

Introduction to Programming Using Java

605.201

Creating Documentation With `javadoc`

By now you already know how to include source code comments in your Java programs using `/* ... */` and `//`. There is a third way that can be used to produce formal documentation for your programs. This involves using something called *documentation comments* in conjunction with a tool called `javadoc` that comes with the JDK.

To use the `javadoc` tool you must add special comments, called documentation comments (also called `javadoc` comments) into the source code using a specific syntax, as illustrated below.

```
/**
 * This class computes and displays the sum of two integers
 * using a class method called sum().
 * @author The Java Master
 * @version 1.0
 * @param firstValue the first value to be summed
 * @param secondValue the second value to be summed
 */
```

@author gives the name of the author
@version gives the version number of the code
@param gives the name & description of a parameter used by the code
@return gives a description of the value returned by a method

A `/**` starts a documentation comment and a `*/` ends a documentation comment. Documentation comments may also include things called *tags*. Each tag starts with an `@` symbol followed by a tag name. Each tag must appear on a separate line, and includes special information such as author name, version number, parameters used by a method, and the value returned by a method. There are other tags, but these four are the ones most commonly used.

When a source code file containing documentation comments is processed by the `javadoc` utility, an HTML document is produced that formats the information in a standard way and which can be viewed in any web browser. If you've ever looked up any information about Java classes on Sun's website, you've probably recognized the standard format that is used. That's exactly the format that can be generated for your own classes.

Introduction to Programming Using Java

605.201

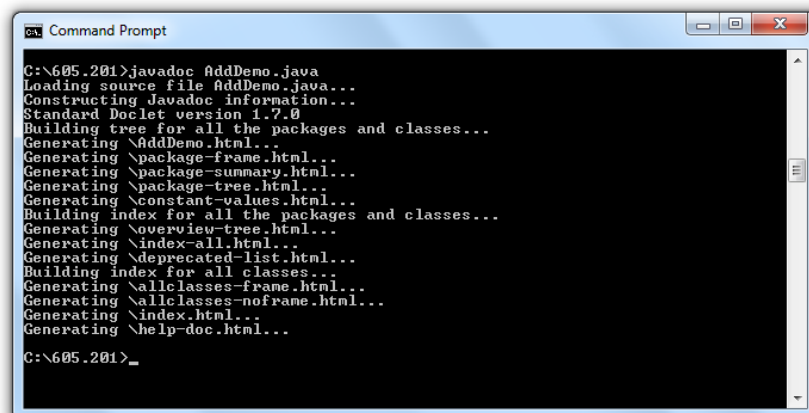
Creating Documentation With javadoc

Let's take an example to see how the `javadoc` utility works. Consider the following Java source code:

```
/**
 * This class computes and displays the sum of two integers
 * using a static method called sum().
 * @author Dr. Jerry Java
 * @version 1.0
 */
public class AddDemo
{
    public static void main( String [] args )
    {
        int firstValue = 10;
        int secondValue = 20;
        System.out.println( "The sum is " + sum( firstValue, secondValue ) );
    }

    /**
     * @param first  the first value
     * @param second the second value
     * @return the sum of first and second is returned
     */
    public static int sum( int first, int second )
    {
        return first + second;
    }
}
```

As you can see, the code contains a number of documentation comments, highlighted here in light blue. To generate the `javadoc` documentation, we simply run the `javadoc` utility that comes with the JDK. In the example below, I ran the `javadoc` utility from a command line window, giving it the name of my source code file.



```
ca Command Prompt
C:\605.201>javadoc AddDemo.java
Loading source file AddDemo.java...
Constructing Javadoc information...
Standard Doclet version 1.7.0
Building tree for all the packages and classes...
Generating \AddDemo.html...
Generating \package-frame.html...
Generating \package-summary.html...
Generating \package-tree.html...
Generating \constant-values.html...
Building index for all the packages and classes...
Generating \overview-tree.html...
Generating \index-all.html...
Generating \deprecated-list.html...
Building index for all classes...
Generating \allclasses-frame.html...
Generating \allclasses-noframe.html...
Generating \index.html...
Generating \help-doc.html...
C:\605.201>
```

Introduction to Programming Using Java

605.201

Creating Documentation With javadoc

As you can see from the above image, the utility generates quite a few files. Most of the files are HTML files, the most important one being `index.html`. When `index.html` is loaded into a web browser, the following page is displayed:

The screenshot shows a web browser displaying the javadoc-generated HTML page for the `AddDemo` class. The page has a navigation bar at the top with links: Package, Class (selected), Tree, Deprecated, Index, and Help. Below the navigation bar, there are links for Prev Class, Next Class, Frames, and No Frames. The main content area is titled "Class AddDemo" and shows the class hierarchy: `java.lang.Object` and `AddDemo`. The class description states: "public class AddDemo extends java.lang.Object. This class computes and displays the sum of two integers using a static method called sum()." Below the class description, there are three sections: "Constructor Summary", "Method Summary", and "Constructor Detail". The "Constructor Summary" section shows a table with one constructor: `AddDemo()`. The "Method Summary" section shows a table with two methods: `main(java.lang.String[] args)` and `sum(int first, int second)`. The "Constructor Detail" section shows the signature: `public AddDemo()`.

Package	Class	Tree	Deprecated	Index	Help
Prev Class	Next Class	Frames	No Frames	Summary: Nested Field Const Method	Detail: Field Const Method

Class AddDemo

java.lang.Object
AddDemo

public class AddDemo
extends java.lang.Object
This class computes and displays the sum of two integers using a static method called sum().

Constructor Summary

Constructors
Constructor and Description
AddDemo()

Method Summary

Methods	
Modifier and Type	Method and Description
static void	main(java.lang.String[] args)
static int	sum(int first, int second)

Methods inherited from class java.lang.Object

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

Constructor Detail

AddDemo
public AddDemo()

The above is just a partial page for my `AddDemo` class. It contains the description of the class I supplied in the source code documentation comments, and it lists the two methods associated with the class, namely `main()` and `sum()`. It also indicates that `AddDemo` is a subclass of `Object`, and lists the methods it inherits from the `Object` superclass.

If I scroll down further on the web page, it will list detailed information about each of the methods in `AddDemo` as illustrated below.

Introduction to Programming Using Java 605.201

Creating Documentation With javadoc

The screenshot displays the Javadoc output for a class named `AddDemo`. It is divided into two main sections: **Constructor Detail** and **Method Detail**.

Constructor Detail: Shows a single constructor `AddDemo` with the signature `public AddDemo()`.

Method Detail: Shows two static methods:

- main:** `public static void main(java.lang.String[] args)`
- sum:** `public static int sum(int first, int second)`. It includes parameter descriptions: `first` is "the first value" and `second` is "the second value". The **Returns:** section states "the sum of first and second is returned".

Using the navigation bar at the top of the web page, I can get additional views of the information associated with my class. For example, clicking on the Index link provides an alphabetical listing of the key class elements, as illustrated below.

The screenshot shows the **Index** page of the Javadoc documentation. The navigation bar at the top includes links for **Package**, **Class**, **Tree**, **Deprecated**, **Index** (which is highlighted), and **Help**. Below the navigation bar, there are links for **Prev**, **Next**, **Frames**, and **No Frames**.

The main content area is titled **A M S** and lists the following elements:

- A**
 - `AddDemo` - Class in <Unnamed>
This class computes and displays the sum of two integers using a static method called `sum()`.
 - `AddDemo()` - Constructor for class `AddDemo`
- M**
 - `main(String[])` - Static method in class `AddDemo`
- S**
 - `sum(int, int)` - Static method in class `AddDemo`

At the bottom of the list, the text **A M S** is repeated.

Clicking on the **Tree** link produces an inheritance tree view, which, in this simple class, is not all that interesting, but it's a nice feature to be able to view an inheritance structure in a graphical way.

The screenshot shows the **Tree** view of the Javadoc documentation. The navigation bar at the top includes links for **Package**, **Class**, **Tree** (which is highlighted), **Deprecated**, **Index**, and **Help**. Below the navigation bar, there are links for **Prev**, **Next**, **Frames**, and **No Frames**.

The main content area is titled **Hierarchy For All Packages** and shows the **Class Hierarchy** for the `AddDemo` class:

- `java.lang.Object`
 - `AddDemo`

Here's a link to more detailed information on using `javadoc`:

<http://www.oracle.com/technetwork/articles/java/index-137868.html>