Comments in Java

In a program, comments take part in making the program become more human readable by placing the detail of code involved and proper use of comments makes maintenance easier and finding bugs easily. Comments are ignored by the compiler while compiling a code.

In Java there are three types of comments:

1. Single – line comments.
2. Multi – line comments.
3. Documentation comments.

Single-line Comments

A beginner level programmer uses mostly single-line comments for describing the code functionality. Its the most easiest typed comments.
Syntax:

```
//Comments here( Text in this line only is considered as comment )
```

Example:

```
//Java program to show single line comments
class Scomment
{
    public static void main(String args[])
    {
        // Single line comment here
        System.out.println("Single line comment above");
    }
}
```

Run on IDE

Multi-line Comments

To describe a full method in a code or a complex snippet single line comments can be tedious to write, since we have to give '//' at every line. So to overcome this multi line comments can be used.
Syntax:

```
/*Comment starts
continues
continues
.
.
.
Commnent ends*/
```

Example:

```
//Java program to show multi line comments
class Scomment
{
    public static void main(String args[])
    {
        System.out.println("Multi line comments below");
        /*Comment line 1
          Comment line 2
          Comment line 3*/
    }
}
```

Run on IDE

We can also accomplish single line comments by using the above syntax as shown below:

```
/*Comment line 1*/
```

Documentation Comments

This type of comments are used generally when writing code for a project/software package, since it helps to generate a documentation page for reference, which can be used for getting information about methods present, its parameters, etc.
For example http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html is an auto generated documentation page which is generated by using documentation comments and a javadoc tool for processing the comments.

Syntax:

```
/**Comment start
*
*tags are used in order to specify a parameter
*or method or heading
*HTML tags can also be used
*such as <h1>
*
*comment ends*/
```

Available tags to use:

TAG	DESCRIPTION	SYNTAX

@author	Adds the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
{@docRoot}	Represents the relative path to the generated document's root directory from any generated page.	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecatedtext
@exception	Adds a **Throws** subheading to the generated documentation, with the classname and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the **nearest** inheritable class or implementable interface.	Inherits a comment from the immediate surperclass.
{@link}	Inserts an in-line link with the visible text label that points to the documentation for the specified package, class, or member name of a referenced class.	{@link package.class#member label}
{@linkplain}	Identical to {@link}, except the link's label is displayed in plain text than code font.	{@linkplain package.class#member label}
@param	Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section.	@param parameter-name description
@return	Adds a "Returns" section with the description text.	@return description
@see	Adds a "See Also" heading with a link or text entry that points to reference.	@see reference
@serial	Used in the doc comment for a default serializable field.	@serial field-description | include | exclude
@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods.	@serialData data-description
@serialField	Documents an ObjectStreamField component.	@serialField field-name field-type field-description
@since	Adds a "Since" heading with the specified since-text to the generated documentation.	@since release
@throws	The @throws and @exception tags are synonyms.	@throws class-name description

{@value}	When {@value} is used in the doc comment of a static field, it displays the value of that constant.	{@value package.class#field}
@version	Adds a “Version” subheading with the specified version-text to the generated docs when the -version option is used.	@version version-text

```
//Java program to illustrate frequently used
// Comment tags

/**
* <h1>Find average of three numbers!</h1>
* The FindAvg program implements an application that
* simply calculates average of three integers and Prints
* the output on the screen.
*
* @author  Pratik Agarwal
* @version 1.0
* @since   2017-02-18
*/
public class FindAvg
{
    /**
    * This method is used to find average of three integers.
    * @param numA This is the first parameter to findAvg method
    * @param numB  This is the second parameter to findAvg method
    * @param numC  This is the second parameter to findAvg method
    * @return int This returns average of numA, numB and numC.
    */
    public int findAvg(int numA, int numB, int numC)
    {
        return (numA + numB + numC)/3;
    }

    /**
    * This is the main method which makes use of findAvg method.
    * @param args Unused.
    * @return Nothing.
    */

    public static void main(String args[])
    {
        FindAvg obj = new FindAvg();
        int avg = obj.findAvg(10, 20, 30);

        System.out.println("Average of 10, 20 and 30 is :" + avg);
    }
}
```

Run on IDE

Output:

```
Average of 10, 20 and 30 is :20
```

For the above code documentation can be generated by using a tool ‘javadoc’ :
Javadoc can be used by running the following command in terminal.

```
javadoc FindAvg.java
```

This article is contributed by **Pratik Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to

contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Java School Programming java-basics

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Does Java support goto?
Interesting facts about null in Java
Encapsulation in Java
Type conversion in Java with Examples
Marker interface in Java
Java Math copySign() method with Examples
Java | Converting an Image into Grayscale using cvtColor()
Java | Removing whitespaces using Regex
Java System.nanoTime() vs System.currentTimeMillis
Java toDegrees() method with Example

(Login to Rate)

1.3 Average Difficulty : **1.3/5.0**
Based on **14** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org