Documentation Tool: Javadoc

Introduction

Javadoc is a tool provided by Oracle that generates HTML documentation from Java source code. It is widely used for creating API documentation directly from comments in Java code. Javadoc processes special comments (`/** ... */`) to automatically generate clean, readable documentation for classes, methods, and fields. It's particularly popular in Java-based projects because of its seamless integration with Java development workflows.

Javadoc Installation Guide:

Javadoc comes pre-installed with the Java Development Kit (JDK). If you already have the JDK installed, you can use Javadoc immediately. Here's a step-by-step guide to check if Javadoc is installed and how to install the JDK if needed:

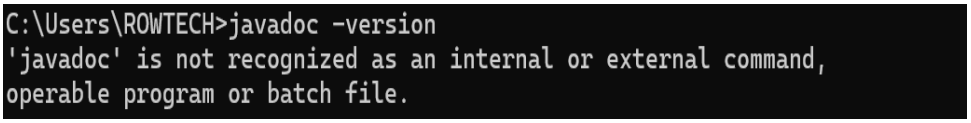
Steps to Install Javadoc (via JDK installation):

1. Check if JDK is installed:

- Open a command prompt (Windows) or terminal (Mac/Linux).
- Run the following command:

```
java -version
```

- If Javadoc is installed, you'll see the version number. If not, follow the steps below to install the JDK.



2. Install the JDK:

- Visit the Oracle JDK download page or OpenJDK page to download the latest version of the JDK.
- Download the JDK appropriate for your operating system (Windows, macOS, or Linux).
- Install the JDK by following the prompts in the installer.

3. Set up JDK on your system:

Windows:

- After installation, open the Environment Variables settings and add a new entry for `JAVA_HOME` pointing to the JDK installation directory (e.g., `C:\Program Files\Java\jdk-<version>`).
- Edit the Path variable and add `%JAVA_HOME%\bin` to it.

macOS/Linux:

- Open a terminal and add the following to your `.bashrc`, `.bash_profile`, or `.zshrc` file:

```
export
↪  JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-<version>.jdk/Contents/Home
export PATH=$JAVA_HOME/bin:$PATH
```

- Replace `<version>` with the version of your installed JDK.
- Reload the terminal configuration:

```
source ~/.bashrc  # or ~/.bash_profile
```

4. Verify Javadoc installation:

- Open a new command prompt/terminal and run:

```
java -version
```

- If the installation is successful, you should see the version number of Javadoc.

```
C:\Users\ROWTECH>java -version
java version "23" 2024-09-17
Java(TM) SE Runtime Environment (build 23+37-2369)
Java HotSpot(TM) 64-Bit Server VM (build 23+37-2369, mixed mode, sharing)

C:\Users\ROWTECH>javadoc --version
javadoc 23
```

Once the JDK (and Javadoc) is installed, you can use the **javadoc** command to generate documentation for your Java code.

Javadoc Documentation:

How to Write Doc Comments for the Javadoc Tool click here.

Documentation Approach Using -Javadoc

1 Setting Up a Simple Java Program

For this, we will use the *National Disaster Alert System* as our example. This system sends alerts to the public and logs a history of these alerts. The code includes well-documented methods and classes, which we will generate documentation for.

1.1 Code Example: NationalDisaster.java

```java
  /**
 * National Disaster Alert System
 *
 * This system is designed to send disaster alerts to citizens
 * and keep a log of the alert history.
 *
 * @version 1.0
 */
public class NationalDisaster {

    private String alertMessage;
    private String alertType;

    /**
     * Constructor to initialize the alert with a type and message.
     *
     * @param alertType The type of the disaster (e.g., earthquake, flood, cyclone)
     * @param alertMessage The message to be broadcasted to the public
     */
```

```
    public NationalDisaster(String alertType, String alertMessage) {
        this.alertType = alertType;
        this.alertMessage = alertMessage;
    }

    /**
     * Sends the disaster alert to the public.
     *
     * @return A confirmation message of the alert being sent.
     */
    public String sendAlert() {
        // Simulate sending alert
        System.out.println("Sending alert: " + alertType);
        System.out.println("Message: " + alertMessage);
        return "Alert sent successfully!";
    }

    /**
     * Logs the disaster alert history.
     *
     * @param alertType The type of the disaster that occurred
     * @param alertMessage The message that was sent out
     */
    public void logAlertHistory(String alertType, String alertMessage) {
        // Simulate logging the alert history
        System.out.println("Logging Alert: [Type: " + alertType + ", Message: " +
        ↪  alertMessage + "]");
    }

    /**
     * Main method to demonstrate sending and logging alerts.
     *
     * @param args Command-line arguments
     */
    public static void main(String[] args) {
        NationalDisaster alertSystem = new NationalDisaster("Earthquake", "Evacuate
        ↪  immediately!");
        alertSystem.sendAlert();
        alertSystem.logAlertHistory("Earthquake", "Evacuate immediately!");
    }
}
```

2 Adding javadoc Comments

Javadoc uses doc comments, which are multi-line comments that start with /** and end with */. These comments are placed above classes, methods, and fields to describe their purpose and behavior.

Here are the key tags we used:

- `@param`: Describes method parameters.
- `@return`: Describes what the method returns.

- `@version`: Provides version information about the class.

3 Generating Documentation with javadoc

After writing the code and embedding the javadoc comments, we can generate the documentation using the javadoc tool.

3.1 Steps to Generate the Documentation

1. **Navigate to the Directory:** Open a command prompt and navigate to the directory where `NationalDisaster.java` is located.

```
cd C:\Users\ROWTECH\IdeaProjects\training_modules\src
```

2. **Run the javadoc Command:** Run the following command to generate documentation in a folder named `doc`:

```
javadoc -d doc NationalDisaster.java
```

   The `-d` flag specifies the directory where the HTML files will be generated.

4 Viewing the Generated Documentation

Once the javadoc command has successfully generated the documentation, follow these steps to view it:

- **Open the `doc` Folder:** Navigate to the `doc` folder that was created during the javadoc process. This folder contains several HTML files, including `index.html`.
- **Open the Documentation:** Double-click `index.html` to open it in your default web browser. This will display the main page of the documentation.
- **Explore the Documentation:** The `index.html` page will provide links to the documentation for the class (`NationalDisaster`), including its fields, methods, constructors, and more.

5 Best Practices for Writing Documentation

Here are some best practices to keep in mind when writing documentation for your Java code:

- **Use Clear and Concise Language:** Ensure that your comments are easy to understand and describe the functionality without ambiguity.
- **Document All Public Classes and Methods:** Always include javadoc comments for public APIs that will be used by others. Document all parameters, return types, and exceptions.
- **Use Tags Appropriately:** Take advantage of common javadoc tags such as:
  - `@param`: For method parameters.
  - `@return`: For describing return values.
  - `@throws`: To describe exceptions that may be thrown by methods.
  - `@see`: To refer to related methods or classes.
- **Update Documentation as Code Changes:** Keep the documentation up-to-date with your codebase to avoid confusion. Always review and edit documentation whenever methods or classes change.

6 Further Reading and Resources

- Official javadoc Documentation: Java Documentation
- Best Practices for Documentation: Consider exploring guides and tutorials on writing technical documentation to improve the quality of your comments.

By following this process, you'll be able to create detailed and professional documentation for your Java projects, ensuring your code is understandable and maintainable.

Why Javadoc?

- **Standard for Java:** Native tool for Java developers.
- **Automatic Documentation:** Generates documentation directly from source code.
- **Extensibility:** Supports custom doclets and taglets for extended functionality.
- **Integration:** Easily integrates with build tools like Maven and Gradle.
- **API Reference Documentation:** Great for creating detailed API documentation.

Why not Javadoc?

- **Limited to Java:** Javadoc works only with Java code.
- **Learning Curve:** Understanding the special comment syntax can take time for beginners.
- **Output Customization:** While functional, customizing the output format (beyond HTML) can require additional tools or knowledge.

Despite some limitations, Javadoc remains the most widely used documentation tool in Java development.