**Chapter 10. Documentation with Javadoc**

**Table of Contents**

[Writing Javadoc Comments](http://www.drjava.org/docs/user/ch10.html#javadoc-writing)

[How to Use Javadoc in DrJava](http://www.drjava.org/docs/user/ch10s02.html)

[Java API Javadoc](http://www.drjava.org/docs/user/ch10s03.html)

Documenting your code is crucial to help others understand it, and even to remind yourself how your own older programs work. Unfortunately, it is easy for most external documentation to become out of date as a program changes. For this reason, it is useful to write documentation as comments in the code itself, where they can be easily updated with other changes. Javadoc is a documentation tool which defines a standard format for such comments, and which can generate HTML files to view the documentation from a web broswer. (As an example, see Oracle's Javadoc documentation for the Java libraries at <http://download.oracle.com/javase/6/docs/api/>.)

You can easily run Javadoc over your programs from within DrJava, using the "Javadoc All Documents" and "Preview Javadoc for Current Document" commands in the Tools menu. These commands will generate Javadoc HTML files from the comments you have written and display them in a browser. This chapter gives a brief overview of these commands and how to write Javadoc comments. More detailed information on writing Javadoc comments can be found online at <http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html>.

**Writing Javadoc Comments**

In general, Javadoc comments are any multi-line comments ("/\*\* ... \*/") that are placed before class, field, or method declarations. They must begin with a slash and two stars, and they can include special tags to describe characteristics like method parameters or return values. The HTML files generated by Javadoc will describe each field and method of a class, using the Javadoc comments in the source code itself. Examples of different Javadoc comments are listed below.

**Simple Comments.**Normal Javadoc comments can be placed before any class, field, or method declaration to describe its intent or characteristics. For example, the following simple Student class has several Javadoc comments.

/\*\*

\* Represents a student enrolled in the school.

\* A student can be enrolled in many courses.

\*/

public class Student {

/\*\*

\* The first and last name of this student.

\*/

private String name;

/\*\*

\* Creates a new Student with the given name.

\* The name should include both first and

\* last name.

\*/

public Student(String name) {

this.name = name;

}

}

**Using Tags.**Tags can be used at the end of each Javadoc comment to provide more structured information about the code being described. For example, most Javadoc comments for methods include "@param" and "@return" tags when applicable, to describe the method's parameters and return value. The "@param" tag should be followed by the parameter's name, and then a description of that parameter. The "@return" tag is followed simply by a description of the return value. Examples of these tags are given below.

/\*\*

\* Gets the first and last name of this Student.

\* @return this Student's name.

\*/

public String getName() {

return name;

}

/\*\*

\* Changes the name of this Student.

\* This may involve a lengthy legal process.

\* @param newName This Student's new name.

\* Should include both first

\* and last name.

\*/

public void setName(String newName) {

name = newName;

}

Other common tags include "@throws e" (to describe some Exception "e" which is thrown by a method) and "@see #foo" (to provide a link to a field or method named "foo").