

## ¿Qué es Javadoc?

Javadoc es una utilidad que acompaña a todas las versiones del SDK jdk\*, cuyo fin es fomentar una correcta documentación del código en Java. Esta utilidad es un programa que, si un conjunto de clases escritas en Java están adecuadamente comentadas, genera automáticamente un conjunto de páginas HTML similares a las que se usan como documentación de las clases estándar de Java. Constituye por tanto un formato de documentación estándar y familiar para cualquier programador en Java, y fomenta asimismo la reutilización y compartición de código en Java.

Para más información vea la página oficial de javadoc:



# Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification.

## Generada con Javadoc

The screenshot shows the 'Overview' page of the Java 2 Platform, Standard Edition, v 1.4.2 API Specification. The browser address bar shows the URL: <http://java.sun.com/j2se/1.4.2/docs/api/index.html>. The page has a navigation bar with links: Overview, Package, Class, Use, Tree, Deprecated, Index, and Help. Below the navigation bar, the title 'Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification' is displayed. A paragraph states: 'This document is the API specification for the Java 2 Platform, Standard Edition, version 1.4.2.' Below this, there is a section titled 'See:' with a link to 'Description'. A table titled 'Java 2 Platform Packages' lists several packages: [java.applet](#), [java.awt](#), and [java.awt.color](#). The left sidebar contains a list of 'All Classes' and 'Packages'.

**Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification**

This document is the API specification for the Java 2 Platform, Standard Edition, version 1.4.2.

See:

[Description](#)

Java 2 Platform Packages	
<a href="#">java.applet</a>	Provides the classes ne applet and the classes a communicate with its a
<a href="#">java.awt</a>	Contains all of the class interfaces and for painti images.
<a href="#">java.awt.color</a>	Provides classes for co

The screenshot shows the 'Method Summary' page for the `String` class. The left sidebar lists various classes and packages, including `java.lang.String`. The main content area displays a list of methods with their signatures and descriptions. The methods listed are: `charAt(int index)`, `compareTo(Object o)`, `compareTo(String anotherString)`, `compareToIgnoreCase(String str)`, `concat(String str)`, `contentEquals(StringBuffer sb)`, `copyValueOf(char[] data)`, and `copyValueOf(char[] data, int offset, int count)`.

Method Summary	
char	<code>charAt(int index)</code> Returns the character at the specified index.
int	<code>compareTo(Object o)</code> Compares this String to another Object.
int	<code>compareTo(String anotherString)</code> Compares two strings lexicographically.
int	<code>compareToIgnoreCase(String str)</code> Compares two strings lexicographically, ignoring case differences.
String	<code>concat(String str)</code> Concatenates the specified string to the end of this string.
boolean	<code>contentEquals(StringBuffer sb)</code> Returns true if and only if this String represents the same sequence of characters as the specified StringBuffer.
static String	<code>copyValueOf(char[] data)</code> Returns a String that represents the character sequence in the array specified.
static String	<code>copyValueOf(char[] data, int offset, int count)</code>

## Documentación

En la documentación general de la implementación y especificación de un proyecto se deben contemplar dos aspectos:

- ✓ La documentación pública
- ✓ La documentación privada.

La primera se hace indispensable cuando ofrecemos un cierto conjunto de clases en algún dominio específico (un paquete). Al usuario de este paquete (quien a su vez es programador) le interesará saber qué clases están disponibles, qué hacen exactamente y bajo qué condiciones, es decir, la especificación completa de la interface correspondiente. Podemos así compartir la especificación pero con ocultamiento de información.

La documentación privada, por su parte, contempla no el uso, sino la misma implementación. En este caso se trata de describir cómo se hacen las cosas, qué estrategias se utilizan y, en general, a incluir toda anotación que ayude a explicar y clarificar una solución implementada.

Mantener presente estas dos perspectivas se constituye en una gran guía para el logro de una buena modularidad en el desarrollo de soluciones, tanto cuando se trata de paquetes, como en aplicaciones completas.

### Documentación pública

La documentación pública –que se suele referenciar en inglés como API, Application Programming Interface– se realiza utilizando la herramienta javadoc (incluido en el kit de desarrollo para Java, JDK). Este programa produce páginas HTML permitiendo examinar la documentación de manera ágil mediante un navegador o visualizador (Netscape, por ejemplo). La información correspondiente se hace a través de comentarios reconocibles por javadoc dentro del código fuente. Un comentario público está conformado por el texto que aparezca entre los caracteres `/**` y los caracteres `*/`.

/\*\*

\* Este es un comentario para documentación javadoc

\*/

El comentario debe comenzar con una descripción concisa pero completa de la entidad o elemento documentado. Los comentarios son reconocidos por javadoc sólo si se ubican inmediatamente antes de la declaración pública o protegida de clase, constructor, método, atributo o interface.

## Documentación privada

En general, la documentación privada debe seguir las mismas pautas que para la documentación pública. Aunque los comentarios especiales serán ignorados por javadoc –ya que los elementos documentados no son públicos– se procura de esta manera un estilo unificado que favorece la legibilidad y el mantenimiento (un elemento no público podría dejar de serlo en algún momento). Se debe hacer buen uso de comentarios concisos que ayuden a clarificar secciones de código que de por sí no sean autoexplicativos.

Los comentarios escritos en javadoc admiten dos características avanzadas:

1. La posibilidad de insertar etiquetas propias de HTML, como `<b></b>` ó `<i></i>`.
2. La posibilidad de incluir etiquetas que documenten características específicas del elemento comentado. Por ejemplo, una clase puede tener un autor, una versión, etc., y un método unos parámetros.

Las características específicas se documentan con etiquetas de la forma “@etiqueta”, en formato de una por línea.

Algunas de las características que reconoce javadoc son:

Etiqueta	Sintaxis	Aplicable a	Efecto
@see	@see <i>clase</i> @see <i>clase#método</i>	Clase, método	Crea, en la página generada para la clase, un hipervínculo a la página de la clase o al método dentro de ella.
@version	@version <i>texto</i>	Clase, interfaz	Muestra el texto en el campo versión de la página generada para la clase o interfaz.
@author	@author <i>texto</i>	Clase, interfaz	Muestra el texto en el campo autor de la página generada para la clase o interfaz.
@param	@param <i>variable texto</i>	Método	Agrega una entrada a la sección “Parameters” de la sección del método en la página generada para la clase.
@return	@return <i>texto</i>	Método	Agrega una sección “Returns” de la sección del método en la página generada para la clase.
@throws	@throws <i>clase texto</i>	Método	Agrega una sección “Throws” de la sección del método en la página generada para la clase.

## La plantilla de prácticas.

El esquema general de un archivo fuente en Java, incluyendo elementos de documentación, será el siguiente.

```
//
// Universidad de Almería
// Ingeniería Técnica de Informática de Sistemas
// Fuente Java según Plantilla
//
// PRACTICA : Título y descripción breve de la práctica
// ASIGNATURA : Nombre de la asignatura
//
package pqt1[.pqt2[.pqt3]];
import pqt1[.pqt2].(clase | *);
otros paquetes que se hacen visibles
/**
 * Descripción general de la clase pública definida: su finalidad
 * o propósito, su funcionalidad con respecto a su superclase y/o
 * interfaces implementadas, etc.
 *
 * @author Nombre(s) del (los) autor(es)
 * @version Número de versión y la fecha
 */
public class NombreClase [extends SuperClase]
[implements Interface1 [, Interface2...]]
{
    atributos
    constructores
    métodos
}
otras clases no públicas
```

## Cada atributo:

```
/**
 * descripción del campo
 */
tipo campo;
```

## Cada constructor:

```
/**
 * Descripción del constructor.
 * PRE: precondiciones
 * POS: poscondiciones
 * @param arg1 descripción del parámetro arg1
 * @param arg2 descripción del parámetro arg2
 * demás parámetros del constructor
 * @exception clase-excepción1 descripción de la causa
 * @exception clase-excepción2 descripción de la causa
 * demás excepciones que puede lanzar el constructor
 */
public NombreClase ( tipo arg1, tipo arg2, otros parámetros )
//
// Descripción breve de la estrategia de creación.
//
{
    implementación del constructor
}
```

## Cada método:

```
/**
 * Propósito del método público operacion
 * PRE: precondiciones
 * POS: poscondiciones
 * @param arg1 descripción del parámetro arg1
 * @param arg2 descripción del parámetro arg2
 * demás parámetros del método
 * @return descripción de los posibles valores de retorno
 * @exception clase-excepción1 descripción de la causa
 * @exception clase-excepción2 descripción de la causa
 * demás excepciones que puede lanzar el método
 */
public tipo operacion ( tipo arg1, tipo arg2, otros parámetros )
//
// Descripción breve de la estrategia usada por el método
//
{
    implementación del método
}
```

## Un ejemplo: la clase CeldaEntero

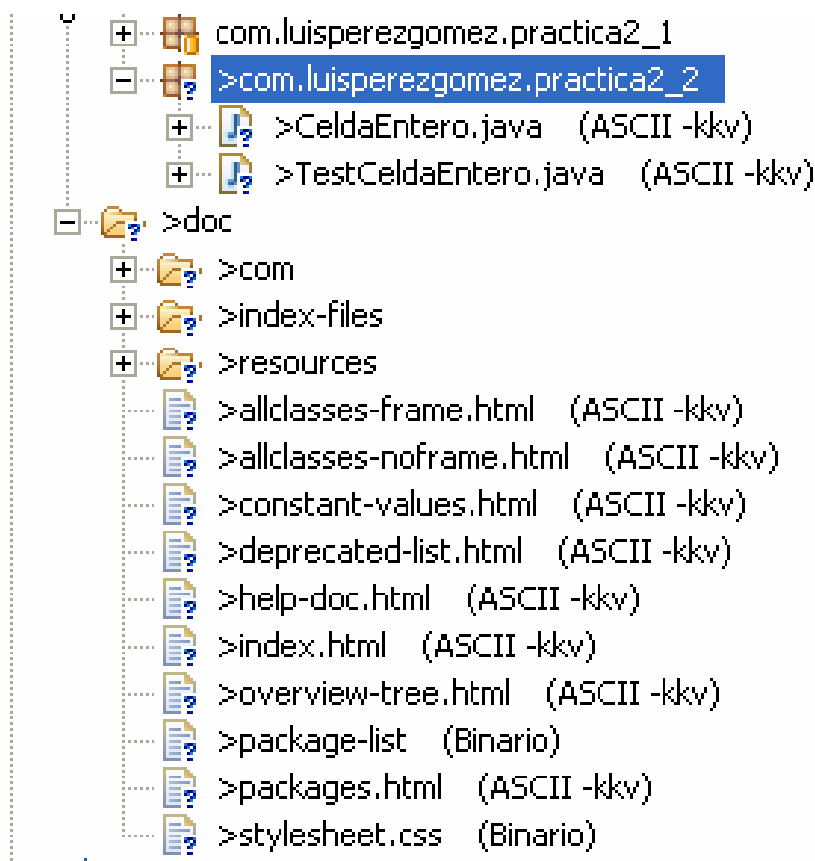
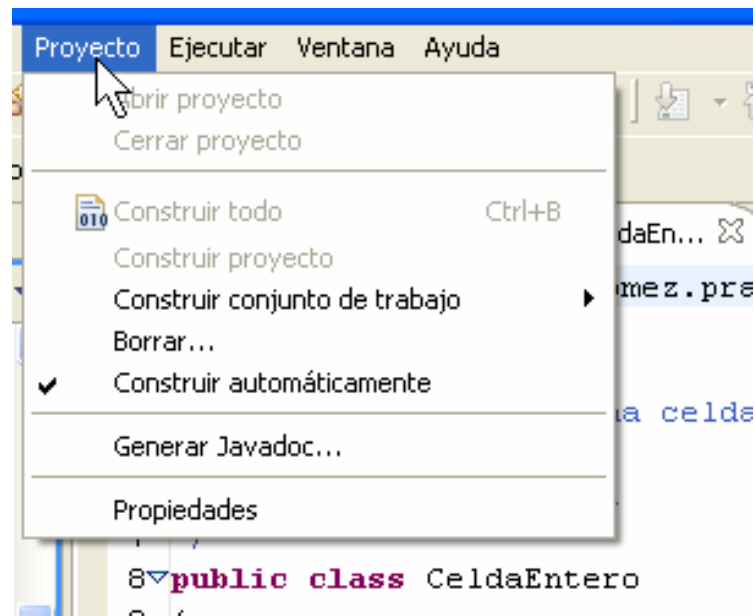
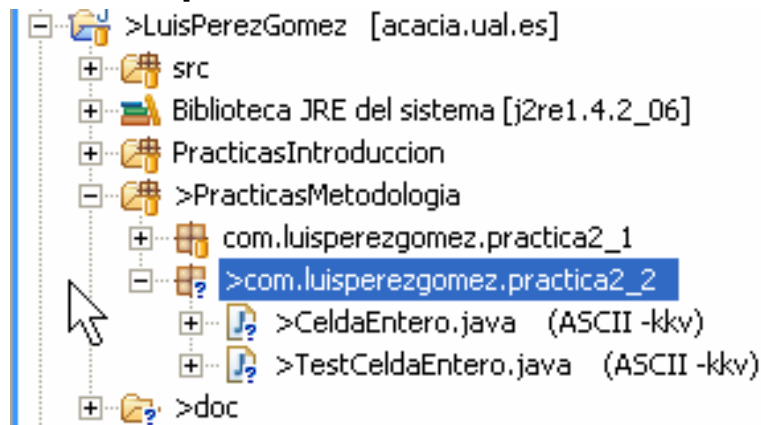
```
//
// Universidad de Almería
// Ingeniería Técnica de Informática de Sistemas
// Fuente Java según Plantilla versión
//
// PRACTICA : Ejemplo de documentación con javadoc
// ASIGNATURA : Metodología de la programación
//
/**
 * Clase para probar el funcionamiento de una celda de memoria para un entero
 * @author M.A. Weiss
 * @version 1.0. 03.2001
 */
public class TestCeldaEntero
{
    public static void main( String [ ] args )
    {
        CeldaEntero m = new CeldaEntero ( );
        // atención, esto seria un comentario del código que no saldría en la documentación...
        m.escribir( 5 );
        System.out.println( "Contenidos de la celda: " + m.leer( ) );
    }
}

/**
 * Clase para simular una celda de memoria para un entero
 * @author M.A. Weiss
 * @version 1.0. 03.2001
 */
public class CeldaEntero
{
    /**
     * Obtiene el valor almacenado.
     * @return el valor almacenado.
     */
    public int leer( )
    {
        return valorAlmacenado;
    }

    /**
     * Almacena un valor
     * @param x el número a almacenar.
     */
    public void escribir( int x )
    {
        valorAlmacenado = x;
    }
}

/**
 * variable privada que almacena el valor
 */
private int valorAlmacenado;
}
```

## Utilizar Javadoc con Eclipse





All Classes

[CeldaEntero](#)  
[TestCeldaEnte](#)

Package **Class** Use Tree Deprecated Index Help

PREV CLASS [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#)

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#) DETAIL: FIELD | [CONSTR](#) | [ME](#)

com.luisperezgomez.practica2\_2

# Class CeldaEntero

java.lang.Object  
└ **com.luisperezgomez.practica2\_2.CeldaEntero**

public class **CeldaEntero**  
extends java.lang.Object

Clase para simular una celda de memoria para un entero

All Classes

[CeldaEntero](#)  
[TestCeldaEntero](#)

**Package** Class Use Tree Deprecated Index He

PREV PACKAGE NEXT PACKAGE [FRAMES](#) [NO FRAMES](#)

Package

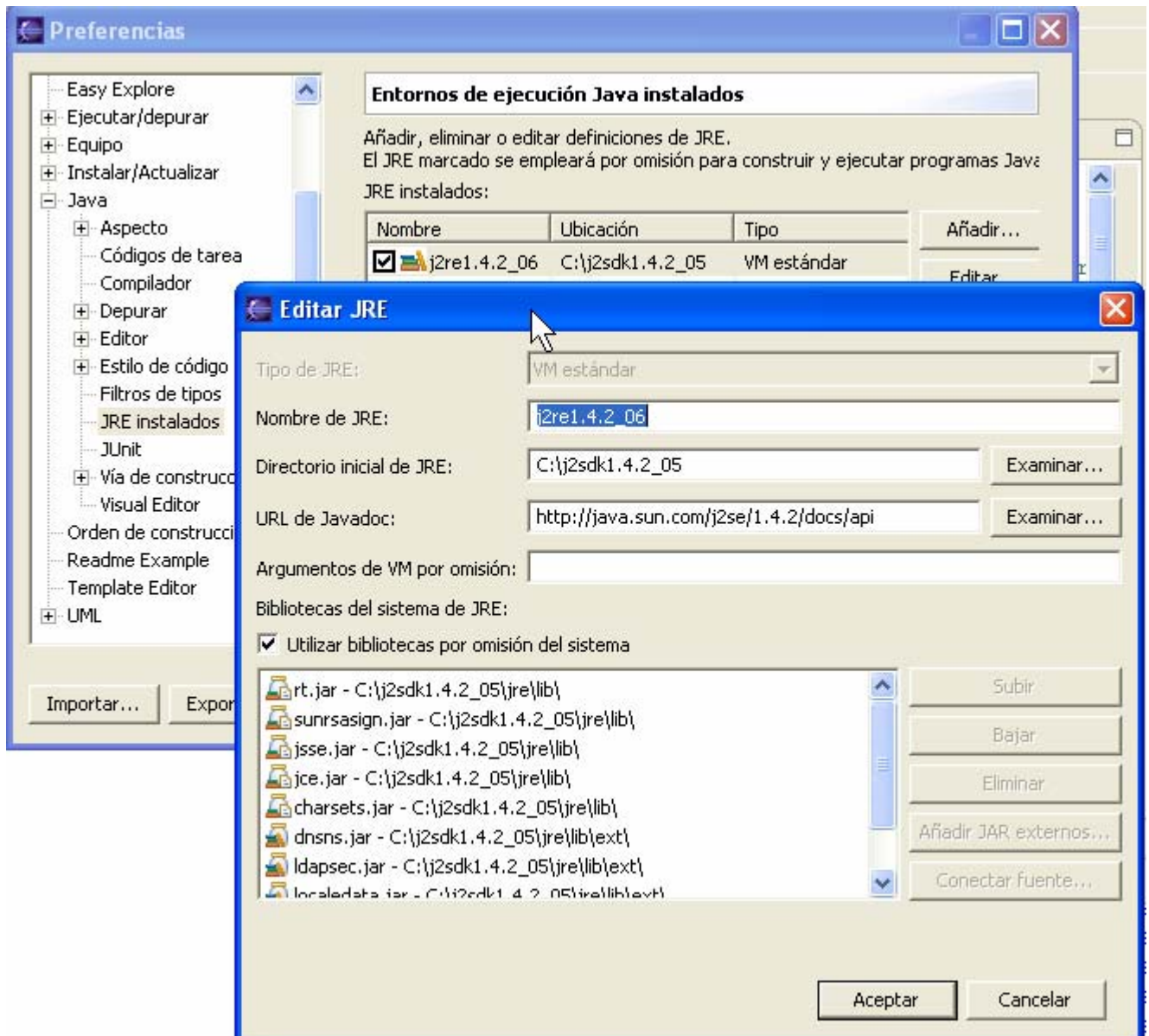
com.luisperezgomez.practica2\_2

## Class Summary

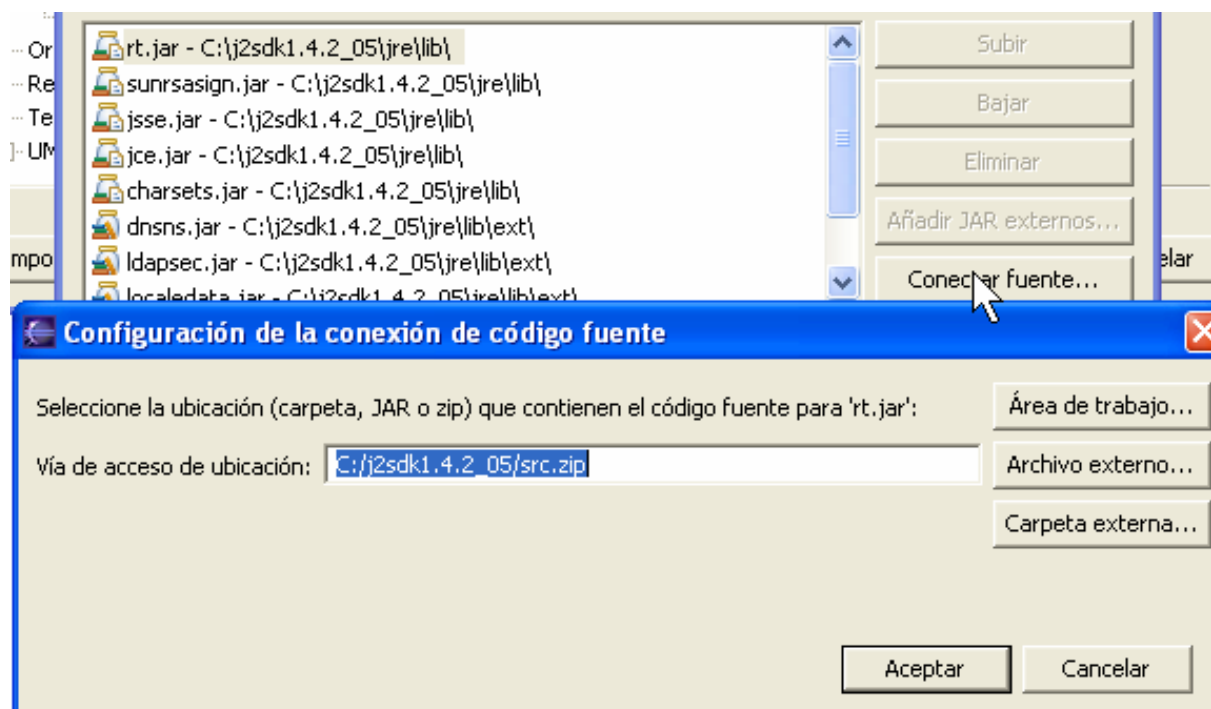
<a href="#">CeldaEntero</a>	Clase para simular una celda de memoria para un entero
<a href="#">TestCeldaEntero</a>	Clase para probar el funcionamiento de una celda de memoria para un entero

¿Qué resulta extraño en estas dos pantallas?

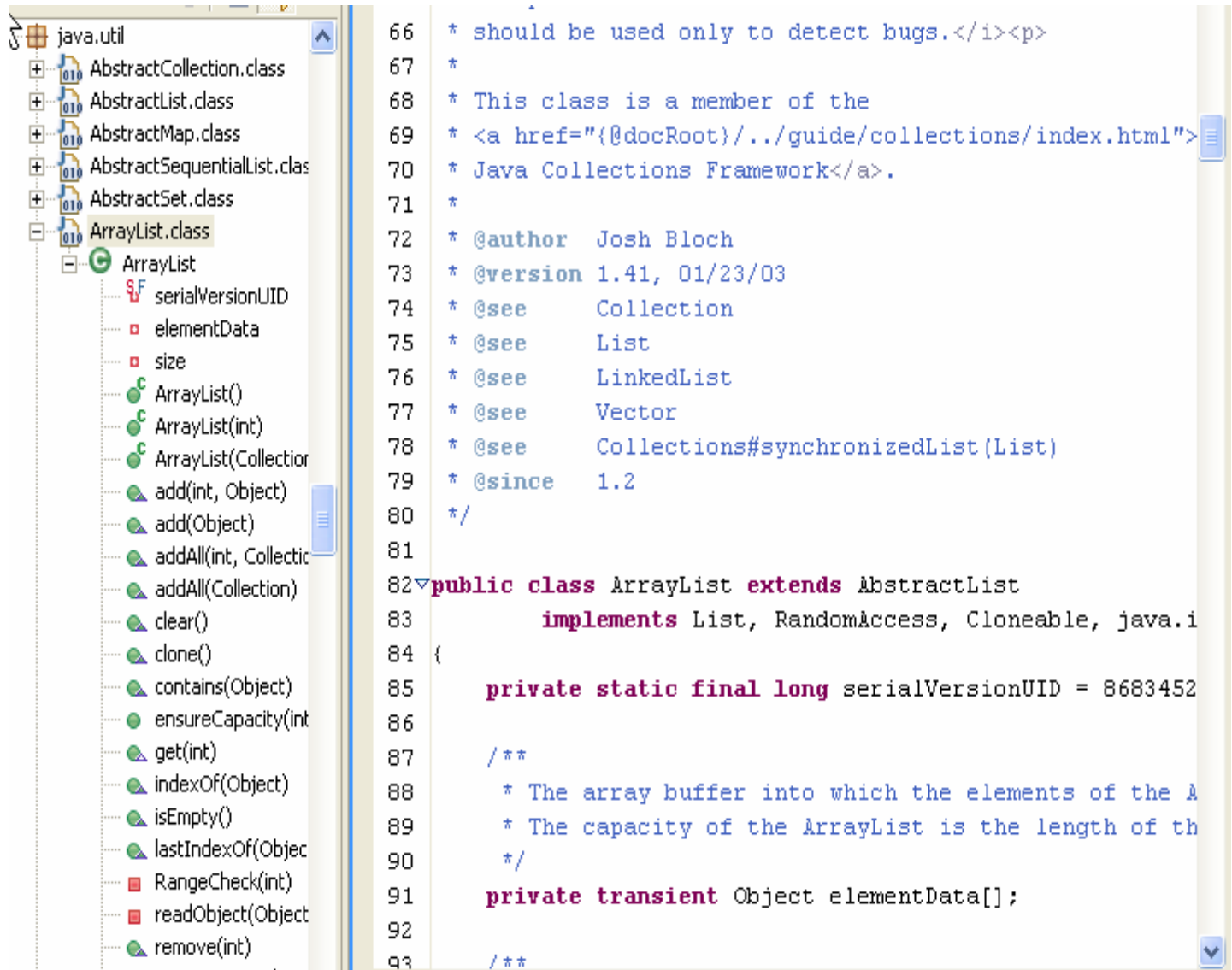
## Configurar el acceso a JavaDocs.



## Conectar código fuente



## Ejemplo de código fuente



The image shows a screenshot of an IDE with the source code of the `ArrayList` class in the `java.util` package. The left pane displays the package structure, and the right pane shows the code starting from line 66.

**Package Structure (Left Pane):**

- java.util
  - AbstractCollection.class
  - AbstractList.class
  - AbstractMap.class
  - AbstractSequentialList.class
  - AbstractSet.class
  - ArrayList.class (selected)
  - ArrayList
    - serialVersionUID
    - elementData
    - size
    - ArrayList()
    - ArrayList(int)
    - ArrayList(Collection)
    - add(int, Object)
    - add(Object)
    - addAll(int, Collection)
    - addAll(Collection)
    - clear()
    - clone()
    - contains(Object)
    - ensureCapacity(int)
    - get(int)
    - indexOf(Object)
    - isEmpty()
    - lastIndexOf(Object)
    - RangeCheck(int)
    - readObject(Object)
    - remove(int)

**Source Code (Right Pane):**

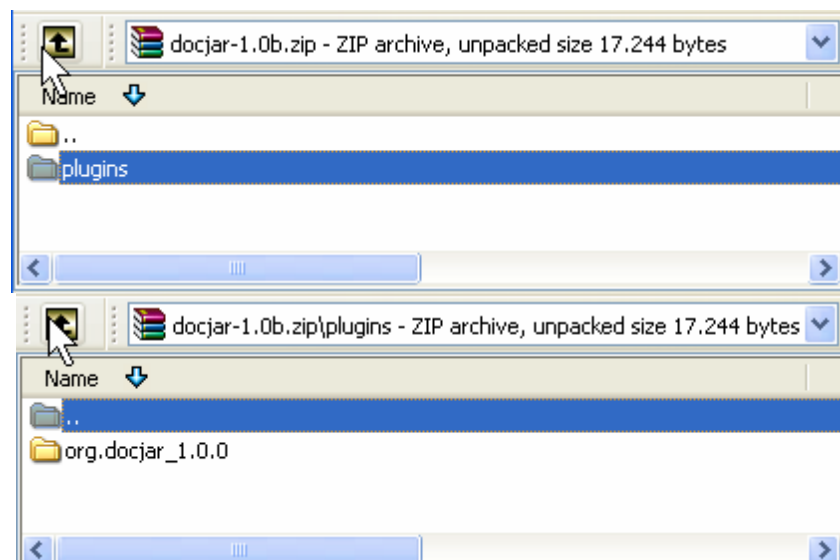
```
66  * should be used only to detect bugs.</i><p>
67  *
68  * This class is a member of the
69  * <a href="{@docRoot}/../guide/collections/index.html">
70  * Java Collections Framework</a>.
71  *
72  * @author   Josh Bloch
73  * @version  1.41, 01/23/03
74  * @see      Collection
75  * @see      List
76  * @see      LinkedList
77  * @see      Vector
78  * @see      Collections#synchronizedList(List)
79  * @since    1.2
80  */
81
82  public class ArrayList extends AbstractList
83      implements List, RandomAccess, Cloneable, java.io.Serializable
84  {
85      private static final long serialVersionUID = 8683452541172981881L;
86
87      /**
88       * The array buffer into which the elements of the ArrayList are stored.
89       * The capacity of the ArrayList is the length of the array.
90       */
91      private transient Object elementData[];
92
93      /**
```

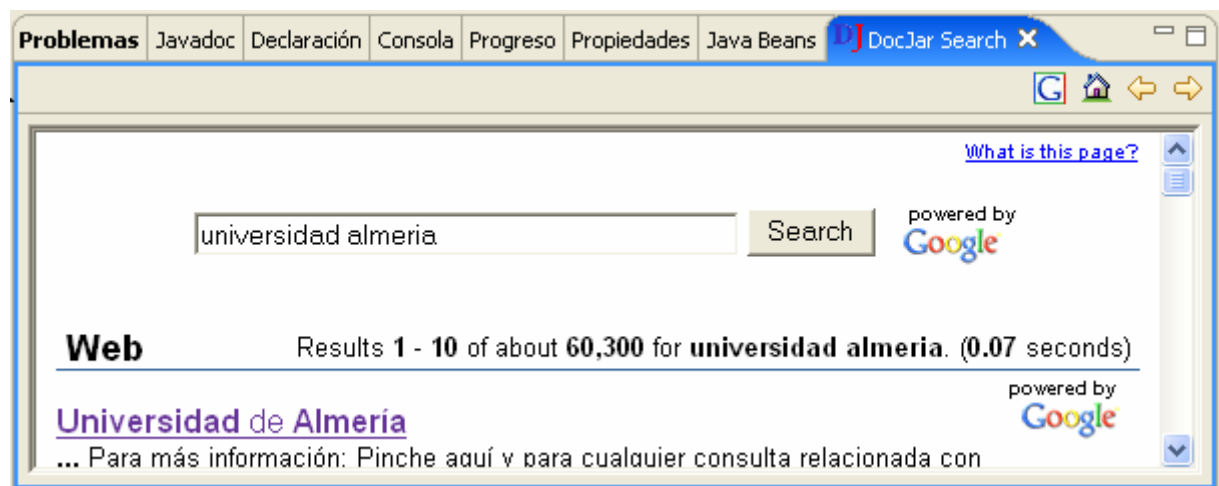
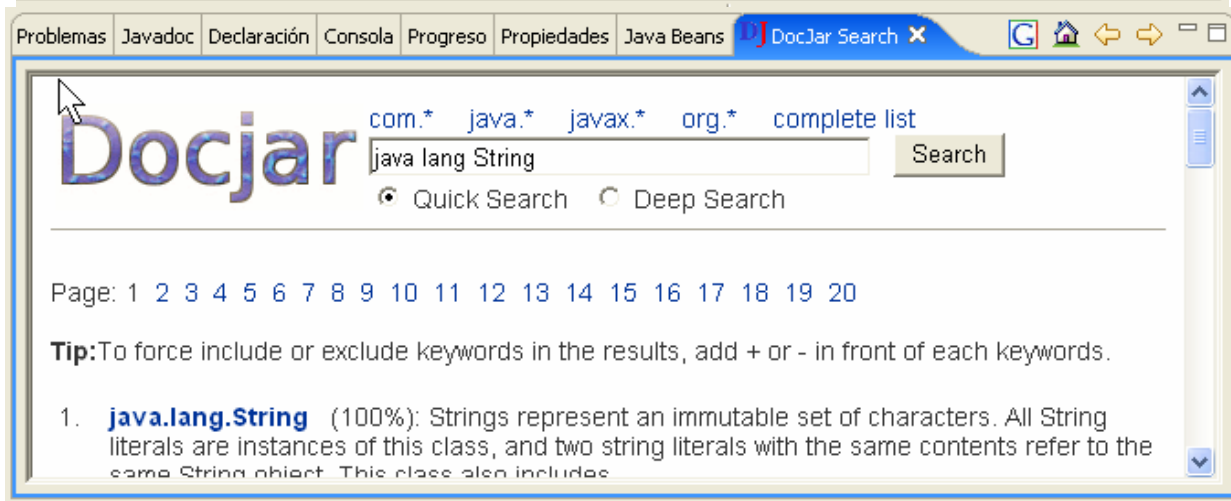
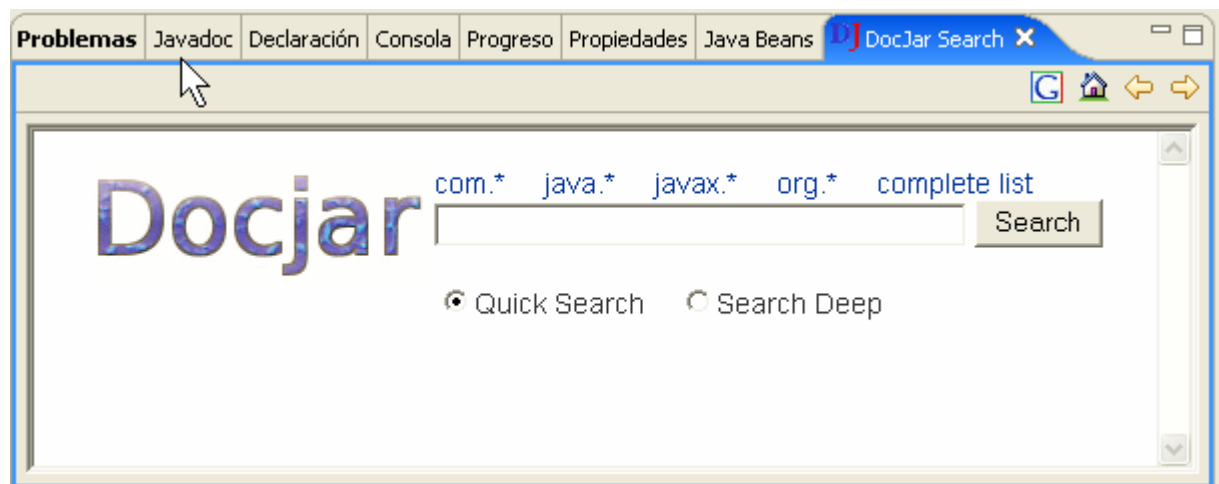
# Eclipse DocJar Google Plugin.



## Instalar un plugging

Basta con dejar caer la carpeta correspondiente sobre los directorios plugging, y en su caso features.





# Con el botón derecho del ratón sobre el texto

The screenshot shows an IDE window titled "PruebaFigura.java" with the following code:

```
43 private static Figura leerFigura( )
44 {
45     double rad;
46     double largo;
47     double ancho;
48     String unaLinea;
49
50     try
51     {
52         System.out.println( "Introduce el tipo de figura:" );
```

Below the code editor, the DocJar search results are displayed. The search bar shows "com.\* java.\* javax.\* org.\* complete list" and "Search". The results show the "Class" tab selected, with the path "Home >> All >> java >> lang". The class "String" is highlighted, and its implemented interfaces are listed: "CharSequence, Comparable, java.io.Serializable".

**Docjar** com.\* java.\* javax.\* org.\* complete list  
Search  
Quick Search Search Deep

Overview Package **Class** Use Deprecated Index

Home >> All >> java >> lang

SUMMARY: @ JAVADOC | SOURCE | DOWNLOAD | NESTED | FIELD | CONSTR | METHOD

PREV CLASS NEXT CLASS  
DETAIL: FIELD | CONSTR | METHOD

**java.lang**  
**Class String**

java.lang.Object  
└─ java.lang.String

**All Implemented Interfaces:**  
CharSequence, Comparable, java.io.Serializable

public final class **String**  
extends Object  
implements java.io.Serializable, Comparable, CharSequence