JavaDoc

Ashley J.S Mills

<ashley@ashleymills.com>

Table of Contents

1. JavaDoc

JavaDoc is a tool which extracts information from a Java source file to create an API. It is specifically oriented toward this kind of documentation. To quote *How to Write Doc Comments for the JavadocTM Tool*, an API should describe "all aspects of the behavior of each method on which a caller can rely". One generally uses JavaDoc to document things such as *Class*es, *Interface*s and *Method*s but one can theoretically use it for anything written in Java by defining custom tags and creating custom *DocLet*s.

JavaDoc is written within the Java source code for a particular project and then the **javadoc** tool is used to extract the JavaDoc-marked-up sections and create the HTML files that comprise the JavaDoc output.

2. Installation

JavaDoc comes with the J2SE (Java 2 Platform Standard Edition), available from http://java.sun.com/j2se/. You should be capable of following the instructions that come with this to install it; see *Configuring A Windows Working Environment* and *Configuring A Unix Working Environment*.

3. JavaDoc Usage

3.1. General Format

The general format of a JavaDoc marked up section is:

```
/**
 * This is the description part of the doc comment.
 *
 * Additional details
 *
 * @tag1 Tag Content
 * @tag2 Tag Content
```

```
 *    .
 *    .
 *    .
 */
```

Take notice of the spacing used, the tags section must be separated from the description by a single blank line. The content of the comment starts one space in from the left hand edge of the comment. Tag lists may be separated into logical blocks by inserting a blank line. Lines should preferably be below 80 columns in width. New paragraphs can be specified by using the *<p>* tag.

The first sentence should summarize the overall behaviour of the Class, Interface or Method being described. It should be terminated with a full stop followed by either a space, tab or newline, it may span more than one line. To include multiple full stops in the first sentence, make sure that the ones that do not signify the end of the first sentence are not followed by a space, tab or newline. This can be achieved by using the HTML entity *&nsbp;*:

```
/**
 * This is the part of the first sentence.&nsbp;This is still part of the first sentence. Thi
 *
 */
```

One should write from the perspective of the third person:

```
/**
 * Correct   - Returns an Integer representing the mean of the numbers provided as parameters
 *
 *             It has the special feature of blah blah blah...
 *
 * Incorrect - This method returns an Integer representing the mean of the numbers you provid
 *
 *             This has the special feature of blah blah blah...
 */
```

Tag inclusion should accord to the following order:

* @author (classes and interfaces only, required)
* @version (classes and interfaces only, required)
*
* @param (methods and constructors only)
* @return (methods only)
* @exception (@throws is a synonym added in Javadoc 1.2)
* @see
* @since
* @serial (or @serialField or @serialData)
* @deprecated

When using Java code in a description, one should use <code>blah</code>:

```
 *
 * Returns <code>true</code> if blah otherwise returns <code>false</code>
 *
```

A description may contain any HTML tags:

```
/**
 * This is the <strong>description</strong> part of the doc comment.
```

```
 * <p>
 *   blah blah blah blah blah
 *   blah blah <b>blah</b> <i>blah</i>blah
 * </p>
 * <pre>
 *     blah
 * blah    blah
 * </pre>
 *
 * @author Ashley <strong>Mills</strong>
 * @version 1.2
 */
public class test {
         .
         .
         .
```

Figure 1. The HTML output from the listing above

---

public class **test**
extends Object

This is the **description** part of the doc comment.

blah blah blah blah blah blah blah **blah** *blah*blah

```
     blah
 blah    blah
```

3.2. Using javadoc To Produce The Output

The command **javadoc** is used to generate the JavaDoc output:

`javadoc blah.java`

Creates JavaDoc for the file `blah.java`.

`javadoc *.java`

Creates JavaDoc for all `java` files in the current directory.

`javadoc -help`

Displays the command line options (so does not providing any).

JavaDoc does not automatically link to the real online API, to achieve this one has to utilise the `-link` option provided by StandardDoclet. Assuming one desires to link to the Java 1.4 API, one would use:

```
javadoc -link http://java.sun.com/j2se/1.4/docs/api *.java
```

This causes all references to Java 1.4 classes in the generated HTML to be resolved.

Tip

Use the `-d` option to direct generated JavaDoc to a specific directory.

3.3. @author

@author *`author name`*

Specifies the author of the class or interface:

```
 *
 * @author Jimmy Petronas
 * @author Mickey Block
 * @author Sally Finch
 *
```

Not included in generated HTML using standard Doclet, unless turned on by specifying `-author` option when running **javadoc**

3.4. @version

@version *`version information`*

Adds version information to a class.

```
*
* @version 1.2
*
```

Only one per class or interface allowed. Not included in generated HTML using standard Doclet, unless turned on by specifying the `-version` option when running **javadoc**.

3.5. @param

@param parameter-name description

Adds a parameter to the parameter list.

```
/**
 * Returns the product of two integers.
 *
 * @param int operand one
 * @param int operand two
 * @return an int
 */
public int mul(int a, int b) {
   return a*b;
}
```

Figure 2. The HTML output from the listing above

Method Summary	
int	**mul**(int a, int b) Returns the product of two integers.

Method Detail

mul

```
public int mul(int a,
               int b)
```

Returns the product of two integers.

Returns:
an int

3.6. @return

@return *description of return value*

Describes the value returned from a method.

```
*
* @return A new BlahBlah Object with a field size of 100.
*
```

Figure 3. The HTML output for the @return above

Returns:
A new BlahBlah Object with a field size of 100.

3.7. @exception

@exception *class-name description*

Describes the exceptions thrown by the constructor, method, class or interface. *class-name* is the name of the exception.

```
/**
 * Replaces test().
 * @throws BlahException unless blah blah blah
 * @exception BlahBlahException
 */
public test(int i) {
}
```

Figure 4. The HTML output for the listing above

test

```
public test(int i)
```

Replaces test().

Throws:
BlahException - unless blah blah blah
BlahBlahException

3.8. @see

@see *classname*

Adds a hyperlink to the section referenced by the classname supplied.

```
/**
 * Returns the product of the squares of a and b.
 *
 * @see #mul
 *
 * @param int operand one
 * @param int operand two
 * @return an int
 */
   public int squaredMul(int a, int b) {
      return mul(mul(a,a),mul(b,b));
   }
```

Figure 5. The HTML output for listing above

---

squaredMul

```
public int squaredMul(int a,
                      int b)
```

Returns the product of the squares of a and b.

Returns:
an int
See Also:
mul(int, int)

The classname can be local like this example, prefixed with a hash '#' or a fully qualified class name like*java.lang.String* or *java.lang.String#charAt*.

3.9. @since

@since *since-text*

Specifies from which version of Java this class, method or interface has been available from.

```
*
* @since JDK1.4
*
```

Figure 6. The HTML output for listing above

Since:
JDK1.4

3.10. @deprecated

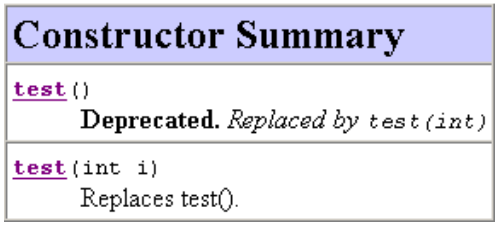
@deprecated *deprecated-text*

Marks a class, interface or method as deprecated.

```
/**
 * @deprecated Replaced by <code>test(int)</code>
 * @see #test(int)
 */
public test() {
}

/**
 * Replaces test().
 */
public test(int i) {
}
```

Figure 7. The HTML output for listing above



3.11. More

More detail pertaining to the available tags can be found at the JavaDoc reference page: http://java.sun.com/j2se/1.4/docs/tooldocs/windows/javadoc.html.

4. Simple Example

Here is a simple example showing a few of the features of JavaDoc:

```
/**
 * Represents a "fuzzy" integer, in that the integer is
 * not exact, but is instead specified as being between a range.
 *
 * @author Ashley Mills
 * @version 1.0b
 */
public class FuzzyInteger {
   private int lowerVal, upperVal;

   /**
    * Creates a new <code>FuzzyInteger</code> object based on the
    * range specified.
    *
    * @param lowerVal The lowest value that the integer could be.
    * @param upperVal The highest value that the integer could be.
    */
   public FuzzyInteger(int lowerVal, int upperVal) {
      this.lowerVal = lowerVal;
      this.upperVal = upperVal;
   }

   /**
    * Doubles the value of the <code>FuzzyInteger</code>.
    */
   public void doubleFuzzy() {
      lowerVal = lowerVal+lowerVal;
      upperVal = upperVal+upperVal;
   }
}
```

Figure 8. The HTML output for listing above

Class FuzzyInteger

java.lang.Object
  |
  +--**FuzzyInteger**

public class **FuzzyInteger**
extends Object

Represents a "fuzzy" integer, in that the integer is not exact, but is instead specified as being between a range.

Constructor Summary

FuzzyInteger(int lowerVal, int upperVal)	Creates a new FuzzyInteger object based on the range specified.

Method Summary

void	**doubleFuzzy**() Doubles the value of the FuzzyInteger.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

FuzzyInteger

```
public FuzzyInteger(int lowerVal,
                    int upperVal)
```

Creates a new FuzzyInteger object based on the range specified.

Parameters:

lowerVal - The lowest value that the integer could be.
upperVal - The highest value that the integer could be.

Method Detail

doubleFuzzy

```
public void doubleFuzzy()
```

Doubles the value of the FuzzyInteger.

5. References (And Links You May Find Useful)

- http://java.sun.com/j2se/javadoc/writingdoccomments/index.html

  How to Write Doc Comments
  for the JavadocTM Tool
  Maintained by javadoc-tool@sun.com

- http://java.sun.com/j2se/1.4/docs/tooldocs/solaris/javadoc.html

  JavaDoc reference page

- http://java.sun.com/j2se/javadoc/faq/index.html

  JavaDoc FAQ