T7-Documentación de programas en Java: JavaDoc

1. Documentación:

Como indican diferentes autores:

"Si su programa no merece la pena documentarlo, probablemente no merece la pena ejecutarlo"
J. Nagler. 1995 Coding Style and Good Computing Practices.

"No documentar código malo, reescribirlo"
R. Caron. 2000 Coding Techniques and Programming Practices.

"Escribe la documentación antes de escribir el código."

S.W. Ambler. 2000.Writing Robust Java Code.

- El proceso de documentación de código, es uno de los aspectos más importantes de la labor de un programador. Documentar el código nos sirve para explicar su funcionamiento punto por punto, de forma que cualquier persona que lea el comentario, puede entender la finalidad del código.
- La labor de documentación es fundamental para la detección de errores y para su mantenimiento posterior, que en muchos casos, es realizado por personas diferentes a las que intervinieron en su creación. Hay que tener en cuenta que todos los programas tienen errores y todos los programas sufren modificaciones a lo largo de su vida.
- La documentación añade explicaciones de la función del código, de las características de un método, etc. Debe tratar de explicar todo lo que no resulta evidente. Su objetivo no es repetir lo que hace el código, sino explicar por qué se hace.
- La documentación explicará cuál es la finalidad de una clase, de un paquete, qué hace un método, para qué sirve una variable, qué se espera del uso de una variable, qué algoritmo se usa, por qué hemos implementado de una manera y no de otra, qué se podría mejorar en el futuro, etc.

2. ¿Qúe documentar en nuestro código?

- 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, lo que se puede resumir 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:

• **Comentario de una línea**: Comienzan con los caracteres "//" y terminan con el fin de la línea. Se utiliza 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 "//".

• **Comentarios tipo leguaje de programación 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 "*/". También se utiliza para eliminar código. Ocurre a menudo que el código obsoleto no queremos que desaparezca, sino mantenerlo "por si acaso". Para que no se ejecute, se comenta (en inglés se suele denominar *"comment out"*).

• **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 "*/". Se usa para generar documentación externa (ver comentarios javadoc más abajo).

4. Herramientas de documentación:

En la actualidad, el desarrollo rápido de aplicaciones, en muchos casos, va en detrimento de una buena documentación del código. Si el código no está documentado, puede resultar bastante difícil de entender, y por tanto de solucionar errores y de mantenerlo. La primera alternativa que surge para documentar código, son los comentarios. Con los comentarios, documentamos la funcionalidad de una línea de código, de un método o el comportamiento de una determinada clase.

Existen diferentes herramientas que permiten automatizar, completar y enriquecer nuestra documentación. Podemos citar JavaDoc, SchemeSpy y Doxygen, que producen una documentación actualizada, precisa y utilizable en línea, incluyendo además, con SchemeSpy y Doxygen, modelos de bases de datos gráficos y diagramas.

Insertando comentario en el código más difícil de entender, y utilizando la documentación generada por alguna de las herramientas citadas anteriormente, se genera la suficiente información para ayudar a cualquier nuevo programador o a nosotros mismos en un futuro.

5. JavaDoc:

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, es decir, en sus métodos y propiedades y las entradas y salidas de las mismas.

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

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

6. Documentación de Clases:

Deben usarse al menos las etiquetas:

- @author: Autor de la clase.
- @version: Versión de la clase.

Puede contener también:

- @see: Si hace referencia a otra clase.
- @since: Fecha desde la que está presente la clase.

7. Documentación de constructores y métodos:

Deben usarse al menos las siguientes etiquetas siempre que existan los elementos en el método:

- @param: Una por argumento de entrada. Parámetros de entrada.
- @return: Si el método no es void indica que devuelve.
- @exception ó @throws: Una por tipo de Exception que se puede lanzar(@exception y @throws se pueden usar indistintamente).

Ejemplo de método obtenido en la wikipedia

```
/**
* Inserta un título en la clase descripción.
* Al ser el título obligatorio, si es nulo o vacío se lanzará
* una excepción.
*
* @param titulo El nuevo título de la descripción.
* @throws IllegalArgumentException Si titulo es null, está vacío
o contiene sólo espacios.
*/
 public void setTitulo (String titulo) throws IllegalArgumentException
 {
    if (titulo == null || titulo.trim().equals(""))
    {
        throw new IllegalArgumentException("El título no puede ser nulo
o vacío."); }
```

```
  else
  {
      this.titulo = titulo;
  } }
```

8. <u>Documentación de campos o atributos:</u>

No es obligatorio documentarlos. Ejemplo completo:

```
/**
* <h1>Ejemplo: círculos</h1>.
* @see dibujo.Figuras
* @author Juan Rodríguez
* @version 12.12.2014
* @since 12.9.2016
*/
      public class Circulo { private double centroX; private double
      centroY; private double radio;
/**
* Constructor.
* @param cx centro: coordenada X.
* @param cy centro: coordenada Y.
* @param r radio.
*/

          public Circulo(double cx, double cy, double r) {
              centroX = cx; centroY = cy; radio = r;
          }
/**
* obtención de X.
* @return centroX: coordenada X.
*/
            public double getCentroX()
                { return centroX; }
/**
* Calcula la longitud de la circunferencia (perímetro del círculo).
* @return circunferencia.
*/
            public double getCircunferencia() {
                return 2 * Math.PI * radio; }
/**
* Desplaza el círculo a otro lugar.
* @param deltaX movimiento en el eje X.
* @param deltaY movimiento en el eje Y.
*/

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

```
/**
* Escala el círculo (cambia su radio).
* @param s factor de escala.
*/
    public void escala(double s) { radio = radio * s; } }
```

9. Generar la documentación con JavaDoc:

Para cada proyecto, podemos generar un conjunto de páginas HTML que describen las clases del proyecto, interfaces, métodos, constructores. La documentación se construye a partir de la estructura del código y los comentarios agregados a este.

Para generar la documentación para un proyecto en NetBeans:

1 Seleccionamos el nombre del proyecto para el cual queremos generar la documentación, en la ventana “Proyectos”.

2 Elegimos el menú “Run” -“Generate Javadoc para el proyecto”.