Bienvenidos

Herramientas de Desarrollo Java

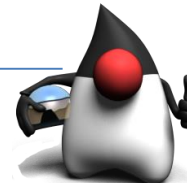
David R. Luna G.
davidrlunag@cibersys.com
0412-3111011

Temario del Curso

- ✓ Introducción al Lenguaje Java
- ✓ Estructura del Lenguaje Java
- ✓ **Herramientas de Desarrollo Java**
- ✓ Programación Orientada a Objetos con Java
- ✓ Manejo de Datos con Archivos
- ✓ Trabajando con Base de Datos Relaciones
- ✓ Desarrollando GUIs

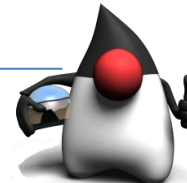
Herramientas de Desarrollo Java

Tool Name	Function
`javac`	The compiler for the Java programming language
`java`	The launcher for Java technology applications
`jdb`	The Java debugger
`javadoc`	The API document generator
`jar`	Java Archive (JAR) file creator and management tool



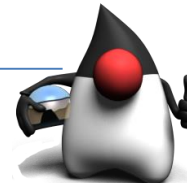
Herramientas de Desarrollo Java

Tool Category	Comments
Security tools	Implement security policies in applications
Internationalization tools	Enable applications to be localized
Remote method invocation (RMI) tools	Create (network) distributed applications
Common object request broker architecture (CORBA) tools	Create network applications that are based on CORBA technology
Java deployment tools	Support application deployment
Java Plug-in tools	Provide utilities for use with the Java Plug-in
Java web start tools	Used with Java web start technology



Herramientas de Desarrollo Java

Tool Category	Comments
Java Monitoring and Management (JMX) console	Used in conjunction with JMX technology
Java web services tools	Support Java web service application development
Experimental tools	Might not be available with future releases of the Java SE JDK

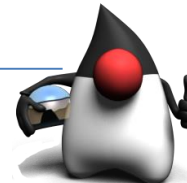


Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

Como se explicó anteriormente, en Java existen tres tipos de comentarios. Los primeros dos son // y /*.
El tercer tipo es llamado un comentario de documento.
Comienza con la sucesión de caracteres /** y termina con */.
Los comentarios de documentación le permiten insertar información acerca de su programa dentro de él mismo.

La idea es que utilicemos una herramienta que posee java llamada JAVADOC para extraer la información y ponerla dentro de un archivo HTML. Y el resultado será una documentación como el de la librería API de Java.

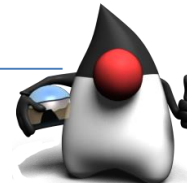


Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

La herramienta JavaDoc utiliza etiquetas especiales que deben ser encerradas entre comentarios de documentación. Estas etiquetas de documentación permiten que se genere una completa API, con el formato especificado en el código fuente. Estas etiquetas especiales inician con un @ y son sensitivas a minúsculas y mayúsculas. Una etiqueta de documentación debe iniciarse al principio de una línea o será tratado como texto normal.
Las etiquetas disponibles para la documentación son:

Etiqueta	Introducida JDK/SDK	Significado
@author	1.0	Identifica al autor de una clase
{@docRoot}	1.3	Especifica la ruta para el directorio raíz de la documentación actual
@deprecated	1.0	Especifica que una clase o miembro está desaprobado
@exception	1.0	Identifica una excepción lanzada por un método
{@link}	1.2	Inserta un vinculo en línea a otro tema
@param	1.0	Documenta los parámetros de un método
@return	1.0	Documenta un valor de retorno por un método
@see	1.0	Especifica un vinculo a otro tema
@serial	1.2	Documenta un campo seriado por defecto
@serialData	1.2	Documenta los datos escritos por los métodos writeObject() o writeExternal()
@serialField	1.2	Documenta un componente de ObjectStreamField
@since	1.1	Establece actualizaciones en la que se introdujo un cambio especifico
@throws	1.2	Lo mismo que @exception
@version	1.0	Especifica la versión de una clase



Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

@author

La etiqueta @author documenta el autor de una clase. Tiene la siguiente sintaxis:

@author *descripción*

Aquí, descripción será usualmente el nombre de la persona que escribió la clase. Puede necesitar especificar la opción –autor cuando vaya a ejecutar javadoc, con el fin de que el campo @author sea incluido en la documentación HTML.

Puede indicar varios autores con varias etiquetas @author, o puede colocar varios autores en la misma línea y JavaDoc separará la lista de autores con coma.

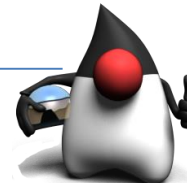
{@docRoot}

Especifica la ruta para el directorio raíz de la documentación actual. Es de gran ayuda cuando se desea incluir un archivo, tal como el logo de la empresa. Y tú deseas que exista una referencia desde todas las páginas generadas.

Esta etiqueta {@docRoot} puede ser utilizada directamente desde la línea de comando o en un comentario de documentación:

Desde la línea de comando:

```
javadoc -bottom '<a href="{@docRoot}/copyright.html">Copyright</a>'
```



Herramientas de Desarrollo Java

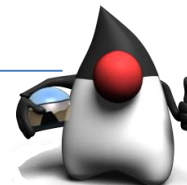
JavaDocs – Documentación en Java

@deprecated

La etiqueta @deprecated especifica que una clase o un miembro están desaprobados. Es recomendable que incluya un rotulo @see para informar al programador acerca de las alternativas disponibles. La sintaxis es la siguiente:
@deprecated *descripción*
donde descripción es el mensaje que describe la desaprobación.
Deberías incluir una etiqueta {@link}, que lleve al API que la reemplaza
Ejemplo:
/* @deprecated As of JDK 1.1, replaced by {@link @setBounds(int,int,int,int)} */.**

@exception

La etiqueta @exception describe una excepción para un método. Tiene la siguiente sintaxis:
@exception *nombreExcepcion descripción*
Hace el nombre totalmente calificado de la excepción es especificada por nombreExcepcion y descripción es una cadena que describe como puede ocurrir la excepción.
Esta etiqueta solo puede utilizarse en los métodos.



Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

{@link}

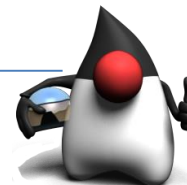
La etiqueta {@link} provee un hipervínculo en línea para información adicional. Tiene la siguiente sintaxis: {@link nombre texto}
Aquí, nombre es el nombre de la clase o el método al que el hipervínculo es añadido y texto es la cadena que se muestra. Esta etiqueta inicia y finaliza con llaves ({}), por lo que si necesitan utilizar llaves "}" dentro de la etiqueta, utiliza la notación HTML &#125.

@param

La etiqueta @param documenta un parámetro de un método. Tiene la siguiente sintaxis:
@param nombre-*parametro descripcion*
Aquí, nombre-parámetro especifica el nombre de parámetro a un método. El significado de ese parámetro es descrito por descripción. El rotulo @param puede ser utilizado solo en la documentación para un método.

@return

La etiqueta @return describe el valor de retorno de un método. Tiene la siguiente sintaxis:
@return *descripción*
Aquí descripción indica el tipo de valor devuelto por un método. Solo puede ser utilizado en la documentación de un método.



Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

@see

La etiqueta @see provee una referencia a información adicional. Aquí se muestran las formas mas comúnmente utilizadas:

@see ancla

@see pkg.clase#miembro texto

En la primera forma, ancla es el hipervínculo a una dirección URL absoluta o relativa.

En la segunda forma, pkg.clase#miembro especifica el numero del elemento y texto es el texto mostrado para ese elemento.

Por ejemplo:

@see "The Java Programming Language"

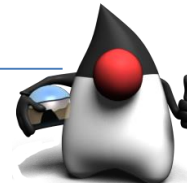
Esto genera:

See Also: "The Java Programming Language"

o@see Java Spec

Esto genera lo siguiente:

See Also: Java Spec



Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

@since

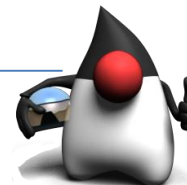
La etiqueta @since establece que una clase o miembro fue introducido en una versión especifica. Tiene la siguiente sintaxis:

@since actualización

Aquí actualización es una cadena para designar la actualización o versión en la que esa característica llego a estar disponible. La etiqueta @since puede ser utilizado en la documentación de variables, métodos y clases.

Por ejemplo:

@since 1.3



Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

@serial

La etiqueta @serial define el comentario para un campo seriado por defecto. Tiene la siguiente sintaxis:
@serial descripción
Aquí, descripción es el comentario para ese campo, si es necesario puede ocupar varias líneas.
La etiqueta @since debería ser añadida para cada campo serializable que haya sido adicionado a la versión inicial de una clase Serializable.

@throws

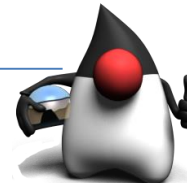
La etiqueta @throws tiene el mismo mecanismo que la etiqueta @exception.

@version

La etiqueta @version especifica la versión de una clase. Tiene la siguiente sintaxis:

@version información

Aquí información es una cadena que contiene la información de la versión, típicamente un número de versión, tal como 2.2. La etiqueta @version puede ser utilizado únicamente en la documentación de una clase. Usted puede necesitar especificar la opción –version cuando este ejecutando javadoc con el fin de que el campo @version sea incluido en la documentación HTML.



Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

Creando la Documentación

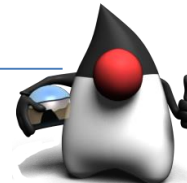
Una vez creado nuestro programa con la documentación, se debe proceder a llamar al documentador de Java con el comando javadoc.
Se puede crear la documentación para todo un paquete o varios, una clase o varias o una combinación.

Documentando uno o mas Paquetes

Para documentar un paquete, los archivos fuentes (*.java) para ese paquete deben estar localizados en un directorio que tenga elmismo nombre del paquete. Si un nombre de paquete tiene varias palabras separadas por punto (.) cada palabra representa un directorio diferente dentro del directorio principal. Todas las clases del paquete java.awt cdeben residir en un directorio llamado java\awt\. Sepuede ejecutar javadoc de dos maneras: cambiando de directorios o usando la opcion -sourcepath option. No se puede usar el * para indicar grupos de paquetes.

```
javadoc -d C:\home\html java.awt java.awt.event

javadoc -d C:\home\html -sourcepath C:\home\src
java.awt  java.awt.event
```

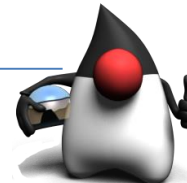


Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

```
import java.io.*;
/**
* Esta clase demuestra los comentarios de documentacion.
* @author David Luna
* @version 1.0
*/
public class javadocs{
/**
* Este Metodo devuelve el cuadrado de num
* Esta es una descripcion con multiples lineas.
* Puede utilizar tantas lineas como quiera.
* @param num el valor para hacer el cuadrado
* @return num al cuadrado
*/
public int cuadrado(int num){
        return(num *num);
}
```

```
/** Este metodo introduce un numero del usuario
* @return El valo ringresado es doble
* @exception IOException En error de Ingreso.
* @see IOException
*/
public int lectura() throws IOException{
        BufferedReader in=new BufferedReader(new
InputStreamReader(System.in));
        return(Integer.parseInt(in.readLine()));
}
}
```



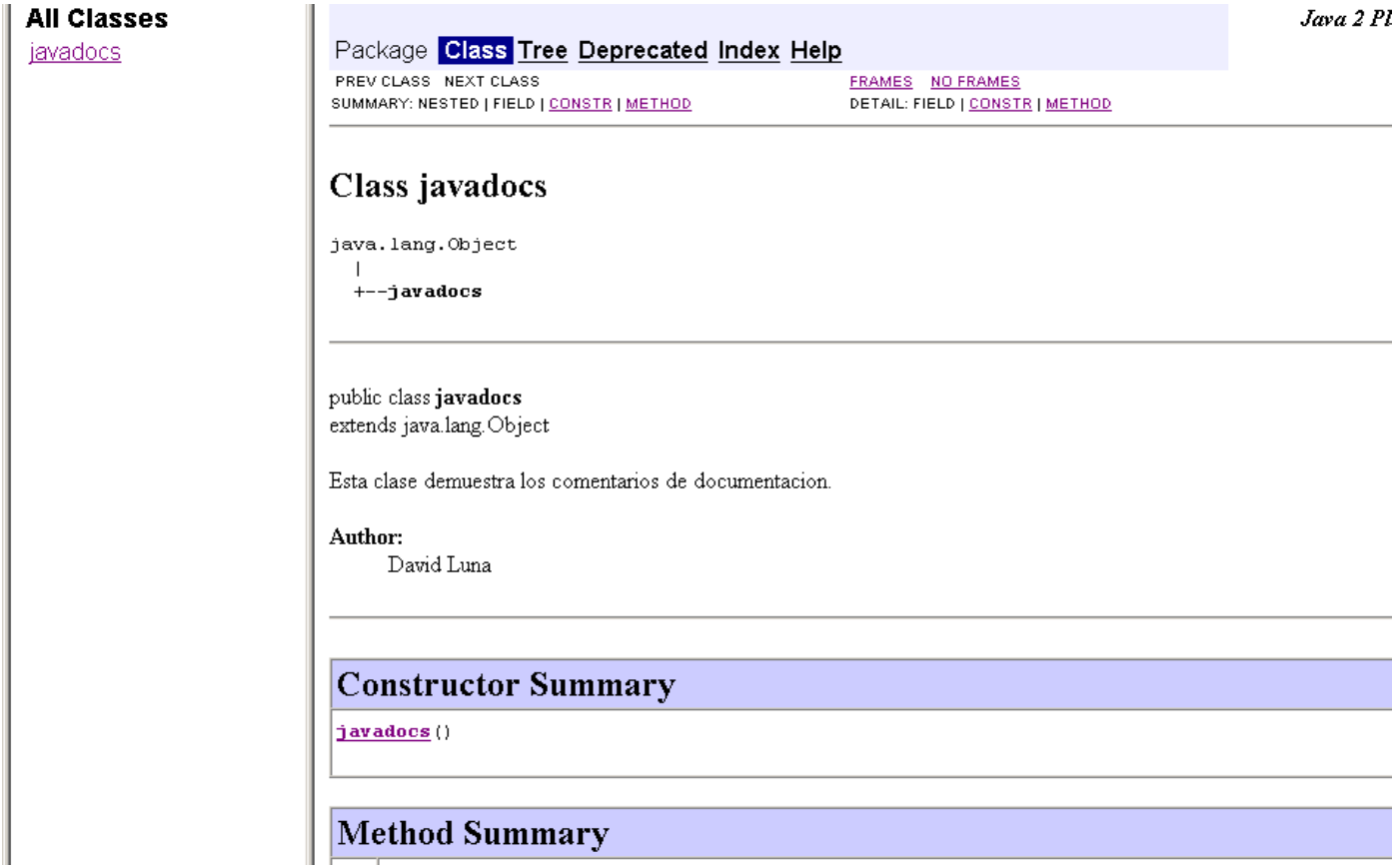
Herramientas de Desarrollo Java

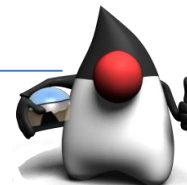
JavaDocs – Documentación en Java

El comando para crear la documentación será:

```
javadoc -d d:\manuales\cursos javadocs.java -
author -header "<b>Java 2 Platform
</b><br>v1.3" -footer "<b>CURSO DE
JAVA</b>“
```

La documentacion de resultado en formato HTML se visualizara de la siguiente manera:





Herramientas de Desarrollo Java

JavaDocs – Documentación en Java

El comando para crear la documentación será:

javadoc -d d:\manuales\cursos javadocs.java -author -header "Java 2 Platform
v1.3" -footer "CURSO DE JAVA“

La documentacion de resultado en formato HTML se visualizara de la siguiente manera:

All Classes

javadocs

int	**cuadrado**(int num) Este Metodo devuelve el cuadrado de num Esta es una descripcion con multiples lineas.
int	**lectura**() Este metodo introduce un numero del usuario

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

javadocs

```
public javadocs()
```

Method Detail

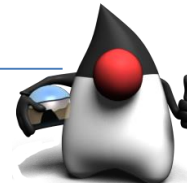
cuadrado

```
public int cuadrado(int num)
```

Este Metodo devuelve el cuadrado de num Esta es una descripcion con multiples lineas. Puede utilizar tantas lineas como quiera.

Parameters:
num - el valor para hacer el cuadrado

Returns:
num al cuadrado



Herramientas de Desarrollo Java

Jar – Haciendo un Ejecutable