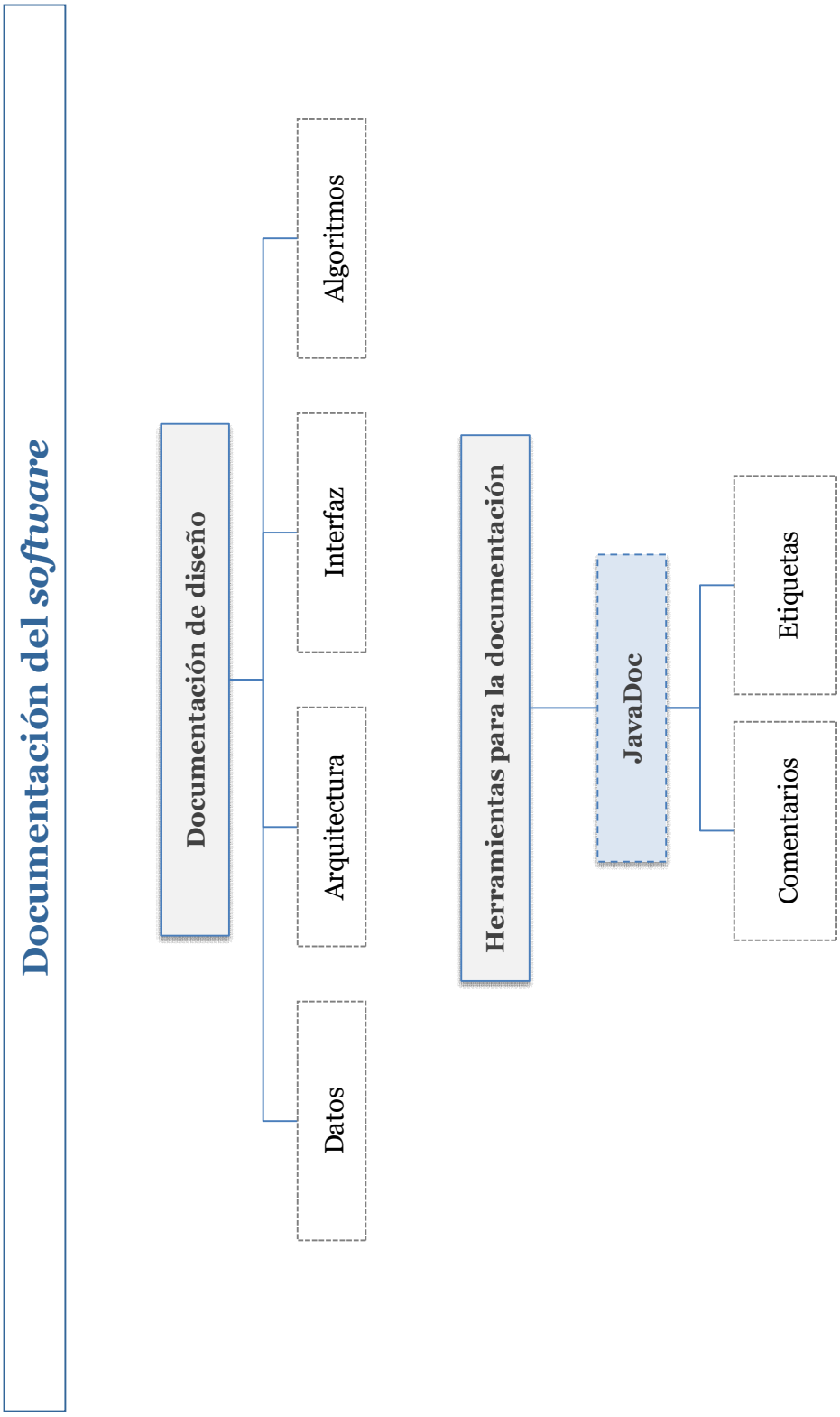
Documentación de *software*

TEMA 11

Esquema



 © Universidad Internacional de La Rioja (UNIR)

Ideas clave

11.1. ¿Cómo estudiar este tema?

Para estudiar este tema debes leer el **capítulo 13 (páginas 219-223, 226-234)** del siguiente libro, disponible en el aula virtual bajo licencia CEDRO:

Pressman, R. S. (2002). *Ingeniería del software. Un enfoque práctico*. Madrid: McGraw Hill.

Para comprobar si has comprendido los conceptos puedes realizar el test de autoevaluación del tema.

En este tema vamos a ver las **etapas de diseño del *software*** y vamos a aprender a utilizar la **herramienta JavaDoc** para generar documentación del código fuente Java.

11.2. ¿Por qué es importante documentar el *software*?

Documentación del *software*

Es todo lo que concierne a la **documentación del propio desarrollo del *software* y de la gestión del proyecto**, pasando por modelaciones (UML), diagramas de casos de uso, pruebas, manuales de usuario, manuales técnicos, etc.; todo con el propósito de eventuales correcciones, usabilidad, mantenimiento futuro y ampliaciones al sistema. Fuente: Wikipedia.

El ciclo de vida básico del desarrollo del *software* es:

- Ingeniería y modelado de sistemas
- Análisis de requerimientos
- Diseño
- Implementación
- Pruebas
- Mantenimiento

 © Universidad Internacional de La Rioja (UNIR)

Cada una de estas etapas debe ir acompañada por una documentación, que rige las etapas del ciclo de vida.

11.3. Documentación de diseño

La documentación de diseño debe reflejar las decisiones adoptadas para que la futura implementación del *software* cumpla con los requisitos.

Generalmente el diseño produce:

Diseño de datos	Diseño de la arquitectura del *software*
Diseño de interfaz	Diseño de algoritmos

Diseño de datos

El diseño de datos se basa en el **esquema entidad-relación** para obtener las estructuras de datos que se utilizarán en la implementación.

Los principios Wasserman para el diseño de datos son (Pressman ,2002):

- Los **principios de análisis sistemático** aplicados a la función y al comportamiento deberían aplicarse también a los datos.
- Todas las **estructuras de datos y las operaciones** a llevar a cabo en cada una de ellas deberían estar claramente identificadas.
- Se debería establecer un **diccionario de datos** y usarlo para definir el diseño de los datos y del programa.
- Las **decisiones de diseño de datos de bajo nivel** deberían dejarse para el final del proceso de diseño.
- La **representación de estructuras de datos** debería conocerla solo aquellos módulos que deban hacer uso directo de los datos contenidos dentro de la estructura.
- Debería desarrollarse una **biblioteca de estructuras** de datos útiles y de las operaciones que se puedan aplicar.
- Un **diseño del *software* y un lenguaje de programación** debería soportar la especificación y realización de los tipos abstractos de datos.

 © Universidad Internacional de La Rioja (UNIR)

Diseño de la arquitectura del *software*

El **diseño arquitectónico del *software*** involucra definir una estructura jerárquica de módulos y la forma en que dichos módulos interactúan entre sí.

Debe especificar:

Propiedades estructurales	Propiedades extra-funcionales	Familias de sistemas relacionados
Componentes del sistema e interacción entre ellos	Cómo se alcanzan los requisitos no funcionales: fiabilidad, rendimiento, etc.	Reutilización de bloques de construcción arquitectónica.

Diseño de interfaz

El diseño de interfaz describe cómo interactúa el software con los usuarios y con otros sistemas.

Se centra en:

- Interfaz entre los distintos módulos del *software*
- Interfaz con otros sistemas.
- Interfaz con los usuarios.

Diseño de algoritmos

El diseño de algoritmos o diseño procedimental especifica los detalles de los algoritmos.

Para realizar el diseño procedimental se utilizan:

- Construcciones estructuradas
- Diagramas de flujo
- Tablas de decisión
- Lenguaje de diseño de programas (LDP)

 © Universidad Internacional de La Rioja (UNIR)

11.4. Uso de herramientas para la documentación: JavaDoc

JavaDoc

Es una utilidad de **Oracle** para la generación de documentación de **API** en formato **HTML** a partir de código fuente **Java**. JavaDoc es el estándar de la industria para documentar clases de Java. La mayoría de los **IDE** los generan automáticamente. Fuente: Wikipedia.

Comentarios para generar documentación

El comentario que se utiliza en JavaDoc para generar la documentación debe comenzar por `/**`. Cada línea que forme parte del comentario debe ir precedida por `*`.

Ejemplo:

```
/**Descripción de MiClase
*  Línea 1
*  Línea 2
*  Línea 3
*  Línea 4
*/
```

Etiquetas

JavaDoc proporciona un **conjunto de etiquetas** para añadir información adicional a nuestro código. Estas etiquetas son palabras reservadas precedidas por el símbolo `@`.

Algunas de las etiquetas son:

- `@author` nombre: indica el nombre del desarrollador. La etiqueta permite indicar más de un autor, o si lo preferimos podemos utilizar tantas etiquetas `@author` como desarrolladores.

- `@deprecated` comentario: indica que el método o clase está desfasado y que por lo tanto no se recomienda su uso.

 © Universidad Internacional de La Rioja (UNIR)

- **`@param`** parámetro descripción: se utiliza para añadir un parámetro y su descripción a un método, en la documentación HTML (sección `Parameters`).

- **`@return`** descripción: añade la descripción de lo que devuelve el método en la documentación (sección `Returns`).

- **`@see`** referencia: añade una referencia a otro método, clase, etc.

- **`@throws nombre_clase`** descripción: añade descripción de la excepción que puede lanzar el método.

- **`@excepction nombre_clase`** descripción: igual que **`@throws`**.

- **`@version`** versión: añade la versión de la clase o método.

- **`@since`** texto: añade el número de versión desde la que existe el método.

 © Universidad Internacional de La Rioja (UNIR)

Lo + recomendado

No dejes de leer…

JavaDoc

Consulta la página de Oracle, en la que podrás encontrar información detallada sobre JavaDoc.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://www.oracle.com/technetwork/articles/java/index-137868.html

Tutorial JavaDoc

Tutorial sobre JavaDoc.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://www.tutorialspoint.com/java/java_documentation.htm

Documentación de programas

Ejemplo de un programa en Java documentado

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://www.dit.upm.es/~pepe/doc/fprg/javadoc.htm

 © Universidad Internacional de La Rioja (UNIR)

+ Información

A fondo

Herramientas de documentación ágiles

Artículo sobre la importancia de la documentación del código.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://issuu.com/_iti_/docs/actualidad_n11

 © Universidad Internacional de La Rioja (UNIR)

Test

1. ¿Cuál de las siguientes afirmaciones es correcta?

A. Las operaciones sobre las estructuras deben indicarse de manera abstracta.

B. Las decisiones de diseño de datos de bajos nivel deben dejarse para el final del diseño.

C. Los principios de análisis aplicados a la función no deben aplicarse a los datos.

D. La representación de las estructuras de datos deben conocerla todos los módulos.

2. El modelado de datos

A. Parte de tablas de decisión.

B. Parte de los diagramas de flujo.

C. Parte de un diccionario de datos.

D. Parte de los diagramas de entidad-relación.

3. El comentario que se utiliza en JavaDoc:

A. Debe comenzar por `@`

B. Debe comenzar por `/**`

C. Debe comenzar por `/**` y cada línea del comentario debe estar precedida por `*`

D. Debe comenzar por `//`

4. Las etiquetas utilizadas por JavaDoc:

A. Comienzan por `@`

B. Comienzan por `/**`

C. Comienzan por `/*`

D. Debe comenzar por `//`

5. La etiqueta `@version`:

A. Indica la descripción de la versión.

B. Añade una referencia.

C. Indica la versión de la clase o método.

D. Indica el número de versión desde la que existe el método.

 © Universidad Internacional de La Rioja (UNIR)

6. El diseño del *software* comprende:

A. Diseño de datos, diseño de interfaz, diseño procedimental y diseño de pruebas.

B. Diseño de datos, diseño arquitectónico, diseño de interfaz y diseño procedimental.

C. Diseño de datos, diseño arquitectónico y diseño de pruebas.

D. Diseño de arquitectónico, diseño de interfaz, diseño procedimental y diseño de pruebas.

7. El diseño de interfaz comprende:

A. Interfaz de usuario.

B. Interfaces internas e interfaces externas.

C. Interfaces internas entre módulos.

D. Interfaces internas, interfaces externas e interfaz de usuario.

8. Los diagramas de flujo se utilizan en:

A. Diseño procedimental.

B. Diseño de interfaz.

C. Diseño de arquitectura.

D. Diseño de datos.

9. Respecto a la herramienta JavaDoc, ¿qué afirmación es correcta?

A. Genera documentación en formato HTML a partir de código fuente Java.

B. Las etiquetas se insertan en los comentarios.

C. Los comentarios soportan la incrustación de código HTML estándar.

D. Todas las afirmaciones anteriores son correctas.

10. ¿Cuál de las siguientes afirmaciones es incorrecta?

A. El diseño de la interfaz forma parte del diseño del *software*.

B. El diseño de pruebas forma parte del diseño del *software*.

C. El diseño de datos forma parte del diseño del *software*.

D. El diseño de la arquitectura forma parte del diseño del *software*.

 © Universidad Internacional de La Rioja (UNIR)