Documentación

Motivación

¿A quién interesa el código fuente?

- Autores del propio código
- Otros desarrolladores del proyecto
- Clientes de la API del proyecto

¿Por qué documentarlo?

- Mantenimiento
- Reutilizar

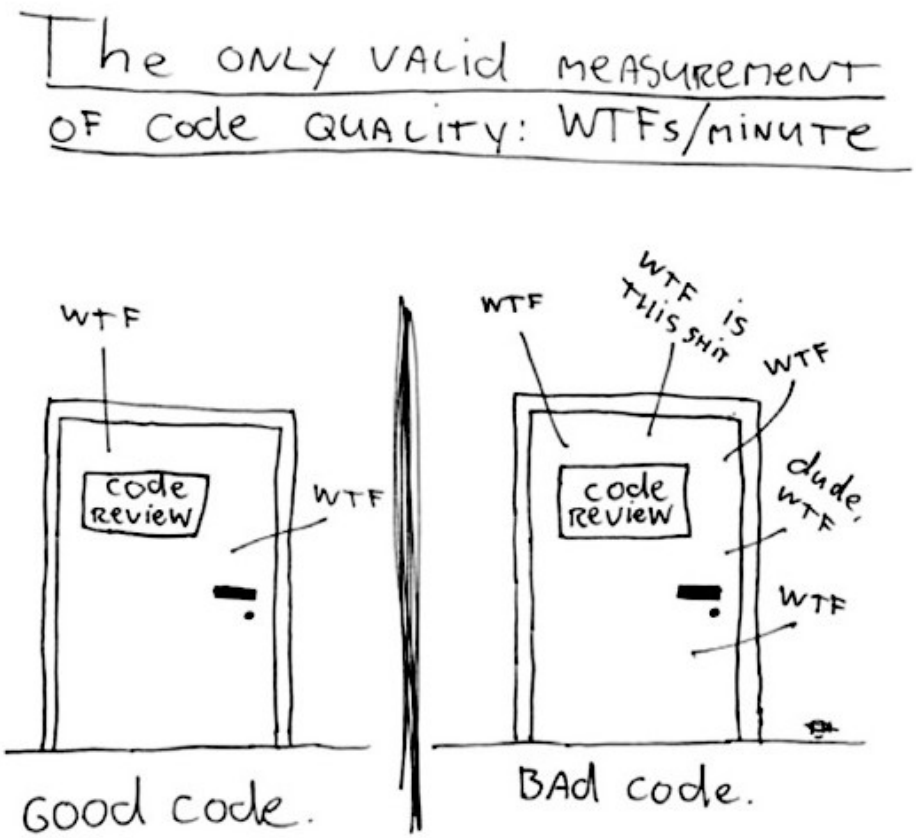
¿Qué es un comentario?

- Añadir información a nuestro código que ayude a entender:
    - qué es lo que hace
    - Por qué así

¿Cómo se comenta?

- Cometario de una línea
- Comentario multilínea
- Comentario javadoc

Código autocomentado:



¿Qué documentar?

- Paquetes, Clases, Constructores, métodos y atributos.

¿Cómo generamos la documentación?

Generarla a mano es tedioso y propenso a errores:

- Gran cantidad de pequeños detalles
- Sincronización de código fuente y documentación
- Duplicidad de esfuerzos (tipos, nombres…)

Vamos a combinar el código fuente con la documentación y generaremos la documentación desde el código.

Javadoc:

- Genera la documentación en HTML
- Usa la información de nombres, tipos, …
- Explicaciones adicionales y referencias cruzadas.
- Otras herramientas se apoyan en javadoc para ayudar a los desarrolladores (por ejemplo, Eclipse).

Comentarios en Javadoc:

Comentarios, con una sintaxis concreta, que se ubican antes de las clases, interfaces, constructores, métodos y atributos a documentar.

/**

* Descripción del trozo de código que se va documentar. (texto en HTML)

* Tags (texto con etiquetas en HTML)

* Una serie de etiquetas: @author, @param, …

*/

Reglas

- La primera frase de cada comentario en Javadoc debe ser una frase resumen con **una descripción concisa y completa,** terminada en punto, y seguida de un espacio, tabulador o retorno de carro.
- Usar la **etiqueta** <code> para palabras clave y nombres
- Se prefiere el uso de la tercera persona:

  Devuelve una cadena de caracteres en minúsculas.

  ~~Devolvemos una cadena de caracteres en minúsculas.~~

- Empezar con un verbo la descripción de los métodos
- Omitir el sujeto cuando es obvio

  @param radio radio de la circunferencia

  ~~@param radio parámetro que indica el radio de la circunferencia~~

Etiquetas:

Palabras claves gestionadas de forma especial.

@param name descripción

- Aplicable a parámetros de constructores y métodos
- Describe los parámetros del constructor / método
- Name: idéntico al nombre del parámetro
- Debe haber una por cada parámetro del método.

@return description

- Aplicable a métodos
- Describe el valor de retorno del método
- Incluir descripción de valores de retorno especiales: null…

@throws, @exception (una u otra, indistintamente. Son sinónimos.)

- Indica que el bloque de código puede lanzar una excepción determinada.
- Aplicable a constructores y métodos
- Describe posibles excepciones del constructor / método
- Puede haber más de una por bloque de código
- Un throws por cada posible excepción
- Si es de ayuda para el usuario, también se pueden documentar las unchecked exception

@see reference

- Aplicable a clases, interfaces, constructores, métodos, atributos y paquetes.
- Añade enlace de referencia a otras partes de la documentación. Puede ser una referencia a un libro, a un enlace en internet, a otro método, a una clase, etc…
    1. @see java.lang.Math
    2. @see java.lang.Math La clase Math
    3. @see Math#random(double)
    4. @see Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley professional computing series.
    5. @see Java Documentation

@deprecated

- Marca el método o la clase como obsoleta

@author

- Aplicable sólo a clases
- Indica el autor de la clase

@version

- Aplicable sólo a clases
- Indica el número de versión actual del bloque de código

@since

- Indica el número de versión desde la que está disponible el código.

Comentar paquetes y módulos:

Para poder documentar los paquetes y los módulos, hay que generar dos ficheros adicionales:

- package-info.java
- module-info.java

package-info.java

```java
/**
 * <p>
 * La Geometría Euclidiana es un sistema matemático que corresponde al estudio de las
 * propiedades geométricas de los espacios euclideos. En ella se se satisfacen los Axiomas
 * definidos en los postulados de Euclides.
 * <p>
 * Todas estas teorías estan descritas en los famosos libros de Euclides «Los Elementos» que puedes encontrar
 * detallados en nuestra web.
 *@see <a href="https://es.wikipedia.org/wiki/Geometr%C3%ADa_euclidiana">Geometria Geometría euclidiana</a>
 *@author Ramón
 */
package org.villablanca.figuras;
```

module-info.java

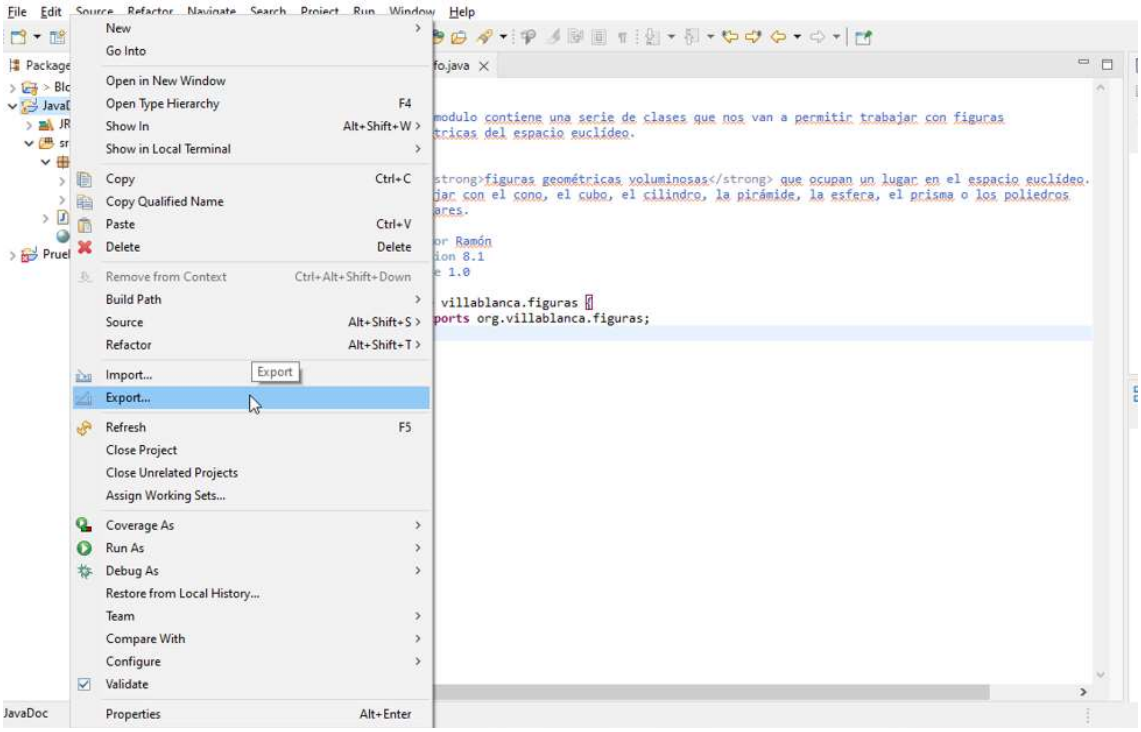
```java
/**
 *<p>
 *Este modulo contiene una serie de clases que nos van a permitir trabajar con figuras
 *geométricas del espacio euclídeo.
 *</p>
 *<p>
 *Las <strong>figuras geométricas voluminosas</strong> que ocupan un lugar en el espacio euclídeo. Vamos a poder
 *trabajar con el cono, el cubo, el cilindro, la pirámide, la esfera, el prisma o los poliedros
 *regulares.
 *</p>
 *@author Ramón
 *@version 8.1
 *@since 1.0
 */
module villablanca.figuras {
    exports org.villablanca.figuras;
}
```

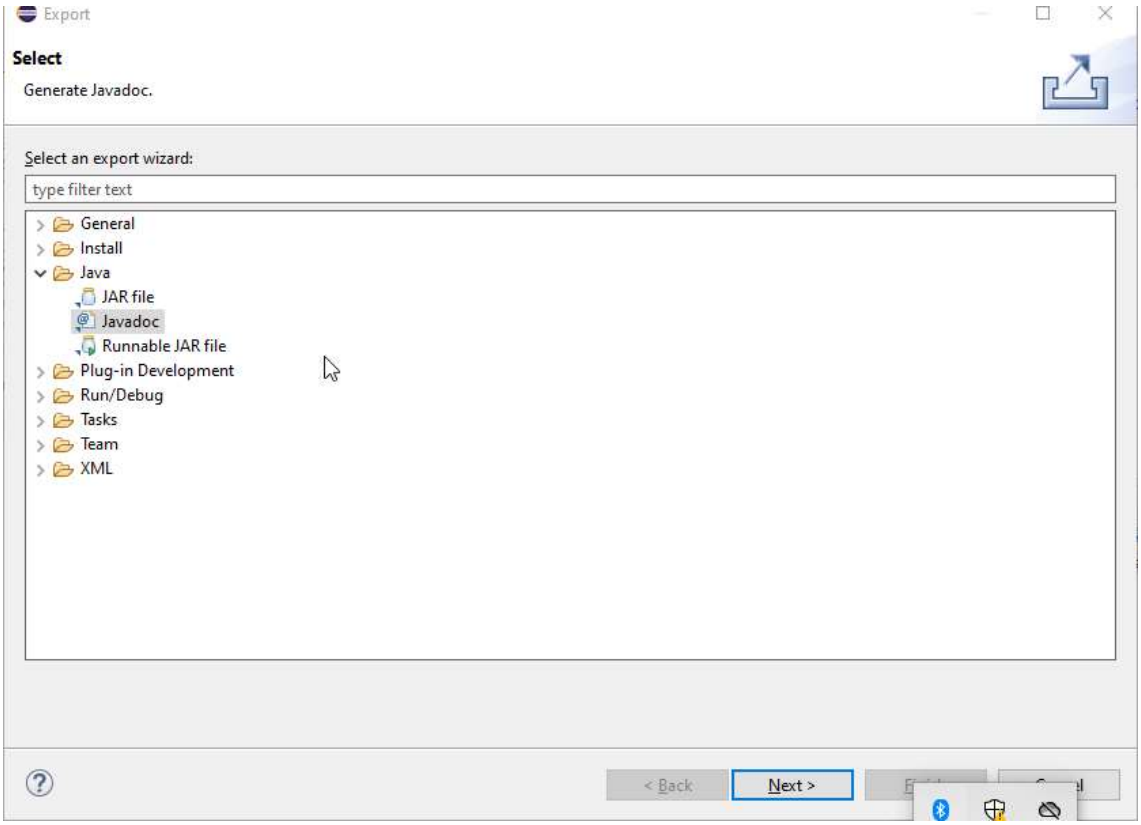
Overview

- Página de bienvenida
- Muestra un texto html y la lista de módulos o paquetes del proyecto
- Tiene que ser un documento html completo y lo que se verá posteriormente en la documentación es todo lo que colocaremos dentro del body.

Generar documentación con javadoc en eclipse

- Proyecto -> Export





Ejemplo:

```
/**
 * Una clase para representar círculos situados sobre el plano.
 * Cada círculo queda determinado por su radio junto con las
 * coordenadas de su  centro.
 * @version 1.2, 24/12/2021
 * @author Ramón
 */
public class Círculo {
   protected double x,y; // coordenadas del centro
   protected double r;  // radio del círculo

   /**
    * Crea un círculo a partir de su origen su radio.
    * @param x La coordenada x del centro del círculo.
    * @param y La coordenada y del centro del círculo.
    * @param r El radio del círculo. Debe ser mayor o igual a 0.
    */
   public Circulo(double x, double y, double r) {
      this.x=x; this.y = y; this.r = r;
   }

   /**
    * Cálculo del área de este círculo.
    * @return El área (mayor o igual que 0) del círculo.
    */
    public double área() {
```

```
    return Math.PI*r*r;
  }

  /**
   * Indica si un punto está dentro del círculo.
   * @param px componente x del punto
   * @param py componente y del punto
   * @return true si el punto está dentro del círculo o false en otro caso.
   */
  public boolean contiene(double px, double py) {
    /* Calculamos la distancia de (px,py) al centro del círculo (x,y),
       que se obtiene como raíz cuadrada de (px-x)^2+(py-y)^2 */
    double d = Math.sqrt((px-x)*(px-x)+(py-y)*(py-y));

    // el círculo contiene el punto si d es menor o igual al radio
    return  d <= r;
  }
}
```