por Mari Chelo Rubio

Entornos de Desarrollo
Bloque 3
Tema 2: Documentación del software. Herramientas

3.2.1. Introducción

No importa lo bueno que sea un software, si la documentación no es lo suficientemente buena, la gente no lo usará. Y si por alguna razón lo deben usar, sin una buena documentación, no la harán de forma efectiva o como desearías que lo hicieran. Para crear documentación bien hecha, se debe tener en cuenta la siguiente clasificación de tipos de documentación y así orientar dicha documentación a su propósito final.

3.2.2. Tipos de documentación

Según su uso la documentación se podría dividir en:

- **Tutoriales**
    - Orientado al aprendizaje
    - Permite a las personas introducirse
    - Es una lección
    - Analogía: enseñar a cocinar a un niño pequeño
- **How-to guides**
    - Orientado a resultados
    - Muestra cómo resolver una necesidad concreta
    - Una serie de pasos
    - Analogía: la receta concreta de un libro de cocina
- **Explicación**
    - Orientada a entender
    - Explica
    - Ofrece contexto
    - Analogía: Un artículo sobre la historia social culinaria
- **Referencia**
    - Orientada a informar
    - Describe cómo está construido

- Es exacto y completo
- Analogía: referencia de una entrada en la enciclopedia

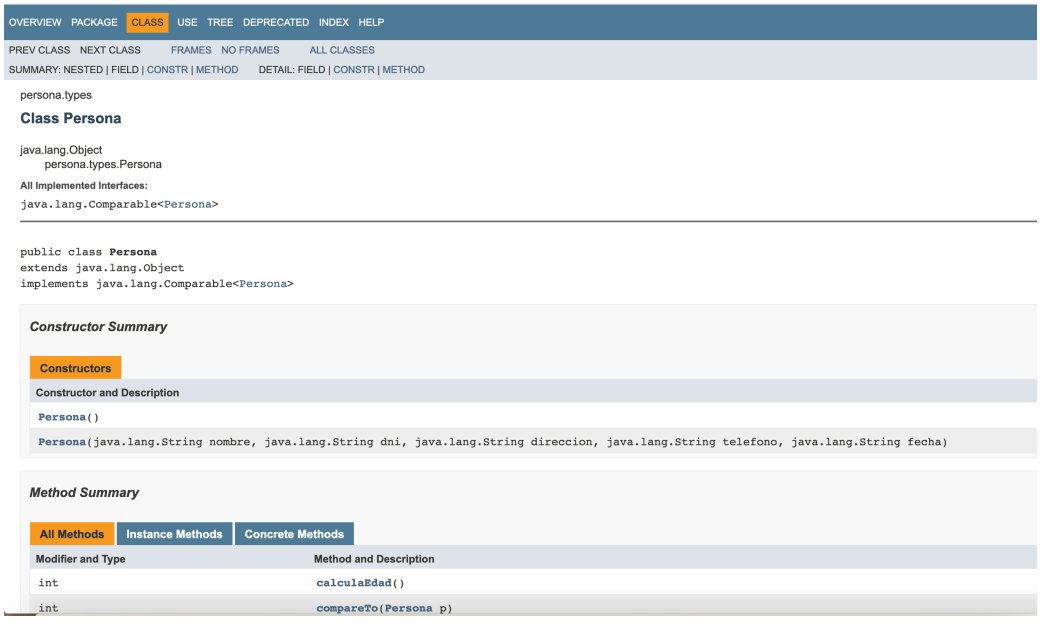
3.2.3. Como construir una Referencia con JavaDoc

Documentar un proyecto es algo fundamental de cara a su futuro mantenimiento. Cuando programamos una clase, debemos generar documentación lo suficientemente detallada sobre ella como para que otros programadores sean capaces de usarla sólo con su interfaz. No debe existir necesidad de leer o estudiar su implementación, lo mismo que nosotros para usar una clase del API Java no leemos ni estudiamos su código fuente.

Javadoc es una utilidad de Oracle para la generación de documentación de APIs en formato HTML a partir de código fuente Java. Javadoc es el estándar para documentar clases de Java. La mayoría de los IDEs utilizan javadoc para generar de forma automática documentación de clases.

La forma de generar esta documentación consiste en añadir ciertos comentarios que siguen una sintaxis específica en nuestro código. De esta forma se generará una referencia similar a las que puedes encontrar en la referencia de Java de Oracle.

En *Eclipse* en el menú principal, en proyecto, existe una opción de generar Javadoc. En *NetBeans* pulsando botón derecho sobre el proyecto. Si le damos a esta opción, aún sin haber añadido comentario alguno, nos saldrá una referencia como esta:



Ahora para que la documentación esté completa debemos comenzar a poner los comentarios adecuados a nuestro código.

3.2.3.1. Datos necesarios

Los datos necesarios para documentar una clase son:

- Nombre de la clase, descripción general, número de versión, nombre de autores.
- Documentación de cada constructor o método (especialmente los públicos) incluyendo: nombre del constructor o método, tipo de retorno, nombres y tipos de parámetros si los hay, descripción general, descripción de parámetros (si los hay), descripción del valor que devuelve.

3.2.3.2. Comentarios Javadoc

La documentación para javadoc ha de incluirse entre símbolos de comentario que han de empezar con una barra y doble asterisco, y terminar con un asterisco y barra simple.

Según donde se ponga el comentario Javadoc interpretará que se trata de un comentario de clase, de método o constructor.

```
3  import java.time.LocalDate;
4  import java.time.Period;
5
6  /**
7   * Clase que define personas y calcula su edad
8   * @author mariaconsuelorubiosanchez
9   * @version 2.1
0   *
1   */
2
3  public class Persona implements Comparable<Persona>
4  {
```

En la imagen anterior se ha definido un comentario de clase. Ese comentario supone la definición general de la clase. Se suelen incluir una serie de anotaciones que Javadoc interpreta para generar la documentación. En la imagen anterior podemos ver @author y @version que suponen el autor y la versión de la clase en cuestión. En el comentario de clase se puede incluir también las anotaciones @see y @since. La primera se suele poner con enlaces a ficheros externos relaciones para consultar y ampliar, significa *See Also* (vea también). La anotación @since suele ir seguida de una fecha que puede indicar la fecha de inicio del desarrollo de la clase.

```
/**
 * Clase que define personas y calcula su edad
 * @author mariaconsuelorubiosanchez
 * @version 2.1
 * @see https://iessanvicente.com
 * @since 01/04/20019
 *
 */
```

Después de este comentario general de la clase, pasamos a documentar los métodos. Para esto utilizaremos las anotaciones: @return para especificar lo que devuelve el método en cuestión y @param para especificar cada uno de los parámetros que debe recibir el método para funcionar correctamente.

Constructores:

```
/**
* Constructor con parámetros.
* @param nombre String correspondiente al nombre de la persona
* @param dni String correspondiente al dni de la persona
* @param direccion String correspondiente a la direccion de la persona
* @param telefono String correspondiente a el telefono de la persona
* @param fecha String correspondiente a la fecha de nacimiento de la pers
*/

public Persona(String nombre,String dni,String direccion,String telefono,
{

    setFechaNac(fecha);
    this.nombre=nombre;
    this.dni=dni;
    this.direccion=direccion;
    this.telefono=telefono;
}
```

Documentamos también getters y Setters:

```
/**
*
* @return nombre
*/

public String getNombre() {
    return nombre;
}

/**
*
* @param nombre
*/

public void setNombre(String nombre) {
    this.nombre = nombre;
}

/**
*
* @return String con la forma dd/mm/yy con la fecha de nacimiento
* de la persona
*/
public String getFechaNac()
{
    int day= fechaNac.getDayOfMonth();
    int month= fechaNac.getMonthValue();
    int year=fechaNac.getYear();
    return day+"/"+month+"/"+year;
}

/**
*
* @param fecha en forma dd/mm/yy para guardar como fecha de nacimiento
*/
public void setFechaNac(String fecha)
{
    String[] fechas=fecha.split("/");
    if (fechas.length <3)
    {
        fechaNac=LocalDate.of(1, 1, 1);
    }
    else
    {
        fechaNac=LocalDate.of(Integer.parseInt(fechas[2]),
            Integer.parseInt(fechas[1]),Integer.parseInt(fechas[0]));
    }
}
```

Y otros métodos:

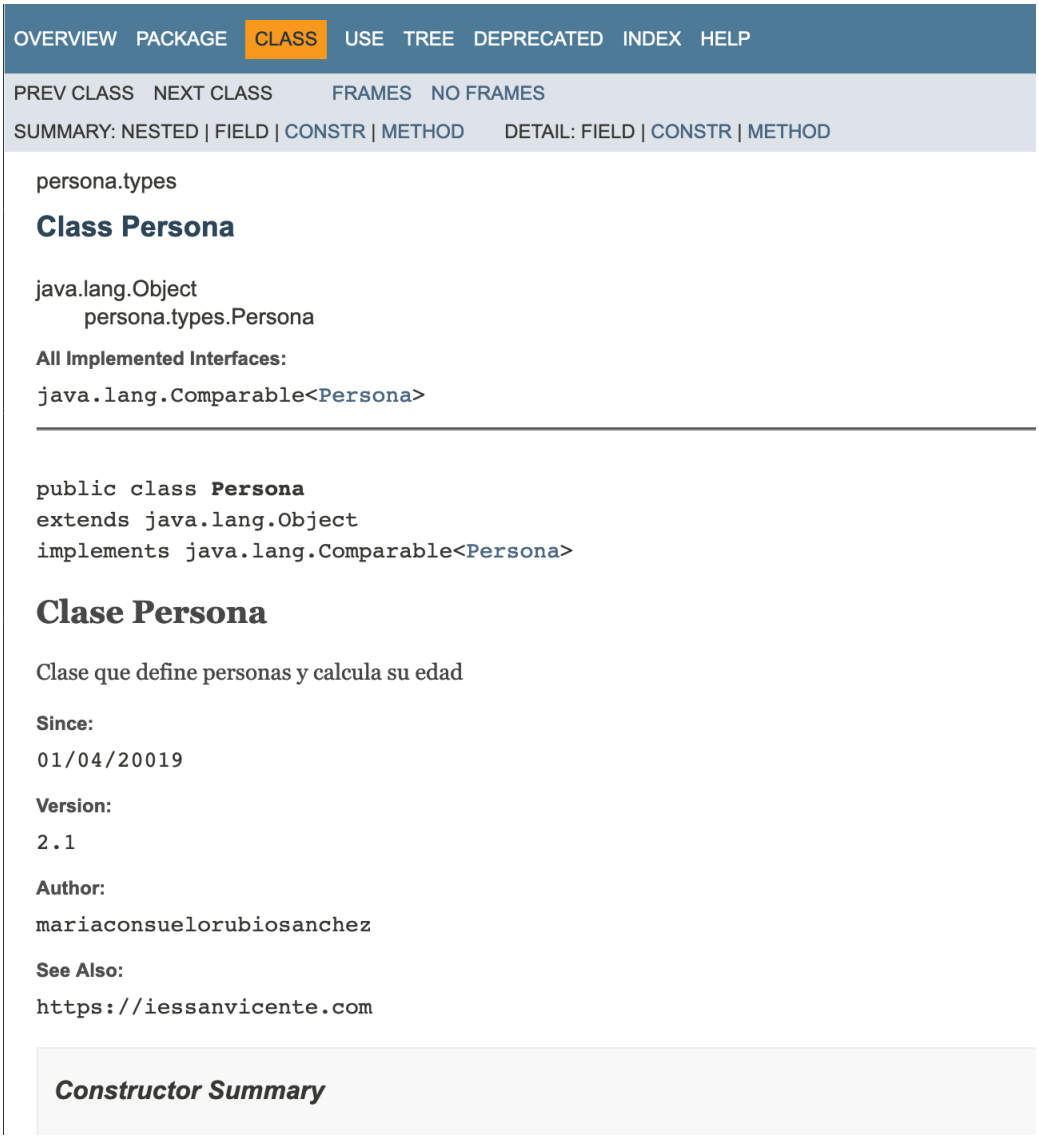
```
/**
* Método que calcula la edad de la persona a partir de su fecha de nacimi
* y la fecha actual
* @return entero que representa los años que tiene la persona
* en el momento actual
*/
public int calculaEdad()
{
    LocalDate ahora =LocalDate.now();
    Period periodo=fechaNac.until(ahora);
    return periodo.getYears();
}
```

Una vez realizados todos los comentarios volvemos a generar la documentación con Javadoc y saldrán los comentarios que hemos añadido como parte de dicha documentación.

Como Javadoc genera un código HTML para mostrar la documentación, podemos añadir etiquetas HTML nosotros también en los comentarios Javadoc. Por ejemplo:

```
/**
 * <h1>Clase Persona<h1>
 * <p>Clase que define personas y calcula su edad</p>
 * @author mariaconsuelorubiosanchez
 * @version 2.1
 * @see https://iessanvicente.com
 * @since 01/04/20019
 *
 */
```

Quedaría:



OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

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

persona.types

Class Persona

java.lang.Object
persona.types.Persona

All Implemented Interfaces:

`java.lang.Comparable<Persona>`

```
public class Persona
extends java.lang.Object
implements java.lang.Comparable<Persona>
```

Clase Persona

Clase que define personas y calcula su edad

Since:

`01/04/20019`

Version:

`2.1`

Author:

`mariaconsuelorubiosanchez`

See Also:

`https://iessanvicente.com`

Constructor Summary

Ejercicios propuestos:

3.2.3.1. Documenta adecuadamente la clase *Animals* desarrollada en el tema 7 en el ejercicio **2.7.1.1.**

3.2.3.2. Documenta también la clase ObjetoCultural del ejercicio **2.7.1.2.**

3.3. Otras herramientas

Existen algunas otras herramientas para generar documentación.

- **Doxygen**: Aplicación que genera documentación a partir de comentarios en el código para lenguajes como C/C++/C#/Objective-C/PHP/Java. Admite la sintaxis Javadoc que hemos visto en el punto anterior y tiene más comandos orientados a los demás lenguajes.
- **Atlassian Confluence**: Jira de Atlassian es uno de los productos para desarrollo ágil más utilizado. El propio fabricante tiene otro módulo muy interesante denominado Confluence que permite documentar los how-to y las referencias técnicas llegando a asociar Ticket de Jira- Documentación

de código – Rama de Git (o Subversion)Esto posibilita una visual del producto mucho más eficaz que ayuda a mantener un histórico de decisiones y una gestión de producto ágil.

- **Read the Docs**: Plataforma gratuita en la nube para hacer una gestión documental de lo generado.
- **SWAGGER**: Otra herramienta para documentar el flujo entero de desarrollo

3.4. Más información

Javadoc en oracle

Documentar con Javadoc

Doxygen