1. **What ant task do we use to generate Javadoc in this course?**

$ ant jdoc

1. **What is the Javadoc comment format?**

* **What is the first sentence of a Javadoc comment used for? Where does it appear in the generated html?**

The first sentence is a summary sentence, containing a concise but complete description of the API item. It appears in the summary section of the generated html.

* **How should paragraphs be separated?**

Paragraphs should be separated using the paragraph html tag <p> on its own line.

* **Should blank lines be used? If so, where/why?**

Yes, blank lines should be used to separate the description and the list of tags. A blank line can also optionally be used to separate groups of similar tags from the other tags.

* **Where does a method or class description end?**

A method or class description ends at the first @ symbol (i.e. the first tag).

1. **How are these tags used? When are they used? What is the proper format or structure?**

|  |  |  |
| --- | --- | --- |
| **Tag** | **Usage** | **Format** |
| @author | Used in a class or interface comment and describes an author(s). | @author author-name |
| @param | Used in method comment and describes a method’s parameter(s). | @param name description |
| @return | Used in a method comments and describes the method’s return value. | @return description |
| @throws | Used in a method comment and describes any exception that may be thrown from the method. | @throws classname description |
| @see | Used in a class, interface, field or method comment and provides a link to another element of the document. | @see reference |
| @since | Used in a class, interface, field or method comment and describes when this functionality has first existed. | @since version |

1. **Demonstrate use of html in Javadoc.**

One use for html in Javadoc is to separate paragraphs using the <p> tag. For example:

This is the first paragraph.

<p>

This is the second paragraph.

Another use is creating a list, ordered or unordered. For example:

<ul>

<li> Line one

<li> Line two

<li> Line three

</ul>

1. **When are comments inherited?**

There are three cases where comments are inherited in Javadoc:

* When a method in a class overrides a method in a superclass.
* When a method in an interface overrides a method in a superinterface.
* When a method in a class implements a method in an interface.

If there are no comments for the method, the Javadoc tool will copy the text of the method it’s overriding or implementing to the generated documentation for the method. If the method does have comments, a subheading with “specified by” or “Overrides” will be generated.