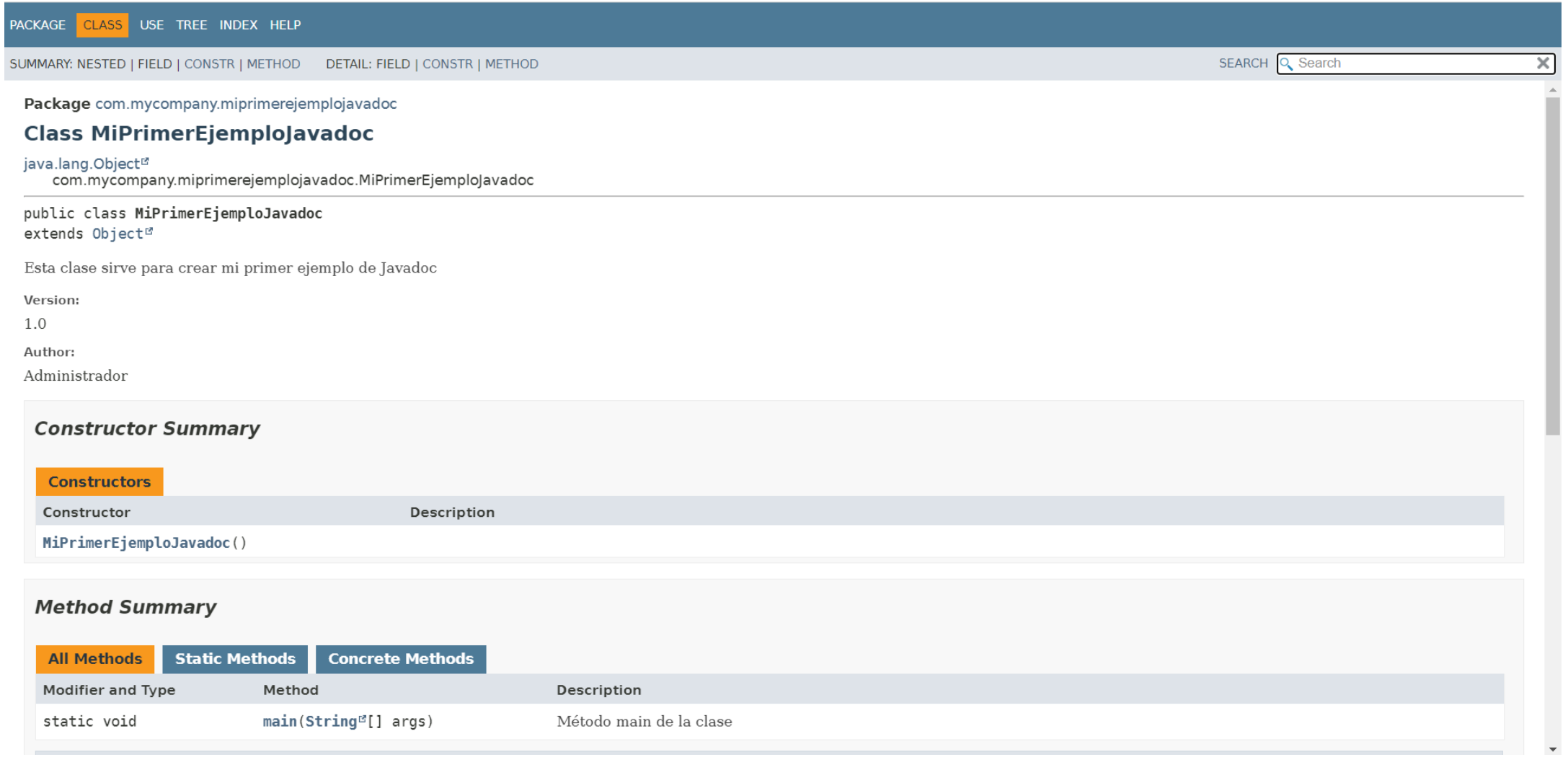
API CLASE FACTORIAL CON JAVADOC

Una vez que NetBeans termine de generar los ficheros, y en el caso de que no haya errores, tendremos que ir a la ruta:

NombreProyecto\target\site\apidocs y abrir el fichero “index.html”



API CLASE FACTORIAL CON JAVADOC



PACKAGE CLASS USE TREE INDEX HELP

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

Package com.mycompany.miprimerejemplojavadoc

Class MiPrimerEjemploJavadoc

java.lang.Object
com.mycompany.miprimerejemplojavadoc.MiPrimerEjemploJavadoc

```
public class MiPrimerEjemploJavadoc
extends Object
```

Esta clase sirve para crear mi primer ejemplo de Javadoc

Version:

1.0

Author:

Administrador

Constructor Summary

Constructors

Constructor	Description
MiPrimerEjemploJavadoc()	

Method Summary

All Methods **Static Methods** **Concrete Methods**

Modifier and Type	Method	Description
static void	main(String[] args)	Método main de la clase

PROGRAMAR Y DOCUMENTAR EN **INGLÉS**

Razones:

- **No es un capricho,** la sintaxis del lenguaje de programación está en inglés así que nuestros programas se estarán acercando más a la naturaleza del lenguaje.
- Es habitual que ciertas empresas te obliguen a hacerlo.
- En cuanto te habitúes a hacerlo verás que la mayor parte de código que encuentres disponible está en inglés.
- Tú mismo querrás compartir tu código en repositorios como GitHub o Bitbucket y si están escritos en inglés tendrás mayor aceptación y mayor número de descargas.