DOCUMENTACIÓN DE APLICACIONES

Introducción

- Documentar el código de un programa es añadir suficiente información como para explicar lo que hace, punto por punto, de forma que no sólo los ordenadores sepan qué hacer, sino que además los humanos entiendan qué están haciendo y por qué
  - Documentar un programa no es sólo un acto de buen hacer del programador por aquello de dejar la obra rematada
  - Es además una necesidad que sólo se aprecia en su debida magnitud cuando hay errores que reparar o hay que extender el programa con nuevas capacidades o adaptarlo a un nuevo escenario
  - Hay dos reglas que no se deben olvidar nunca:
    - todos los programas tienen errores y descubrirlos sólo es cuestión de tiempo y de que el programa tenga éxito y se utilice frecuentemente
    - todos los programas sufren modificaciones a lo largo de su vida, al menos todos aquellos que tienen éxito
- Por una u otra razón, todo programa que tenga éxito será modificado en el futuro, bien por el programador original, bien por otro programador que le sustituya. Pensando en esta revisión de código es por lo que es importante que el programa se entienda: para poder repararlo y modificarlo

¿Qué documentar?

- **Hay que añadir explicaciones a todo lo que no es evidente.**
- **No hay que repetir lo que se hace, sino explicar por qué se hace.**

- ¿de qué se encarga una clase? ¿un paquete?
- ¿qué hace un método? ¿cuál es su uso esperado?
- ¿para qué se usa una variable? ¿cuál es su uso esperado?
- Algoritmo utilizado, fuente, limitaciones, mejoras….

Tipos de comentarios en Java

- **Utilizados para la documentación externa (Javadoc)**
  - Comienzan con /** y acaban con */ pueden ocupar varias líneas que normalmente empiezan con un *
- **Comentarios internos** para clarificar el código y que no se verán en la documentación externa:
  - De una sola línea
    - Comienzan por //
  - De varias líneas
    - Comienzan por /* y acaban con */ pueden ocupar varias líneas
    - Habitualmente se utilizan para "ocultar" parte del código que no se desea ejecutar pero tampoco eliminarlo (Ej.: para hacer pruebas)

¿Cuándo se pone un comentario?

- Por obligación (**javadoc**):
  - al principio de cada clase
  - al principio de cada método
  - ante cada variable de clase

- Por conveniencia (**una línea**):
  - al principio de fragmento de código no evidente
  - a lo largo de los bucles

Javadoc

El paquete de desarrollo Java incluye la herramienta **javadoc** para generar un conjunto de páginas web a partir de los ficheros de código

Esta herramienta toma en consideración algunos comentarios para generar una documentación bien presentada de clases y componentes de clases (variables y métodos)

Javadoc

- Javadoc no se centra en los detalles del código, sino en la arquitectura de la aplicación:
  - módulos (clases) y componentes de estos (atributos y métodos)
- Javadoc realza algunos comentarios, de los que exige una sintaxis especial:
  - Deben comenzar por "/**" y terminar por "*/", incluyendo una descripción y algunas etiquetas especiales
- Estos comentarios especiales deben aparecer **justo antes de la declaración de una clase, un campo o un método** en el mismo código fuente
- En los comentarios **se puede incluir código html** que será interpretado como tal a la hora de generarse la documentación

Documentación de clases e interfaces

@author	**nombre del autor**
@version	**identificación de la versión y fecha**
@see	referencia a otras clases y métodos

Documentación de constructores y métodos

@param	una por argumento de entrada	nombre del parámetro	descripción de su significado y uso
@return	si el método no es *void*		descripción de lo que se devuelve
@exception	una por tipo de *Exception* que se puede lanzar	nombre de la excepción	excepciones que pueden lanzarse
@throws		nombre de la excepción	excepciones que pueden lanzarse

@exception y @throws se pueden usar indistintamente.

Ejemplo:

```
1 /**
2  * Ejemplo: círculos.
3  *
4  * @author XXXXX
5  * @version 17-04-2017
6  */
7 public class Circulo {
8     private double centroX;
9     private double centroY;
10     private double radio;


12     /**
13      * Constructor.
14      * @param x centro: coordenada X.
15      * @param y centro: coordenada Y.
16      * @param r radio.
17      */
18     public Circulo(double x, double y,
double r) {
19         centroX = x;
20         centroY = y;
21         radio = r;
22     }
24     /**
25      * Getter.
26      * @return centro: coordenada X.
27      */
28     public double getCentroX() {
29         return centroX;
30     }
48     /**
49      * Calcula la longitud de la circunferencia
(perímetro del círculo).
50      * @return perímetro.
51      */
52     public double getPerimetro() {
53         return 2 * Math.PI * radio;
54     }
64     /**
65      * Desplaza el círculo a otro lugar.
66      * @param despX movimiento en el eje X.
67      * @param despY movimiento en el eje Y.
68      */
69     public void mueve(double despX, double despY) {
70         centroX = centroX + despX;
71         centroY = centroY + despY;
72     }
81 }
```

Documentación

- **Documentar todas las clases** (ej. líneas 1-6), indicando:
  1. Qué hace la clase (ej. línea 2)
  2. El autor (ej. línea 4)
  3. La versión del programa, señalada (por ejemplo) por su fecha (ej. línea 5)

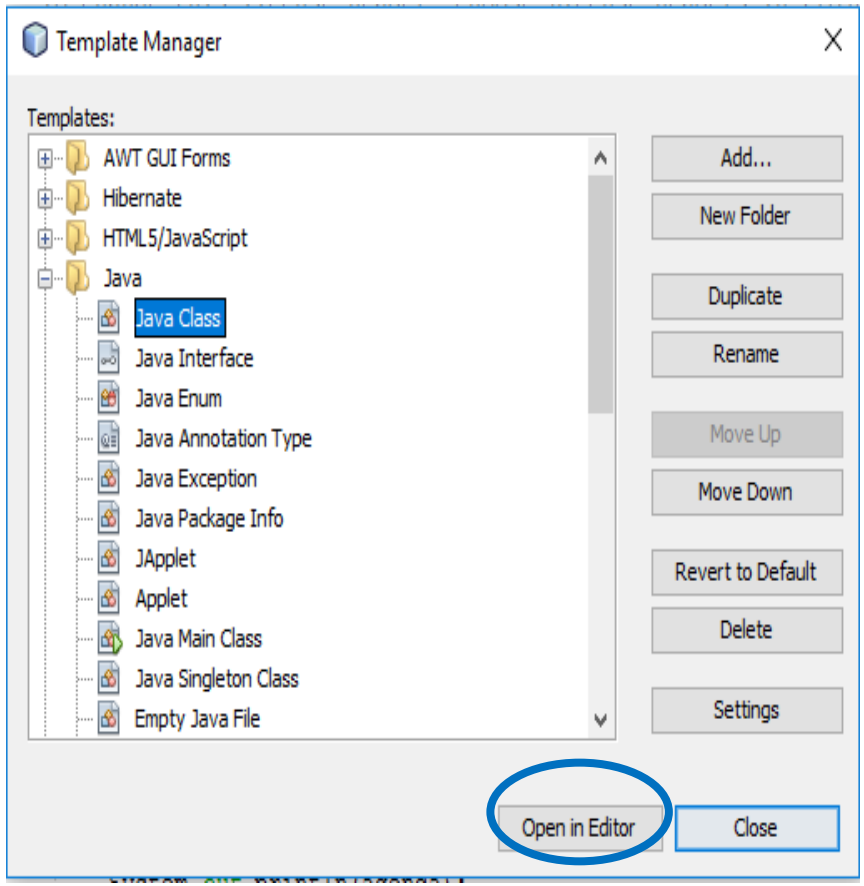
- **Documentar todos y cada uno de los métodos** (ej. líneas 12-17, 24-27, 48-51, 64-68), indicando:
  1. Qué hace el método (ej. línea 13, 25, 49, 65)
  2. Los parámetros de entrada (ej. líneas 14-16, 66-67). Para cada parámetro hay que escribir:
     a. @param
     b. el nombre del parámetro
     c. una pequeña descripción de lo que se espera en el parámetro

     OJO: un método puede carecer de parámetros de entrada (ej. líneas 28 y 52)
  3. El resultado que devuelve (ej. línea 26)
     a. @return
     b. una somera descripción de lo que devuelve

     OJO: un método puede no devolver nada (ej. líneas 18 y 78)

Preparando Netbeans

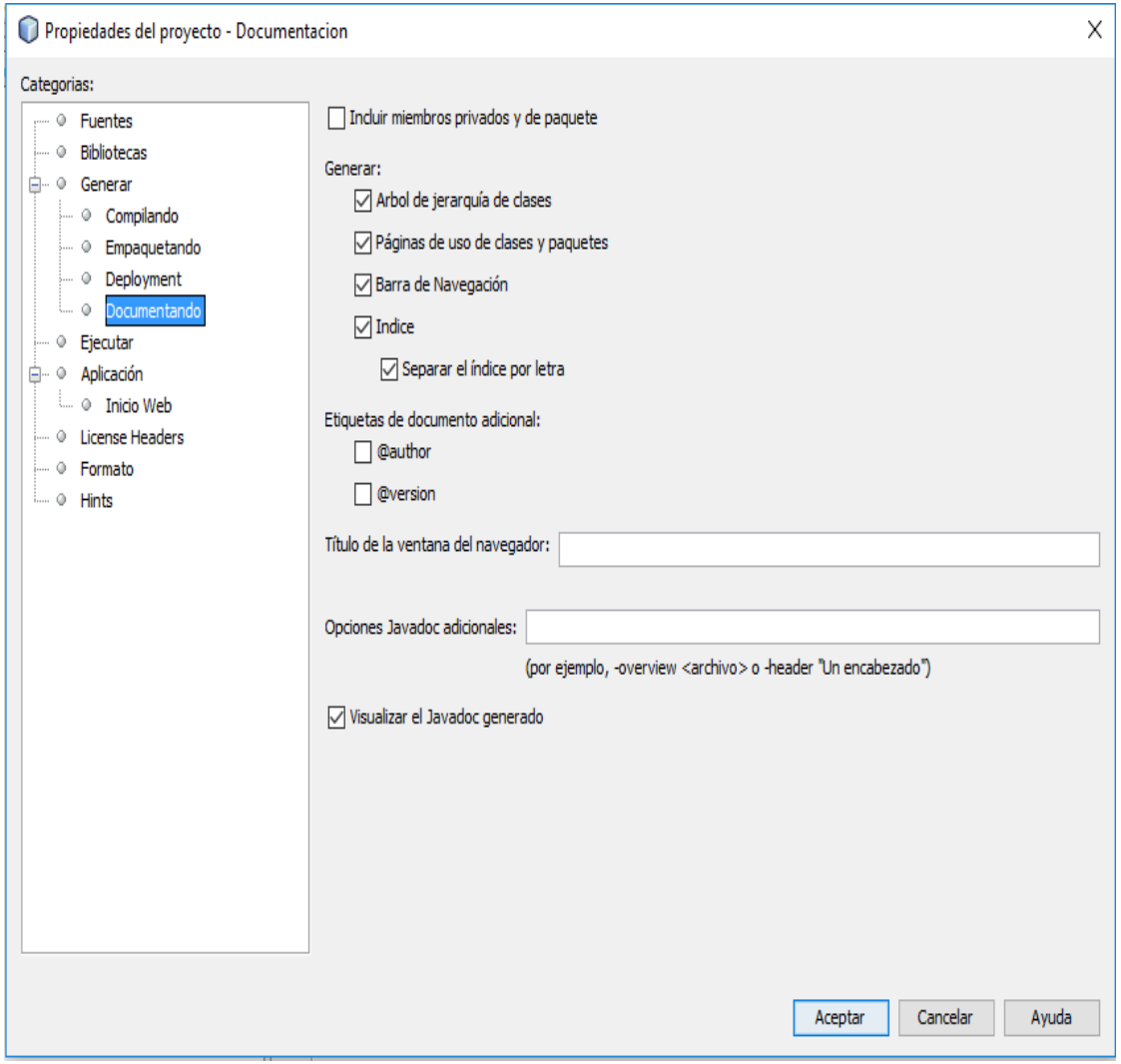
- Si modificamos la plantilla de las clases de java, podemos preparar parte de la documentación para que aparezca por defecto
- Ej.: el autor y la versión



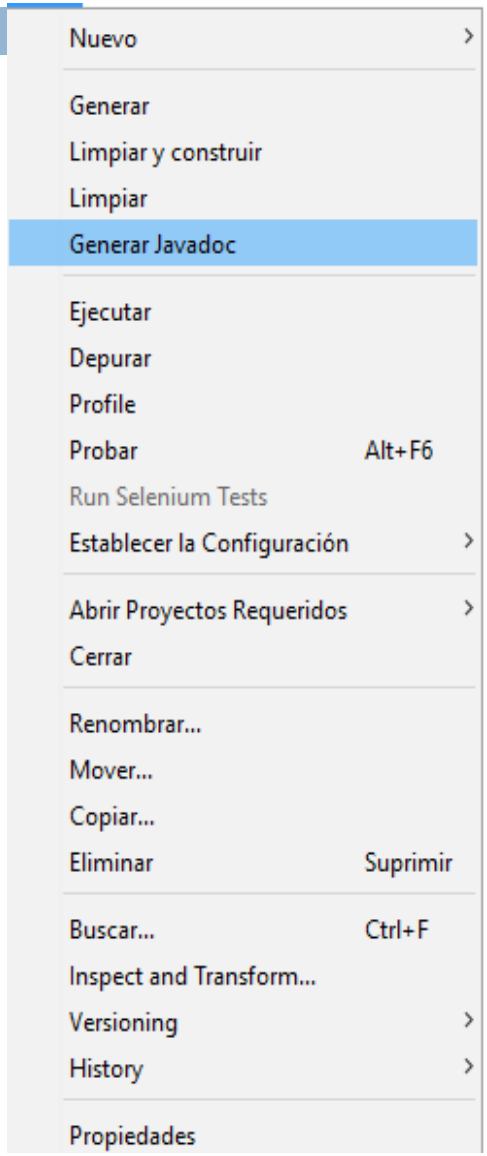
Configurar la documentación

Botón derecho sobre el proyecto

> Propiedades



Creando la documentación



Botón derecho sobre el proyecto

- Generar Javadoc

Se genera en la carpeta del proyecto \dist\javadoc