As explained in Module 1, Java supports three types of comments. The first two are the // and the /* */. The third type is called a *documentation comment.* It begins with the character sequence /**. It ends with */. Documentation comments allow you to embed information about your program into the program itself. You can then use the **javadoc** utility program (supplied with the JDK) to extract the information and put it into an HTML file. Documentation comments make it convenient to document your programs. You have almost certainly seen documentation generated with **javadoc**, because that is the way the Java API library was documented by Sun.

The javadoc Tags

The **javadoc** utility recognizes the following tags:

Tag	Meaning
@author	Identifies the author of a class.
{@code}	Displays information as-is, without processing HTML styles, in code font. (Added by J2SE 5.)
@deprecated	Specifies that a class or member is deprecated.
{@docRoot}	Specifies the path to the root directory of the current documentation.
@exception	Identifies an exception thrown by a method.
{@inheritDoc}	Inherits a comment from the immediate superclass.
{@link}	Inserts an in-line link to another topic.
{@linkplain}	Inserts an in-line link to another topic, but the link is displayed in a plain-text font.
{@literal}	Displays information as-is, without processing HTML styles. (Added by J2SE 5.)
@param	Documents a method's parameter.
@return	Documents a method's return value.
@see	Specifies a link to another topic.
@serial	Documents a default serializable field.
@serialData	Documents the data written by the **writeObject( )** or **writeExternal( )** methods.
@serialField	Documents an **ObjectStreamField** component.
@since	States the release when a specific change was introduced.
@throws	Same as **@exception**.
{@value}	Displays the value of a constant, which must be a **static** field.
@version	Specifies the version of a class.

Document tags that begin with an "at" sign (@) are called *stand-alone* tags, and they must be used on their own line. Tags that begin with a brace, such as **{@code}**, are called *in-line* tags, and they can be used within a larger description. You may also use other, standard HTML tags in a documentation comment. However, some tags such as headings should not be used, because they disrupt the look of the HTML file produced by **javadoc**.

You can use documentation comments to document classes, interfaces, fields, constructors, and methods. In all cases, the documentation comment must immediately precede the item being documented. When you are documenting a variable, the documentation tags you can use are **@see**, **@since**, **@serial**, **@serialField**, **{@value}**, and **@deprecated**. For classes, you can use **@see**, **@author**, **@since**, **@deprecated**, **@param**, and **@version**. Methods can be documented with **@see**, **@return**, **@param**, **@since**, **@deprecated**, **@throws**, **@serialData**, **{@inheritDoc}**, and **@exception**. A **{@link}**, **{@docRoot}**, **{@code}**, **{@literal}**, or **{@linkplain}** tag can be used anywhere. Each tag is examined next.

@author

The **@author** tag documents the author of a class. It has the following syntax:

@author *description*

Here, *description* will usually be the name of the person who wrote the class. The **@author** tag can be used only in documentation for a class. You will need to specify the **-author** option when executing **javadoc** in order for the **@author** field to be included in the HTML documentation.

{@code}

The **{@code}** tag enables you to embed text, such as a snippet of code, into a comment. That text is then displayed as-is in code font, without any further processing such as HTML rendering. It has the following syntax:

{@code *code-snippet*}

@deprecated

The **@deprecated** tag specifies that a class or a member is deprecated. It is recommended that you include **@see** or **{@link}** tags to inform the programmer about available alternatives. The syntax is the following:

@deprecated *description*