

JAVADOC SUMMARY

Documentation Comments

A documentation comment is delimited by `/**` and `*/`. You can comment

- Classes
- Methods
- Instance variables

Each comment is placed *immediately above* the feature it documents.

Each `/** . . . */` documentation comment contains introductory text followed by tagged documentation. A tag starts with an `@` character, such as `@author` or `@param`. Tags are summarized in Table 1. The *first sentence* of the introductory text should be a summary statement. The javadoc utility automatically generates summary pages that extract these sentences.

You can use HTML tags such as `em` for emphasis, `code` for a monospaced font, `img` for images, `ul` for bulleted lists, and so on.

Table 1 Common javadoc Tags

Tag	Description
<code>@param</code> <i>parameter explanation</i>	A parameter of a method. Use a separate tag for each parameter.
<code>@return</code> <i>explanation</i>	The return value of a method.
<code>@throws</code> <i>exceptionType explanation</i>	An exception that a method may throw. Use a separate tag for each exception.
<code>@deprecated</code>	A feature that remains for compatibility but that should not be used for new code.
<code>@see</code> <i>packageName.ClassName</i> <code>@see</code> <i>packageName.ClassName</i> <i>#methodName(Type₁, Type₂, . . .)</i> <code>@see</code> <i>packageName.ClassName#variableName</i>	A reference to a related documentation entry.
<code>@author</code>	The author of a class or interface. Use a separate tag for each author.
<code>@version</code>	The version of a class or interface.

Here is a typical example. The summary sentence (in color) will be included with the method summary.

```
/**
    Withdraws money from the bank account. Increments the
    transaction count.
    @param amount the amount to withdraw
    @return the balance after the withdrawal
    @throws IllegalArgumentException if the balance is not sufficient
*/
public double withdraw(double amount)
{
    if (balance - amount < minimumBalance)
    {
        throw new IllegalArgumentException();
    }
    balance = balance - amount;
    transactions++;
    return balance;
}
```

Generating Documentation from Commented Source

To extract the comments, run the javadoc program:

```
javadoc [options] sourceFile1|packageName1|@fileList1
        sourceFile2|packageName2|@fileList2 . . .
```

See the documentation of the javac command in Appendix G for an explanation of file lists. Commonly used options are summarized in Table 2.

To document all files in the current directory, use (all on one line)

```
javadoc -link http://download.oracle.com/javase/7/docs/api -d docdir *.java
```

Table 2 Common javadoc Command Line Options

Option	Description
-link <i>URL</i>	Link to another set of javadoc files. You should include a link to the standard library documentation, either locally or at http://download.oracle.com/javase/7/docs/api .
-d <i>directory</i>	Store the output in <i>directory</i> . This is a useful option, because it keeps your current directory from being cluttered up with javadoc files.
-classpath <i>locations</i>	Look for classes on the specified paths, overriding the CLASSPATH environment variable. If neither is specified, the current directory is used. Each <i>location</i> is a directory, JAR file, or ZIP file. Locations are separated by a platform-dependent separator (: Unix, ; Windows).
-sourcepath <i>locations</i>	Look for source files on the specified paths. If not specified, source files are searched in the class path.
-author, -version	Include author, version information in the documentation. This information is omitted by default.