How to Write Doc Comments for the "Javadoc" Tool

Format of a Doc Comment

A doc comment is written in HTML and must **precede** a class, field, constructor or method declaration. It is made up of two parts - a **description** followed by **block tags**. In this example, the javadoc comment for the method getImage is shown. The block tags used are **@param**, **@return**, @link and @see.

Example

```
/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
 * argument is a specifier that is relative to the url argument.
 * <p>
 * This method always returns immediately, whether or not the
 * image exists. When this applet attempts to draw the image on
 * the screen, the data will be loaded. The graphics primitives
 * that draw the image will incrementally paint on the screen.
 *
 * @param  url  an absolute URL giving the base location of the image
 *         name the location of the image, relative to the url argument
 * @return     the image at the specified URL
 * @see        Image
 */
public Image getImage(URL url, String name) {
            return getImage(new URL(url, name));
}
```

Notes:

- Each line above is indented to align with the code below the comment.
- The first line contains the begin-comment delimiter (/**).
- The leading asterisks in front of every line are optional.
- The first sentence is a short summary of the method, as Javadoc automatically places it in the method summary table (and index).
- If you have more than one paragraph in the doc comment, separate the paragraphs with a paragraph tag, as shown. Note that the tag does not need to be closed with .
- Insert a blank comment line between the description and the list of tags, as shown.
- The first line that begins with an "@" character (in this example @param) ends the description. There is only one description block per doc comment; you cannot continue the description following block tags.
- The last line contains the end-comment delimiter (*/) Note that unlike the begin-comment delimiter, the end-comment contains only a **single** asterisk.

The resulting HTML javadoc comment page for the above example looks like this:

---

getImage

public Image **getImage**(URL url,
String name)

Returns an Image object that can then be painted on the screen. The url argument must specify an absolute URL. The name argument is a specifier that is relative to the url argument.

This method always returns immediately, whether or not the image exists. When this applet attempts to draw the image on the screen, the data will be loaded. The graphics primitives that draw the image will incrementally paint on the screen.

Parameters:

url - an absolute URL giving the base location of the image

name - the location of the image, relative to the url argument

Returns:

the image at the specified URL

See Also:

Image

TIP: Lines won't wrap, limit any doc-comment lines to 80 characters!

Descriptions

First Sentence

The first sentence of each doc comment should be a summary sentence, containing a concise but complete description of the API item. This means the first sentence of each member, class, interface or package description. The Javadoc tool copies this first sentence to the appropriate member, class/interface or package summary. The comment must preferably end with a period and should be aligned with the constructor of the corresponding method. For example:

```
/**
 * This method returns the first name.
 * @return          The first name in the database
 */
   public static String getFirstName () {
       return first;
   }
```

Order of Tags

Include tags in the following order:
* @author (classes and interfaces only, required)
* @version (classes and interfaces only, required)
* @param (methods and constructors only)
* @return (return methods only)
* @exception/@throws (any method that throws any exception)

Required Tags

An @param tag is required for every parameter, even when the description is obvious. The @return tag is required for every method that returns something other than void, even if it is redundant with the method description. (Whenever possible, find something non-redundant (ideally, more specific) to use for the tag comment.)

Tag Comments Examples

@author John Smith
@version 1.40, Jan 29,2017
@param <variable name> <variable description>
e.g @param title The title of the window
@return brief description of the return variable
e.g. @return The first name of the individual.
@throws/@exception <exception> <reason>
e.g. @throws/@exception NumberFormatException If an invalid integer has been detected

There are a few more such as @see, @deprecated, @serial, @since, etc. which are of not much use for this course.

Once all class files are complete, the javadoc can be generated by the following methods (assuming a java SDK has been installed and configured):

Command Line:
For one or more files:

```
cd x:\path\to\java\files
javadoc -d x:\path\to\generate File1.java File2.java File3.java
```

NetBeans/Sun ONE Studio:
For one or more files:
Right click on the folder containing the sources, point to **Tools-> Generate Javadoc.**

Additional resources on the javadoc tool:

http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html