Anexo II JAVADOC

1. Introducción

Documentar el código de un programa es añadir suficiente información para explicar lo que hace, punto por punto, para que cualquiera que lo tenga que leerlo pueda entender lo que hará el programa y por qué.

No siempre lo que se explica que un programa tiene que hacer y lo que hace coincide.

Es una necesidad que sólo se aprecia en su debida magnitud cuando hay errores que reparar o nuevas funcionalidades que extender al programa. Hay dos reglas que no se deben olvidar nunca:

- todos los programas tienen errores y descubrirlos sólo es cuestión de tiempo y de la frecuencia de su uso
- todos los programas sufren modificaciones a lo largo de su vida por adaptarse a cambios de contexto tecnológico o cambios en su funcionalidad

Por una u otra razón, todo programa será modificado en el futuro, bien por el programador original, bien por otro programador que le sustituya. Pensando en la revisión de código es por lo que es importante que el programa se entienda, para repararlo y modificarlo.

2. ¿Qué hay que documentar?

a) Hay que añadir explicaciones a todo lo que no es evidente.

b) No hay que repetir lo que se hace, sino explicar por qué se hace.

Y eso se traduce en:

- ¿de qué se encarga una clase? ¿un paquete?
- ¿qué hace un método? ¿cuál es el uso esperado de un método?
- ¿para qué se usa una variable? ¿cuál es el uso esperado de una variable?
- ¿qué algoritmo estamos usando? ¿de dónde lo hemos sacado?
- ¿qué limitaciones tiene el algoritmo? ¿... la implementación?
- ¿qué se debería mejorar ... si hubiera tiempo?

3. Tipos de comentarios

En Java disponemos de tres notaciones para introducir comentarios:

I. **javadoc**

Comienzan con los caracteres "/**", se pueden prolongar a lo largo de varias líneas (que probablemente comiencen con el carácter "*") y terminan con los caracteres "*/".

II. **una línea**

Comienzan con los caracteres "//" y terminan con la línea

III. **tipo C**

Comienzan con los caracteres "/*", se pueden prolongar a lo largo de varias líneas (que probablemente comiencen con el carácter "*") y terminan con los caracteres "*/".

Cada tipo de comentario se debe adaptar a un propósito:

I. **javadoc**

Para generar documentación externa (ver comentarios javadoc más abajo).

II. **una línea**

Para documentar código que no necesitamos que aparezca en la documentación externa (que genere javadoc).

Este tipo de comentarios se usará incluso cuando el comentario ocupe varias líneas, cada una de las cuales comenzará con "//".

III. **tipo C**

Para eliminar código. Ocurre a menudo que código obsoleto no queremos que desaparezca, sino mantenerlo "por si acaso". Para que no se ejecute, se comenta. (En inglés se suele denominar *"commentout"*). De todas maneras, este código debería eliminarse y no mantenerlo "por si acaso".

4. ¿Cuándo hay que poner un comentario?

Por obligación (**javadoc**):

1. al principio de cada clase
2. al principio de cada método
3. ante cada variable de clase (similares a las constantes)

Por conveniencia (**una línea**):

4. al principio de fragmento de código no evidente
5. a lo largo de los bucles

Y por si acaso (**una línea**):

6. siempre que hagamos algo raro
7. siempre que el código no sea evidente

Es decir, es mejor que los comentarios sobren que falten.

Y una nota de cautela, cuando un programa se modifica, los comentarios deben modificarse al tiempo, no sea que los comentarios acaben refiriéndose a un algoritmo que ya no utilizamos.

5. Javadoc: documentación de APIs

El paquete de desarrollo Java incluye una 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).

Aunque javadoc no ayuda a la comprensión de los detalles de código, si ayuda a la comprensión de la arquitectura de la solución, lo que no es poco. Se dice que javadoc se centra en la interfaz *(API – Application Programming Interface)* de las clases y paquetes Java.

Javadoc realza algunos comentarios, de los que exige una sintaxis especial. Deben comenzar por "/**" y terminar por "*/", incluyendo una descripción y algunas etiquetas especiales:

```
/**
   * Parte descriptiva.
   * Que puede consistir de varias frases o párrafos.
   *
   * @etiqueta texto específico de la etiqueta
   */
```

Estos comentarios especiales deben aparecer **justo antes de la declaración de una clase, un campo o un método** en el mismo código fuente. En las siguientes secciones se detallan las etiquetas (*tags*) que javadoc sabe interpretar en cada uno de los casos.

Como regla general, hay que destacar que la primera frase (el texto hasta el primer punto) recibirá un tratamiento destacado, por lo que debe aportar una explicación concisa y contundente del elemento documentado. Las demás frases entrarán en detalles.

5.1. Documentación de clases e interfaces

Deben usarse al menos las etiquetas:

- @author
- @version

La tabla muestra todas las etiquetas posibles y su interpretación:

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

5.2. Documentación de constructores y métodos

Deben usarse al menos las etiquetas:

- **@param**
  una por argumento de entrada
- **@return**
  si el método no es *void*
- **@exception @throws**
  una por tipo de *Exception* que se puede lanzar
  (@exception y @throws se pueden usar indistintamente).

La tabla muestra todas las etiquetas posibles y su interpretación:

@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

@exception y @throws se pueden usar indistintamente.

5.3. Ejecución de javadoc

La mayor parte de los entornos de desarrollo incluyen un botón para llamar a javadoc así como opciones de configuración, aunque también se puede generar desde la consola.

...> {directorio de instalación}/javadoc *.java

La herramienta javadoc admite miles de opciones.

usage: javadoc [options] [packagenames] [sourcefiles] [classnames] [@files]

javadoc -help despliega un catálogo completo de opciones.

6. Ejemplo

http://jungla.dit.upm.es/~pepe/doc/fprg/javadoc.htm

Todos los programas deben venir adecuadamente documentados.

A continuación se muestra un programa documentado, que se comenta más abajo:

```
public class Circulo {
private double centroX;
    private double centroY;
    private double radio;




public Circulo(double cx, double cy,
       double r) {
centroX = cx;
centroY = cy;
       radio = r;
    }
```

```
1 /**
2  * Ejemplo: círculos.
  3  *
  4  * @author José A. Mañas - fprg5000
5  * @version 23.9.2005
  6  */
  7 public class Circulo {
  8     private double centroX;
  9     private double centroY;
 10     private double radio;

12  /**
 13  * Constructor.
 14  * @param cx centro: coordenada X.
 15  * @paramcy centro: coordenada Y.
16  * @param r radio.
 17  */
 18  public Circulo(double cx, double cy,
double r) {
19         centroX = cx;
20         centroY = cy;
21         radio = r;
22     }

24     /**
25      * Getter.
26      * @return centro: coordenada X.
27      */
```

```
public double getCentroX() {
 return centroX;
}

public doublegetCircunferencia(){
      return 2 * Math.PI * radio;
}

public void mueve(double deltaX,
    double deltaY) {
      centroX = centroX + deltaX;
      centroY = centroY + deltaY;
   }

public void escala(double s) {
      radio = radio * s;
   }
}
```

```
28     public double getCentroX() {
29         returncentroX;
30     }

31     /**
32      * Calcula la longitud de la
33      * circunferencia (perímetro del
34      * círculo).
35      * @return circunferencia.
36      */
37    public double getCircunferencia() {
38         return 2 * Math.PI * radio;
39     }

40     /**
41      * Desplaza el círculo a otro
42      * lugar.
43      * @paramdeltaX movimiento en el
44      * eje X.
45      * @paramdeltaY movimiento en el
46      * eje Y.
47      */
48     public void mueve(double deltaX,
49     double deltaY) {
50centroX = centroX + deltaX;
51         centroY = centroY + deltaY;
52     }

53     /**
54      * Escala el círculo (cambia su
55      * radio).
56      * @param s factor de escala.
57      */
58     public void escala(double s) {
59         radio = radio * s;
60     }
61 }
```

Hay que ...

- Documentar todas las clases (ej. líneas 1-6), indicando:
  - lo que hace la clase (ej. línea 2)
  - el autor (ej. línea 4)
  - 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, 31-36, 40-47 y 53-57),indicando:
  - lo que hace el método (ej. línea 13, 25, 32-34, 41-42, 54-55)
  - los parámetros de entrada (ej. líneas 14-16, 43-46 y 56).

    Para cada parámetro hay que escribir
    1. @param
    2. el nombre del parámetro
    3. una somera descripción de lo que se espera en el parámetro

OJO: un método puede carecer de parámetros de entrada (ej. líneas 28 y 37)

- el resultado que devuelve(ej. línea 26)
    1. @return
    2. una somera descripción de lo que devuelve

OJO: un método puede no devolver nada (ej. líneas 48 y 58)

7. Javadoc y Eclipse

En Ventana→ Preferencias → Java → Compilador →Javadoc están las opciones de configuración.

Una vez que tenemos el código con los comentarios:

Proyecto →export-> Java->Javadoc.

Si no está asociado tendremos que asociar javadoc.exe, que está en el jdk…bin

Generará las páginas html y además si en el código nos ponemos encima de la clase en un tooltip aparecerá la documentación.