OpenAI

Java documentation comments

Java documentation comments detail tutorial

Sure! Let’s go step-by-step through **Java documentation comments**, often called **Javadoc comments**, including **syntax, usage, tags, and examples**.

🧠 What Are Documentation Comments?

In Java, **documentation comments** are **multi-line comments** that start with `/**` and end with `*/`. They are used to **generate API documentation** automatically using the **Javadoc tool**.

Example:

```java
/**
 * This class represents a simple calculator that can add and subtract numbers.
 */
public class Calculator {
    // ...
}
```

🧩 Syntax

```java
/**
 * Description of the class, method, or field.
 *
 * @tag Description of the tag (optional, depends on context)
 */
```

- Starts with `/**`
- Each line often begins with `*` for readability (but not required)
- Ends with `*/`

💬 Where Can You Use Javadoc Comments?

Element	Description
Class / Interface / Enum	Describe purpose and usage
Method / Constructor	Describe behavior, parameters, return values, exceptions
Field	Describe what the variable represents
Package	Describe overall contents and purpose

🧱 Common Javadoc Tags

Here's a table of the **most commonly used tags**:

Tag	Meaning	Example
`@author`	Author of the class	`@author John`
`@version`	Version number	`@version 1.0`
`@since`	When it was introduced	`@since 1.8`
`@param`	Describe a parameter	`@param x The first number`
`@return`	Describe return value	`@return The sum of x and y`
`@throws` or `@exception`	Exception thrown	`@throws IllegalArgumentException if x < 0`
`@see`	References another class or method	`@see Math#pow(double, double)`
`@deprecated`	Marks element as obsolete	`@deprecated Use addNumbers() instead.`
`@serial`	For serializable fields	`@serial include`
`@link`	Hyperlink within docs	`{@link String#length()}`
`@code`	Inline code snippet	`{@code int x = 0;}`

🧰 Example: Full Javadoc in Action

```java
/**
 * The Calculator class provides basic arithmetic operations.
 *
 * <p>This class can perform addition and subtraction of integers.</p>
 *
 * @author Jane
 * @version 1.0
 * @since 2025-11
 */
public class Calculator {

    /**
     * Adds two integers.
     *
     * @param a the first number
     * @param b the second number
     * @return the sum of a and b
     */
    public int add(int a, int b) {
        return a + b;
    }

    /**
     * Subtracts one integer from another.
     *
     * @param a the number to subtract from
     * @param b the number to subtract
     * @return the result of a - b
     */
    public int subtract(int a, int b) {
        return a - b;
    }

    /**
```

```
     * Divides two integers.
     *
     * @param a numerator
     * @param b denominator
     * @return result of a / b
     * @throws ArithmeticException if b is zero
     */
    public double divide(int a, int b) throws ArithmeticException {
        if (b == 0)
            throw new ArithmeticException("Cannot divide by zero");
        return (double) a / b;
    }
}
```

📚 Generating HTML Documentation

Once you've written Javadoc comments, you can generate documentation using the **`javadoc` command**.

Command:

```bash
javadoc -d docs Calculator.java
```

- **`-d docs`** → Output directory for documentation (creates folder **`docs`**)
- **`Calculator.java`** → File to generate docs for

Open **`docs/index.html`** in a browser to view the generated documentation.

🧠 Best Practices

✅ Use **clear, concise** language
✅ Start descriptions with a **verb** (e.g., "Returns", "Calculates", "Initializes")
✅ Document **parameters and return values** for every public method
✅ Use **`{@code ...}`** for inline code
✅ Keep tags **consistent** across files
✅ Use **HTML tags** like **`<p>`**, **`<br>`**, **`<pre>`** for formatting

---

✨ Quick Example Summary

Purpose	Example
Class doc	`/** Class for performing math operations. */`
Method doc	`/** Adds two numbers. @param a first, @param b second */`
Inline link	`{@link Math#sqrt(double)}`
Inline code	`{@code int result = add(2, 3);}`

Would you like me to show you a **sample generated Javadoc HTML page** (the kind the **`javadoc`** command produces)? I can describe or recreate its structure visually for you.