Conceptualización

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 solo con su interfaz; esto es: no debe existir la necesidad de leer o estudiar su implementación para entender que hace. Así como los desarrolladores utilizan una clase de la biblioteca de Java sin necesidad de estudiar el código fuente de ella, de la misma manera otros desarrolladores deben poder utilizar las clases desarrolladas por otros desarrolladores sin necesidad de estudiar el "cómo" logran su funcionalidad. Esto significa que, a la hora de elegir entre una u otra clase para desarrollar una actividad debe ser suficiente recurrir a la documentación del código que indica "que o cual" es la funcionalidad de los métodos de las clases y las características de esas clases a través de sus atributos. ***Javadoc*** es una utilidad del JDK para la generar la documentación del código en formato html. ***Javadoc*** es el estándar para documentar clases Java. La mayoría de los IDEs utilizan ***javadoc*** para generar de forma automática la documentación de clases.
Javadoc documenta los comentarios que comienzan con /** (notar el doble *) y que terminan con */. A la vez, dentro de estos comentarios se puede escribir código HTML y operadores para que sean interpretados por ***javadoc*** (generalmente precedidos por @):

Tag	Descripción	Uso
@author	Nombre del desarrollador creador de la clase o interfaz	nombre
@deprecated	Indica que el método o clase es antigua y que no se recomienda su uso porque posiblemente desaparecerá en versiones posteriores	descripción
@param	Definición de un parámetro de un método, es requerido para todos los parámetros del método	nombre_parametro descripción
@return	Informa lo que devuelve el método, no se puede usar en constructores o métodos "void"	descripción
@see	Asocia con otro método o clase, brinda más información o elementos relacionados	referencia (#método(); clase#método(); paquete.clase; paquete.clase#método()).
@throws	Excepción lanzada por el método (lo veremos más adelante)	nombre_clase descripción
@version	Versión del método o clase	versión

Las etiquetas @author y @version se usan para documentar clases e interfaces. Por tanto, no son válidas en cabecera de constructores ni métodos. La etiqueta @param se usa para documentar constructores y métodos. La etiqueta @return se usa solo en métodos que devuelven un valor.
Observe el siguiente ejemplo:

```
import java.util.Date;

/**
 * Clase que representa a cualquier Jugador registrado en
 * el Torneo. Es importante considerar que sólo puede pertenecer
 * a un Equipo inscripto.
 *
 * @author Ariel Alejandro Vega
 * @see futsal.equipos.Equipo
 */
public class Jugador {

	private String nombre;
	private String apellido;
	private Date fechaNacimiento;

	/**
	 * Calculamos la edad del jugador en base a su fecha de nacimiento
	 *
	 * @return Edad
	 */
	public int calcularEdad () {
		// TODO recordar implementar este método
		return 0;
	}

	/**
	 * Comprobamos si el Jugador puede participar en un torneo en
	 * base a la edad mínima pasada por parámetros.
	 *
	 * @param edadMinima Edad mínima (inclusive) en años
	 * @return Verdadero si puede participar
	 */
	public boolean puedeParticiparPorSuEdad (int edadMinima) {
		return edadMinima >= this.calcularEdad();
	}

}
```

En el ejemplo anterior puede notarse un comentario que inicia con las letras “TODO”, se trata de una convención utilizada por los desarrolladores para denotar código que aún debe implementarse, en inglés “To Do” o “Pendiente”. Algunos IDE agregan automáticamente estos comentarios y además permiten resaltarlos para recordarnos las porciones de código que nos faltan escribir. En Eclipse se vería así:

```
24    public int calcularEdad () {
25        // TODO recordar implementar este metodo
26        return 0;
27    }
```

La documentación generada por JavaDoc tendra una apariencia similar a la captura de pantalla de la Figura 1.

Cuando documentamos para documentación el IDE utiliza esta documentación para brindarnos ayuda de la clase, constructor, método y atributo comentado. Para ello simplemente debe posar el puntero del mouse sobre la clase, constructor, método o atributo (solo si accede directamente al atributo) que desea utilizar. Por ejemplo, observe el código documentado para una clase `Libro` y otra `Autor` (Figuras 2 y 3). En otra clase denominada `Principal` utilizamos estas clases, podemos ver como aparece el comentario de documentación de la clase `Autor` simplemente al posar el puntero del mouse sobre el tipo de datos `Autor` (Figura 4) y la descripción del método usado del objeto `unLibro` (Figura 5).

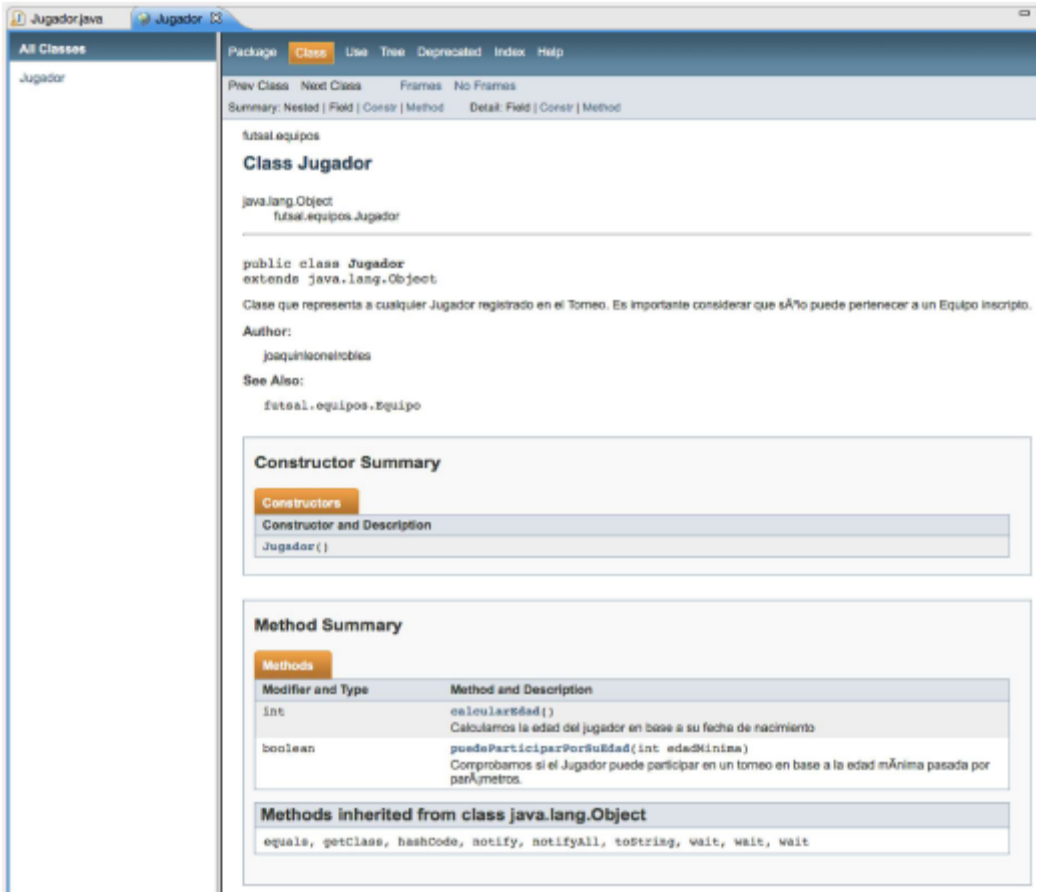


Figura 1. Ejemplo de documentación generada por javadoc

```java
package ar.edu.unju.fi.model;

/**
 * Clase que representa a un Autor de un libro. Es importante
 * considerar que un autor pudo haber escrito muchos libros
 * @author Ariel Alejandro Vega
 * @version 1.0
 *
 */
public class Autor {
    /*
     *---------------------------------
     *-------- atributos --------------
     *---------------------------------
     */

    /**
     * Atributo que representa el apellido del Autor
     */
    private String apellido;
    /**
     * Atributo que representa los nombres del autor
     */
    private String nombres;

```

```java
/*
 *---------------------------------
 *-------- constructores --------------
 *---------------------------------
 */

/**
 * Constructor por defecto
 */
public Autor() {
    // TODO Auto-generated constructor stub
}

/**
 * Constructor parametrizado
 * @param apellido valor del apellido del Autor
 * @param nombres valores de los nombres del Autor
 */
public Autor(String apellido, String nombres) {
    // asignacion del parámetro apellido al atributo apellido
    this.apellido = apellido;
    // asignacion del parámetro nombres al atributo nombres
    this.nombres = nombres;
}

/*
 *---------------------------------
 *-------- metodos accesores --------------
 *---------------------------------
 */

/**
 * Devuelve el apellido del Autor
 * @return this.apellido
 */
public String getApellido() {
    return apellido;
}

/**
 * Asigna un valor al apellido del Autor
 * @param apellido valor del apellido
 */
public void setApellido(String apellido) {
    this.apellido = apellido;
}

/**
 * Devuelve los nombres del Autor
 * @return this.nombres
 */
public String getNombres() {
    return nombres;
}

/**
 * Asigna un valor a los nombres del Autor
 * @param nombres valor para los nombres
 */
public void setNombres(String nombres) {
    this.nombres = nombres;
}
```

Figura 2. Clase Autor Documentada

```java
package ar.edu.unju.fi.model;

import java.time.LocalDate;

/**
 * Clase que representa el libro de la biblioteca
 * que se puede prestar
 *
 * @author Ariel Alejandro Vega
 *
 */
public class Libro {
    private String titulo;
    private Autor autor;
    private LocalDate fechaPrestamo;
    /**
     * EStablece si el libro ha sido prestado
     * true: prestado; false: disponible;
     */
    private boolean prestado;
    private LocalDate fechaDevolucion;

    /**
     * Cambia el estado del libro a prestado
     * y define la fecha de devolución
     * @param cantidadDias representa la cantidad de dias que se presta el libro
     */
    public void prestarLibro(int cantidadDias) {
        // como se presta el libro.. se cambia el estado a true
        this.prestado = true;
        // asignar a fechaPrestamo la fecha del sistema
        fechaPrestamo = LocalDate.now();
        // la fecha de devolucion es la fechaPrestamo + cantidad de dias
        fechaDevolucion = fechaPrestamo.plusDays(cantidadDias);
    }

    /**
     * Sobreescribe el método para que muestre la fecha de prestamo, si está prestado
     * y la fecha de devolución del Libro
     */
    @Override
    public String toString() {
        return "Libro [fechaPrestamo=" + fechaPrestamo + ", prestado=" + prestado + ", fechaDevolucion="
                + fechaDevolucion + "]";
    }
}
```

Figura 3. Documentación de la clase Autor.

```java
package ar.edu.unju.fi.principal;

import ar.edu.unju.fi.model.Autor;

public class Principal {

    public static void main(String[] args) {

        Autor a = new Autor("Vega", "Ariel");

    }

}
```

ar.edu.unju.fi.model.Autor

Clase que representa a un Autor de un libro. Es importante considerar que un autor pudo haber escrito muchos libros

Version:
1.0
Author:
Ariel Alejandro Vega

Press 'F2' for focus

Figura 4. El IDE Eclipse muestra los comentarios de documentación de la clase

```java
package ar.edu.unju.fi.principal;

import ar.edu.unju.fi.model.Autor;

public class Principal {

    public static void main(String[] args) {

        Autor a = new Autor("Vega", "Ariel");

        Libro unLibro = new Libro();
        unLibro.prestarLibro(4);

        System.o
    }

}
```

void ar.edu.unju.fi.model.Libro.prestarLibro(int cantidadDias)

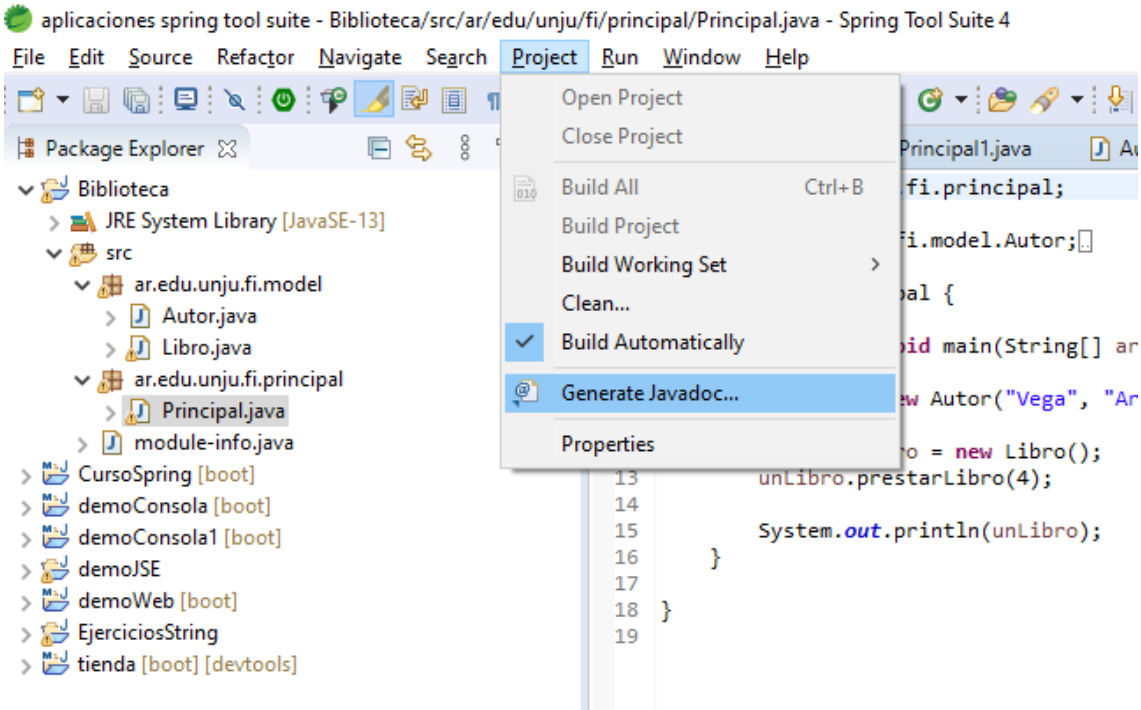
Cambia el estado del libro a prestado y define la fecha de devolución

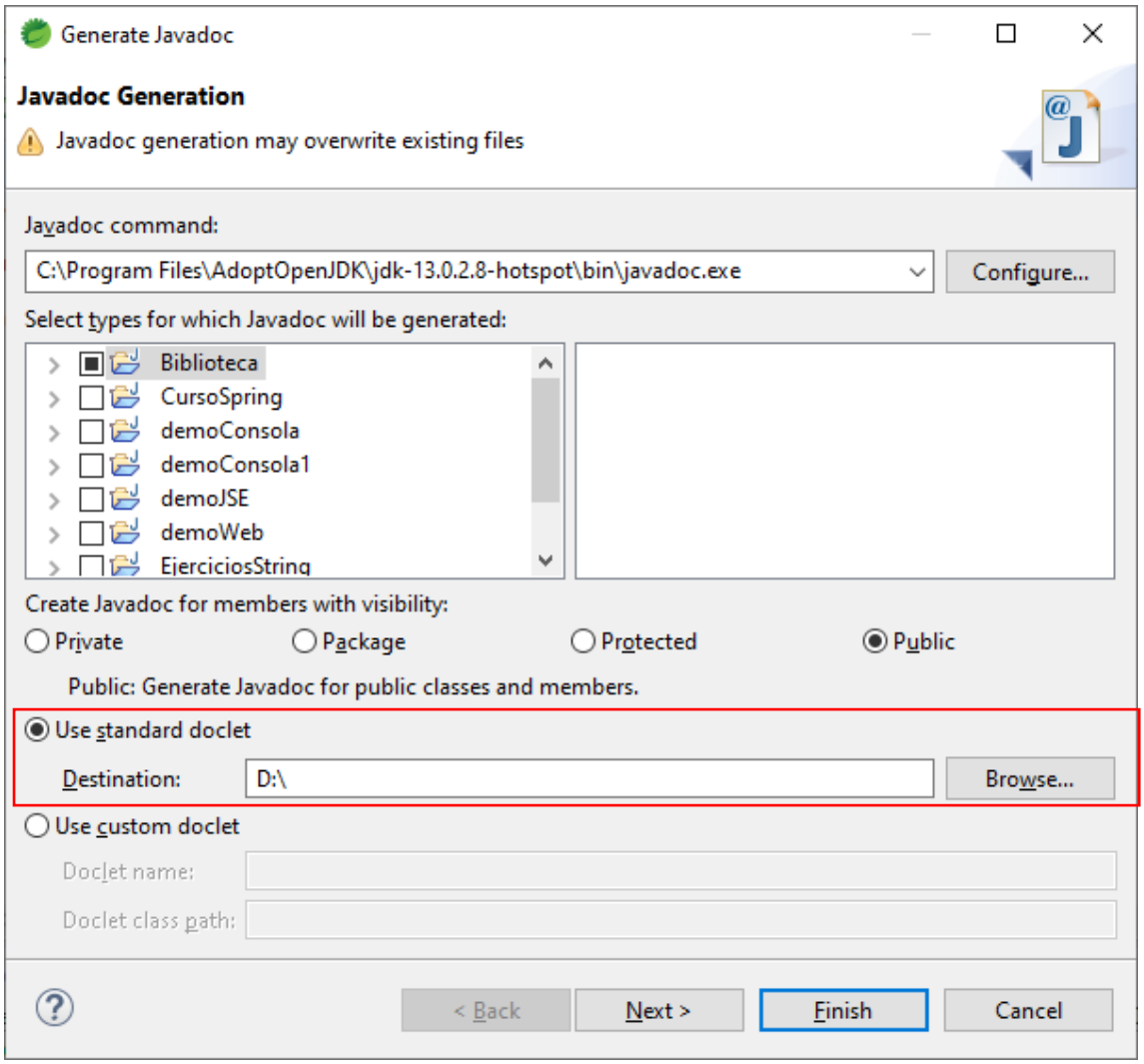
Parameters:
cantidadDias representa la cantidad de dias que se presta el libro

Press 'F2' for focus

Figura 5. Ejemplo de visualización del comentario de documentación de un método en el IDE Eclipse

Finalmente, para generar la documentación del Código usando javadoc desde el IDE Eclipse se debe ir al menú Project-> Generate javadoc y luego elegir donde se guardará las paginas generadas.




Generate Javadoc
Javadoc Generation
Javadoc generation may overwrite existing files
Javadoc command:
C:\Program Files\AdoptOpenJDK\jdk-13.0.2.8-hotspot\bin\javadoc.exe
Configure...
Select types for which Javadoc will be generated:
Biblioteca
CursoSpring
demoConsola
demoConsola1
demoJSE
demoWeb
EjerciciosString
Create Javadoc for members with visibility:
Private
Package
Protected
Public
Public: Generate Javadoc for public classes and members.
Use standard doclet
Destination:
D:\
Browse...
Use custom doclet
Doclet name:
Doclet class path:
< Back
Next >
Finish
Cancel