The Javadoc tool

By generating Javadoc documentation for your code, you can produce an HTML file that contains the class documentation and other information about your code. This can be useful for sharing your code with others or for creating reference documentation for a project.

See the example of the documentation of the `java.utils.LinkedList` from Java:

https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/util/LinkedList.html

Generating Java documentation

The command to generate Java documentation using Javadoc from the command line is:

```
javadoc -d <output_directory> <source_files>
```

Here's what each part of the command means:

- `javadoc`: This is the command to run the Javadoc tool.
- `-d`: Specifies the output directory where the generated documentation will be saved.
- `<output_directory>`: Replace this with the path to the output directory where you want to save the generated documentation.
- `<source_files>`: Replace this with the path to the directory or file containing the Java source code you want to generate documentation for. You can also specify multiple source files or directories separated by spaces.

For example, let's say you have a Java application with source code located in a package/directory called `lab3`, and you want to generate documentation in a directory called `docs`. You can run the following command:

```
javadoc -d docs lab3
```

This will generate Java documentation for all the source code in the `lab3` directory and save it in the `docs` directory.

Javadoc comments

By including Javadoc comments in your code, you can provide clear and comprehensive documentation that will be automatically generated by the Javadoc tool. There are special

comments that should be included in your Java code to provide documentation that will be extracted by the Javadoc tool and included in the generated documentation. These comments are called Javadoc comments, and they have a specific format.

A **Javadoc comment for a method** would look like this:

```
/**
 * This method returns the sum of two integers.
 *
 * @param a the first integer
 * @param b the second integer
 * @return the sum of a and b
 */
public int sum(int a, int b) {
    return a + b;
}
```

In this example, the Javadoc comment is enclosed in a multi-line comment block that starts with `/**` and ends with `*/`. The comment includes a description of the method, followed by special tags that provide additional information:

- `@param`: This tag is used to describe each parameter of the method. The tag should be followed by the name of the parameter and a description of its purpose.
- `@return`: This tag is used to describe the return value of the method. The tag should be followed by a description of what the method returns.

Other common Javadoc tags include:

- `@throws`: This tag is used to describe exceptions that the method may throw.
- `@see`: This tag is used to provide a reference to other classes or methods that are related to the current one.

You can also include **Javadoc comments for attributes** (fields) in your Java classes. Here's an example of a Javadoc comment for an attribute:

```
public class Person {
    /**
     * The name of the person.
     */
    private String name;

    /**
     * The age of the person.
     */
    private int age;
}
```

In this example, the Javadoc comments for the name and age attributes provide a brief description of each attribute.

However, note that Javadoc comments for attributes are not as common as Javadoc comments for methods, since attributes are typically private and accessed through getter and setter methods. However, if you have public attributes in your class, it can be helpful to include Javadoc comments for them to provide documentation for users of your class.

In addition, provide a **general description of a class using Javadoc**, follow these steps:

1. Write a comment block immediately before the class declaration that begins with the `/**` Javadoc opening tag.
2. Write a brief summary of the class's purpose and functionality on the first line of the comment block, followed by an empty line.
3. Write a more detailed description of the class on the following lines, using complete sentences and proper punctuation. You can also include information about the class's properties, methods, and relationships to other classes.
4. Use Javadoc tags to annotate the comment block with additional information, such as the class's author, version, or usage instructions. For example, you can use the `@author` tag to specify the name of the class's author.
5. Close the comment block with the `*/` Javadoc closing tag.

```
/**
 * This class represents a car object that can be used to simulate
 * driving scenarios.
 *
 * The object car has properties such as make, model, year, current
 * speed, and methods such as accelerate, brake, and turn. It also
 * has a relationship to the driver object, which can control its
 * movements and behaviour.
 *
 * @author My groupmates and me
 * @version 1.0
 */
public class Car {
    // class implementation goes here
}
```

The package lab3 itself is a directory. **The description of a package can be included in a special file called `package-info.java`**. This file should contain package-level documentation that can be used to provide an overview or description of the package.

The `package-info.java` file should be located in the same directory as the Java source files for the package it describes. The file should have the following format:

```
/**
 * Package-level documentation goes here.
 */
package complex;
```

In this example, the package-level documentation is enclosed in a multi-line comment block and should provide a brief overview or description of the package. When you run the Javadoc on the package, the documentation from the `package-info.java` file will be included in the generated documentation for the package.

Please refer to the Javadoc documentation to understand how to use other comments not included in this document:

https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html