JavaDoc

JavaDoc tool is a **document generator tool** in Java programming language for generating standard documentation in HTML format. It generates API documentation. It parses the declarations ad documentation in a set of source file describing classes, methods, constructors, and fields.

The JavaDoc comments is different from the normal comments

```
// Single-Line Comment

/*
* Multiple-Line comment
*/

/**
* JavaDoc comment
*/
```

JavaDoc Tags

Tag	Parameter	Description
@author	author_name	Describes an author
@param	description	provide information about method parameter or the input it takes
@see	reference	generate a link to other element of the document
@version	version-name	provide version of the class, interface or enum.
@return	description	provide the return value

```
package doc
import java.util.Scanner;

/**
*
* @author Debasihs
*/
public class Add{
  /**
* This is a program for adding two numbers in java.
  * @param args
*/
  public static void main(String[] args)
  {
    /**
```

```
     * This is the main method
     * which is very important for
     * execution for a java program.
     */

     int x, y;
     Scanner sc = new Scanner(System.in);
     /**
     * Declared two variables x and y.
     * And taking input from the user
     * by using Scanner class.
     *
     */

     x = sc.nextInt();
     y = sc.nextInt();
     /**
     * Storing the result in variable sum
     * which is of the integer type.
     */
     int sum = x + y;

     /**
     * Using standard output stream
     * for giving the output.
     * @return null
     */
     System.out.println("Sum is: " + sum);
   }
 }
```

Tags	Syntax with parameters	Description
@author	@author <name-text>	It is used to show the name of the author of the code.
@code	@code <text>	It is used to display the text in code font without interpreting it.
@docRoot	@docRoot	It shows the path to the root directory of the generated document from any generated page.
@deprecated	@deprecated <deprecatedtext>	Adds a comment to describe if an API should no longer be used.
@version	@version <version-text>	It adds a “Version” subheading to show the version of the code.
@param	@param <parameter-name description>	It provides information about the parameters passed in a method.

Tags	Syntax with parameters	Description
@return	@return <description>	Adds a “Returns” section that is used to describe the return value for a method.
@see	@see <reference>	Adds a “See Also” heading with a link or text to add a reference to another part of the code.
@since	@since <release>	It adds a “Since” heading with the year of release of the code.
@exception	@exception <class-name description	It is used to specify the type of exception that occurred in the code if the “throw” keyword is used in any method. The @throws tag also does the same thing.

```
import java.io.*;

/**
* <h1>Find the Product of two numbers</h1>
* This program multiplies two integer values and outputs their product.
* <p>
*
* @author Debasish Sahoo
* @version 1.0
 * @since 2023-06-16
 */
 public class calcProduct {
   /**
   * This method is used to find the product of two integers.
    This is a simple class method,
   * demonstrating the use of some Java doc Tags.
   * @param number1 This is the first parameter to the calcProduct method.
   * @param number2 This is the second parameter to the calcProduct method.
   * @return int It will return the product of both parameters in integer.
   */
   public int calcProduct (int number1, int number2) {
    return number1 * number2;
   }

   /**
   * This is the main method which makes use of the calcProduct method.
   * @param args Unused.
   * @return Nothing.
   * @throws IOException on input error.
   * @see IOException
   */

 public static void main(String args[]) throws IOException {
    calcProduct obj1 = new calcProduct();
    int product = obj1. calcProduct(7, 3);

    System.out.println("Product of 7 and 3 :" + product);
   }
 }
```