

# Java Documentation Comments

The javadoc Tags  
javadoc Tool



# Java comments

---

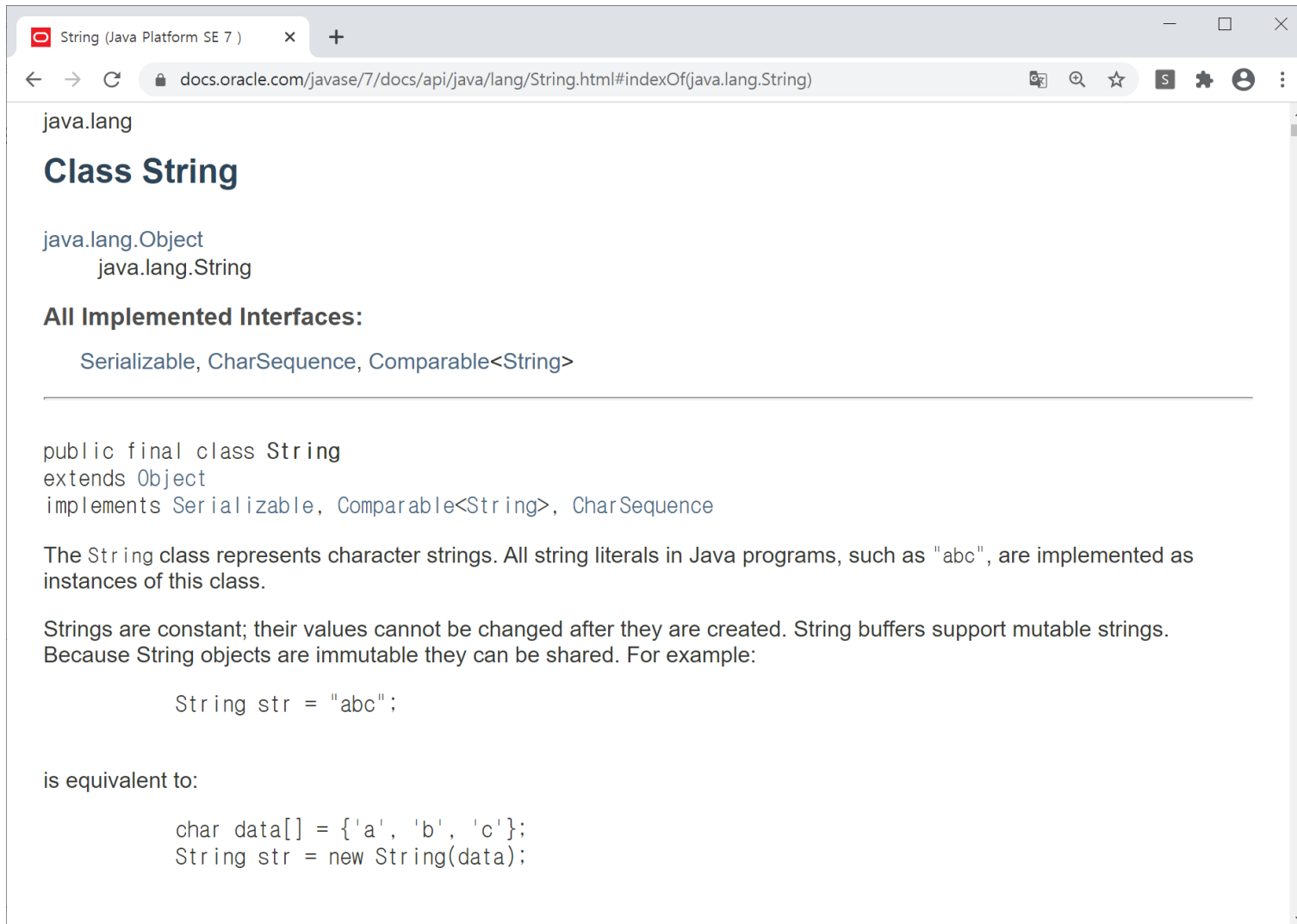
## ❖ Java Comments

- `/* comments */`
- `// line comment`
- `/** Java documentation comment */`

## ❖ Java Documentation Comment

- allow you to embed information about your program into the program itself.
- Documentation comments make it convenient to document your programs.
- You can then use the javadoc utility program to extract the information and put it into an HTML file.

# Java API Documentation



The screenshot shows a web browser window displaying the Java API documentation for the `String` class. The browser's address bar shows the URL `docs.oracle.com/javase/7/docs/api/java/lang/String.html#indexOf(java.lang.String)`. The page content includes the package `java.lang`, the class name `Class String`, and its inheritance hierarchy: `java.lang.Object` and `java.lang.String`. It lists the implemented interfaces: `Serializable`, `CharSequence`, and `Comparable<String>`. The source code for the `String` class is shown, indicating it is a public final class that extends `Object` and implements `Serializable`, `Comparable<String>`, and `CharSequence`. A descriptive paragraph states that the `String` class represents character strings and that all string literals in Java programs are implemented as instances of this class. Another paragraph explains that strings are constant and immutable, and that `String` buffers support mutable strings. An example code snippet shows `String str = "abc";`, followed by a statement that this is equivalent to a more verbose construction using a character array.

String (Java Platform SE 7 )

docs.oracle.com/javase/7/docs/api/java/lang/String.html#indexOf(java.lang.String)

java.lang

## Class String

java.lang.Object  
java.lang.String

**All Implemented Interfaces:**

Serializable, CharSequence, Comparable<String>

---

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because `String` objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

```
package javadoc;  
import java.io.*;
```

```
/**  
 * This class demonstrates documentation comments.  
 * @author Ayan Amhed  
 * @version 1.2  
 */
```

```
public class SquareNum {  
    /**  
     * This method returns the square of num.  
     * This is a multiline description. You can use  
     * as many lines as you like.  
     * @param num The value to be squared.  
     * @return num squared.  
     */  
    public double square(double num) {  
        return num * num;  
    }  
}
```



```

/**
 * This method inputs a number from the user.
 * @return The value input as a double.
 * @exception IOException On input error.
 * @see IOException
 */
public double getNumber() throws IOException {
    InputStreamReader isr = new InputStreamReader(System.in);
    BufferedReader inData = new BufferedReader(isr);
    String str;
    str = inData.readLine();
    return (new Double(str)).doubleValue();
}

/**
 * This method demonstrates square().
 * @param args Unused.
 * @return Nothing.
 * @exception IOException On input error.
 * @see IOException
 */
public static void main(String args[]) throws IOException
{
    SquareNum ob = new SquareNum();
    double val;
    System.out.println("Enter value to be squared: ");
    val = ob.getNumber();
    val = ob.square(val);
    System.out.println("Squared value is " + val);
}
}

```



# How to use javadoc tool

---

❖ % javadoc SquareNum.java

# The Generated Documentation

## Class SquareNum

java.lang.Object  
javadoc.SquareNum

```
public class SquareNum
extends java.lang.Object
```

This class demonstrates documentation comments.

### Version:

1.2

### Author:

Ayan Amhed

## Constructor Summary

### Constructors

#### Constructor and Description

SquareNum()

## Method Summary

### All Methods

### Static Methods

### Instance Methods

### Concrete Methods

#### Modifier and Type

#### Method and Description

double

getNumber()

This method inputs a number from the user.

static void

main(java.lang.String[] args)

This method demonstrates square().

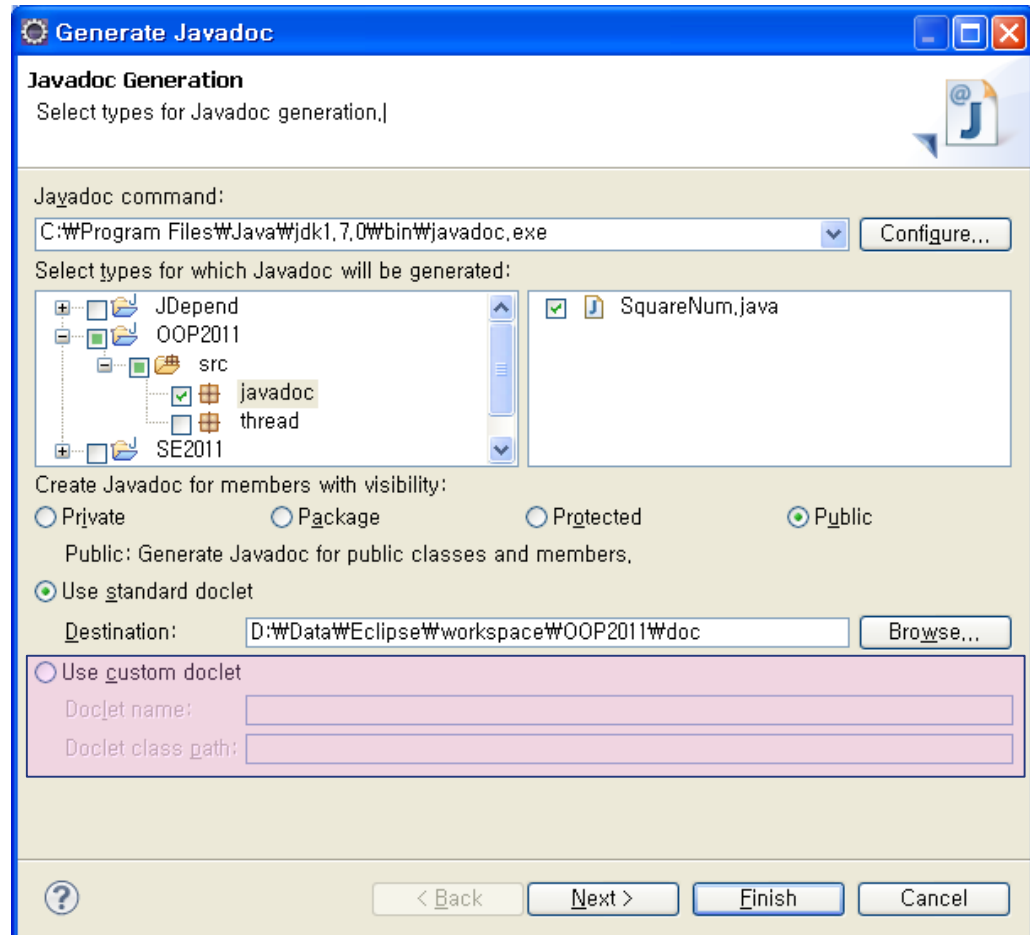
double

square(double num)

This method returns the square of num.

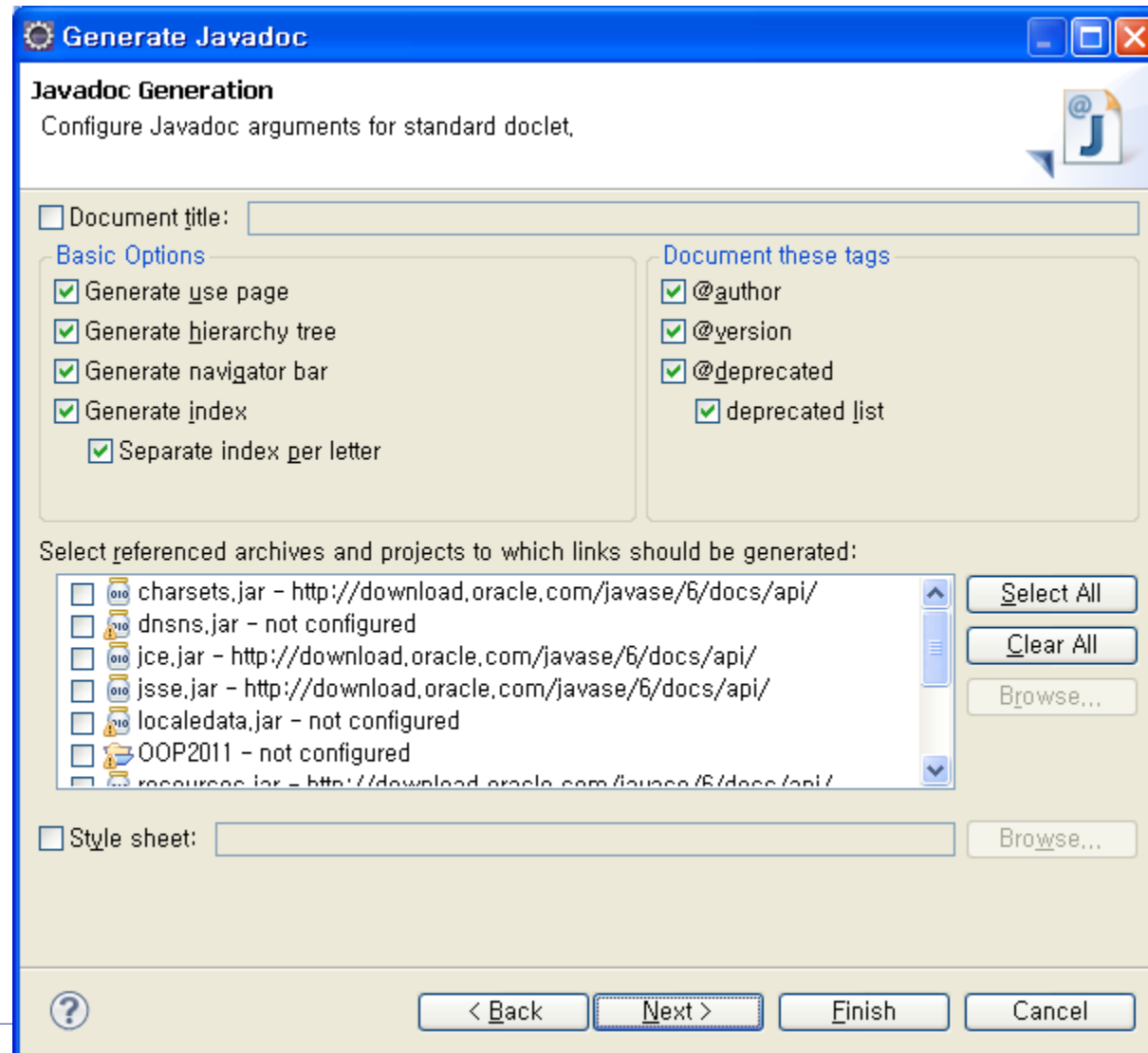
# Invoking javadoc in Eclipse

## ❖ Project – Generate Javadoc...





# Invoking javadoc in Eclipse



# JavaDoc Jags: Summary

Tag & Parameter	Usage	Applies to
<b>@author</b> <i>name</i>	Describes an author.	Class, Interface, Enum
<b>@version</b> <i>version</i>	Provides software version entry. Max one per Class or Interface.	Class, Interface, Enum
<b>@since</b> <i>since-text</i>	Describes when this functionality has first existed.	Class, Interface, Enum, Field, Method
<b>@see</b> <i>reference</i>	Provides a link to other element of documentation.	Class, Interface, Enum, Field, Method
<b>@param</b> <i>name description</i>	Describes a method parameter.	Method
<b>@return</b> <i>description</i>	Describes the return value.	Method
<b>@exception</b> <i>classname description</i> <b>@throws</b> <i>classname description</i>	Describes an exception that may be thrown from this method.	Method

---

# Doclet

# Doclet

---

- ❖ Doclets are programs written in the Java™ programming language that use the doclet API to specify the content and format of the output of the Javadoc tool.
- ❖ By default, the Javadoc tool uses the "standard" doclet to generate API documentation in HTML form.
- ❖ However, you can supply your own doclets to customize the output of Javadoc as you like

# Steps to Create and Use Your Own Doclet

---

- ❖ 1) Write your own doclet in Java
  - Your program should import `com.sun.javadoc.*` in order to use the doclet API.
  - The entry point of your program is a class with a public static boolean `start` method that takes a `RootDoc` as a parameter.
- ❖ 2) Compile your doclet.
  - You can use the compiler in the Java 2 SDK, `javac`.
- ❖ 3) Run the javadoc tool
  - using the `-doclet startingclass` option to produce the output specified by your doclet

# A Simple Example Doclet

---

```
import com.sun.javadoc.*;

public class ListClass {
    public static boolean start(RootDoc root) {
        ClassDoc[] classes = root.classes();
        for (int i = 0; i < classes.length; ++i) {
            System.out.println(classes[i]);
        }
        return true;
    }
}
```

❖ 2) Compile your doclet.

❖ C:\W ... \WOOP\src>**javac -classpath "C:\WProgram Files\Java\jdk1.8.0\_121\lib\tools.jar"** ListClass.java

# A Simple Example Doclet

---

## ❖ 3) Run the javadoc tool

```
C:\W ... \WOOP\src>javadoc -doclet ListClass -docletpath . javadoc\SquareNum.java
```

```
Loading source file javadoc\SquareNum.java...
```

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

```
javadoc.SquareNum
```

```
C:\W ... \WOOP\src>
```

# References

---

## ❖ **Java Documentation Comments** from Tutorials Point

- [http://www.tutorialspoint.com/java/java\\_documentation.htm](http://www.tutorialspoint.com/java/java_documentation.htm)

## ❖ **How to Write Doc Comments for the Javadoc Tool**

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

## ❖ **Javadoc Tool**

- <http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html>

## ❖ **Doclet Overview**

- <https://docs.oracle.com/javase/7/docs/technotes/guides/javadoc/doclet/overview.html>