



# **SNS COLLEGE OF ENGINEERING**

Kurumbapalayam (Po), Coimbatore – 641 107



## **An Autonomous Institution**

Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

### **23CSB101**

## **OBJECT ORIENTED PROGRAMMING**

**JavaDoc Comments, Constants, Identifiers Garbage  
Collection**

**By**

**M.Kanchana**

**Assistant Professor/CSE**



# JavaDoc Comments



Javadoc is a tool which comes with JDK and it is used for generating Java code documentation in HTML format from Java source code. Java documentation can be created as part of the source code.

## TYPES OF COMMENTS:

### *Class Comments*

The class comment must be placed *after* any import statements, directly before the class definition.

```
import java.io.*;
/** class comments should be written here */
Public class sample
{
....
}
```



# JavaDoc Comments



## *2.Method Comments*

The method comments must be placed immediately before the method that it describes.

Tag	Description	Syntax
<b>@param</b>	It describes the method parameter	<b>@param name description</b>
<b>@return</b>	This tag describes the return value from a method with the exception void methods and constructors.	<b>@return description</b>
<b>@throws</b>	This tag describes the method that throws an exception.	<b>@throws class description</b>



# JavaDoc Comments



## *3. Field Comments*

Field comments are used to document public fields—generally that means static constants.

```
/**  
 * Account number  
 */  
public static final int acc_no = 101;
```



# JavaDoc Comments



Tag	Meaning
@author	Identifies the author.
{@code}	Displays information as-is, without processing HTML styles, in code font.
@deprecated	Specifies that a program element is deprecated.
{@docRoot}	Specifies the path to the root directory of the current documentation.
@exception	Identifies an exception thrown by a method or constructor.
@hidden	Prevents an element from appearing in the documentation.
{@index}	Specifies a term for indexing.
{@inheritDoc}	Inherits a comment from the immediate superclass.
{@link}	Inserts an in-line link to another topic.
{@linkplain}	Inserts an in-line link to another topic, but the link is displayed in a plain-text font.
{@literal}	Displays information as-is, without processing HTML styles.
@param	Documents a parameter.



# JavaDoc Comments



Tag	Meaning
@provides	Documents a service provided by a module.
@return	Documents a method's return value.
@see	Specifies a link to another topic.
@serial	Documents a default serializable field.
@serialData	Documents the data written by the <code>writeObject( )</code> or <code>writeExternal( )</code> methods.
@serialField	Documents an <code>ObjectStreamField</code> component.
@since	States the release when a specific change was introduced.
{@summary}	Documents a summary of an item. (Added by JDK 10.)
@throws	Same as <b>@exception</b> .
@uses	Documents a service needed by a module.
{@value}	Displays the value of a constant, which must be a <b>static</b> field.
@version	Specifies the version of a program element.



# JavaDoc Comments



**javadoc -d docDirectory nameOfPackage**

for a single package. Or run

**javadoc -d docDirectory nameOfPackage1 nameOfPackage2...**

to document multiple packages.

If your files are in the default package, then instead run

**javadoc -d docDirectory \*.java**

If you omit the -d docDirectory option, then the HTML files are extracted to the current directory.



# JavaDoc Comments



```
/**
```

- \* This program performs the addition of two numbers.
- \* It demonstrates the use of Javadoc comments for documentation.
- \*
- \* @author ICSE
- \* @version 1.0
- \*/

```
public class Addition {
```

```
    /** Stores the first number for addition. */  
    private int num1;
```

```
    /** Stores the second number for addition. */  
    private int num2;
```





# JavaDoc Comments



```
/**
```

```
 * Constructor to initialize the numbers.
```

```
 *
```

```
 * @param num1 The first number
```

```
 * @param num2 The second number
```

```
 */
```

```
public Addition(int num1, int num2) {  
    this.num1 = num1;  
    this.num2 = num2;  
}
```

```
/**
```

```
 * Adds the two numbers and returns the sum.
```

```
 *
```

```
 * @return The sum of num1 and num2
```

```
 */
```

```
public int addNumbers() {  
    return num1 + num2;  
}
```



# JavaDoc Comments



```
/**
```

```
 * Main method to test the addition operation.
```

```
 *
```

```
 * @param args Command-line arguments (not used)
```

```
 */
```

```
public static void main(String[] args) {
```

```
    Addition addition = new Addition(10, 5); // Creating an object with  
numbers 10 and 5
```

```
    int result = addition.addNumbers(); // Performing addition
```

```
    // Displaying the result
```

```
    System.out.println("The sum is: " + result);
```

```
 }
```

```
}
```



# JavaDoc Comments





# JavaDoc Comments



```
D:\JAVA Programs>javadoc -d docs Addition.java
```

```
Loading source file Addition.java...
```

```
Constructing Javadoc information...
```

```
Standard Doclet version 1.8.0_201
```

```
Building tree for all the packages and classes...
```

```
Generating docs\Addition.html...
```

```
Generating docs\package-frame.html...
```

```
Generating docs\package-summary.html...
```

```
Generating docs\package-tree.html...
```

```
Generating docs\constant-values.html...
```

```
Building index for all the packages and classes...
```

```
Generating docs\overview-tree.html...
```

```
Generating docs\index-all.html...
```

```
Generating docs\deprecated-list.html...
```

```
Building index for all classes...
```

```
Generating docs\allclasses-frame.html...
```

```
Generating docs\allclasses-noframe.html...
```

```
Generating docs\index.html...
```

```
Generating docs\help-doc.html...
```



# JavaDoc Comments



The screenshot shows a Windows File Explorer window titled "JAVA Programs" located on "New Volume (D:)". The ribbon includes "File", "Home", "Share", and "View" tabs. The ribbon buttons are organized into groups: Clipboard (Pin to Quick access, Copy, Paste, Copy path, Paste shortcut), Organize (Move to, Copy to, Delete, Rename), New (New folder, New item, Easy access), Open (Properties, Open, History), and Select (Select all, Select none, Invert selection). The address bar shows the path: "This PC > New Volume (D:) > JAVA Programs". The main pane displays a list of 25 items:

Name	Date modified	Type	Size
docs	04-03-2025 12:29	File folder	
mypackage1	27-02-2025 12:21	File folder	
mypackage2	27-02-2025 12:21	File folder	
Addition.class	04-03-2025 12:29	CLASS File	1 KB
Addition.java	04-03-2025 12:29	JAVA File	2 KB
Animal.class	15-02-2025 12:09	CLASS File	1 KB
Arithmetic.class	20-02-2025 11:51	CLASS File	2 KB
Arithmetic.java	20-02-2025 12:28	JAVA File	2 KB
Arithmetic.txt	20-02-2025 10:12	Text Document	1 KB
Box.class	26-02-2025 09:59	CLASS File	1 KB
BoxDemo.class	26-02-2025 09:59	CLASS File	1 KB
BoxDemo.java	26-02-2025 09:55	JAVA File	1 KB
Calculator.class	04-03-2025 12:01	CLASS File	2 KB
Calculator.java	04-03-2025 12:01	JAVA File	3 KB
Cat.class	15-02-2025 12:09	CLASS File	1 KB
Counter.class	17-02-2025 13:21	CLASS File	1 KB
Counter.java	17-02-2025 13:21	JAVA File	1 KB
Dog.class	15-02-2025 12:09	CLASS File	1 KB
Dog.java	14-02-2025 12:05	JAVA File	1 KB
Example.class	26-02-2025 10:03	CLASS File	1 KB
first.class	12-02-2025 10:46	CLASS File	1 KB
Main.class	04-03-2025 11:53	CLASS File	1 KB
Main.java	04-03-2025 11:56	JAVA File	3 KB
NC.java	26-02-2025 09:59	JAVA File	1 KB

The taskbar at the bottom shows the Start button, a search bar with the text "Type here to search", and several application icons. The system tray on the right shows the date and time as "13:08 04-03-2025".



# JavaDoc Comments



The screenshot shows a Windows File Explorer window titled 'docs' located at 'This PC > New Volume (D:) > JAVA Programs > docs'. The window displays a list of 16 files and folders. The files are primarily HTML documents generated by Microsoft Edge, with sizes ranging from 1 KB to 14 KB. There is also a JavaScript file and a Cascading Style Sheet file.

Name	Date modified	Type	Size
Quick access			
OneDrive			
This PC			
3D Objects			
Desktop			
Documents			
Downloads			
Music			
Pictures			
Videos			
Local Disk (C:)			
New Volume (D:)			
Network			
Addition.html	04-03-2025 12:29	Microsoft Edge H...	10 KB
allclasses-frame.html	04-03-2025 12:29	Microsoft Edge H...	1 KB
allclasses-noframe.html	04-03-2025 12:29	Microsoft Edge H...	1 KB
Calculator.html	04-03-2025 12:02	Microsoft Edge H...	13 KB
constant-values.html	04-03-2025 12:29	Microsoft Edge H...	4 KB
deprecated-list.html	04-03-2025 12:29	Microsoft Edge H...	4 KB
help-doc.html	04-03-2025 12:29	Microsoft Edge H...	8 KB
index.html	04-03-2025 12:29	Microsoft Edge H...	3 KB
index-all.html	04-03-2025 12:29	Microsoft Edge H...	5 KB
overview-tree.html	04-03-2025 12:29	Microsoft Edge H...	4 KB
package-frame.html	04-03-2025 12:29	Microsoft Edge H...	1 KB
package-list	04-03-2025 12:29	File	1 KB
package-summary.html	04-03-2025 12:29	Microsoft Edge H...	4 KB
package-tree.html	04-03-2025 12:29	Microsoft Edge H...	4 KB
script.js	04-03-2025 12:29	JavaScript File	1 KB
stylesheet.css	04-03-2025 12:02	Cascading Style S...	14 KB



# JavaDoc Comments



PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

## Class Calculator

java.lang.Object  
Calculator

---

public class Calculator  
extends java.lang.Object

This is a simple Calculator program that performs basic arithmetic operations. It includes methods for addition, subtraction, multiplication, and division.

### Constructor Summary

Constructors
Constructor and Description
Calculator()

### Method Summary

All Methods	Static Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description		



# Java Comments



- Java comments are either explanations of the source code or descriptions of classes, interfaces, methods, and fields.
- Comments in Java do not show up in the executable program.

**Line comment:** When you want to make a one line comment type `"/"` and follow the two forward slashes with your comment.

**Syntax:** `// text`

## **Block Comment:**

To start a block comment type `"/*`". Everything between the forward slash and asterisk, even if it's on a different line, will be treated as comment until the characters `*/` end the comment.

**Syntax:** `/* text */`





# JAVA - CONSTANTS



- A constant is an identifier written in uppercase (convention and not a rule) that prevents
- its contents from being modified by the program during the execution.
- If an attempt is made to change the value, the compiler will give an error message.
- In Java, the keyword **final** is used to declare constants.
- The value of a final variable cannot change after it has been initialized.

**final datatype variablename=value;**

**final float PI=3.14f;**



# JAVA - IDENTIFIERS



- Identifiers are names given to the variables, classes, methods, objects, labels, package and interface in our program.
- The name we are giving must be meaningful and it may have random length.

The following rule must be followed while giving a name:

1. The first character must not begin with a number.
2. The identifier is formed with alphabets, number, dollar sign (\$) and underscore (\_).
3. It should not be a reserved word.
4. Space is not allowed in between the identifier name.



## JAVA – RESERVED WORDS (KEYWORDS)



<b>abstract</b>	<b>assert</b>	<b>boolean</b>	<b>break</b>	<b>byte</b>	<b>case</b>
<b>catch</b>	<b>char</b>	<b>class</b>	const*	continue	default
<b>double</b>	do	else	enum	extends	false
<b>final</b>	<b>finally</b>	<b>float</b>	<b>for</b>	goto*	<b>if</b>
<b>implements</b>	<b>import</b>	instanceof	<b>int</b>	<b>interface</b>	long
native	<b>new</b>	null	<b>package</b>	private	protected
<b>public</b>	return	short	<b>static</b>	strictfp	<b>super</b>
switch	synchronized	<b>this</b>	<b>throw</b>	<b>throws</b>	transient
true	<b>try</b>	<b>void</b>	volatile	while	



# GARBAGE\_COLLECTION



- Since objects are dynamically allocated by using the **new** operator, you might be wondering how such objects are destroyed and their memory released for later reallocation.
- Garbage collection (GC) in Java is a process that automatically removes unused objects from memory, helping to manage memory efficiently.
- The JVM's Garbage Collector (GC) looks for objects that no longer have references.
- Once an object isn't reachable, it is marked for deletion.
- The GC reclaims memory, preventing memory leaks.



# GARBAGE\_COLLECTION



The **finalize()** method allows an object to clean up resources before it is destroyed.(Think of it like packing up your things before leaving a house.)

```
class Example {  
    // Constructor to create an object  
    Example() {  
        System.out.println("Object Created!");  
    }  
  
    // finalize() method  
    @Override  
    protected void finalize() {  
        System.out.println("Object is being  
deleted...");  
    }  
}
```

```
public static void main(String[] args) {  
    Example obj = new Example(); // Creating an object  
  
    obj = null; // Making the object eligible for garbage  
collection  
  
    System.gc(); // Requesting garbage collection  
  
    System.out.println("End of program.");  
    }  
}
```



# GARBAGE\_COLLECTION



Output:

Object Created!

End of program.

Object is being deleted...



# THANK YOU