**Javadoc**

Java, as with many programming languages has standard comments that allow developers to understand and maintain their code. One of the main principles of OO programming languages is that Classes (which form templates for Objects) are written to wrap up behaviour and therefore become reusable by other developers.

The [Java Standard Edition API](https://docs.oracle.com/javase/8/docs/api/) contains thousands of classes available to any Java developer. The API documentation, as you will have discovered in previous exercises, tells developers how to use classes, without them having to view or understand the original source code.

You can write similar documentation for your own classes that can then be made available to other developers or used by yourself to understand how your classes public interface (i.e. constructors/methods) works.

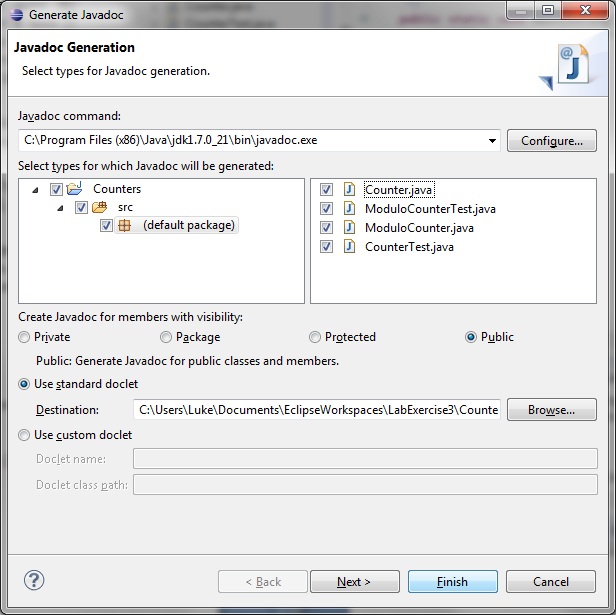
For very simple classes, it may seem a bit 'overkill' to be spending more time writing documentation than it takes to write code - this can indeed sometimes be true and you are not expected to write Javadoc for every class you create during your time on this module. You do however do need to appreciate how to write Javadoc, its purpose and how it can actually assist in understanding your own design as it forces you as the developer to consider how the public interface of your class should behave and be used.

***Notes - Writing Javadoc comments****:*

* A standard inline comment: //my comment
* A standard multi-line block comment: /\* my comment \*/
* A Javadoc comment: /\*\* my comment \*/
* For any class, if you wish to write Javadoc comments, you should place them directly above the **class header**, as well as each **constructor** and **method**.
* You do not need to write Javadoc comments for your fields as they are set to be private and thus hidden from the outside world.
* The **class header** Javadoc should give a brief overview of what the class does and how it behaves.
* The **constructor** Javadoc should state how the class is initialized, i.e. its fields.
* The **methods** Javadoc should state how the method behaves and how it should be used.
* Javadoc **tags** assist in the auto-generation and formatting of the HTML Javadoc.
* Notable tags include: **@author**, **@param** and **@return**.

*Please turn over...* To help you to better understand the relevance of Javadoc:

1. Within Eclipse go to and select the menu item: **Project > Generate Javadoc**. You will be faced with a dialogue window such as this:



Here you can select the Java project and/or individual Java class files that you wish to generate Javadoc for. Ensure all of your Counter files are selected. You may also need to click on the **Configure...** button and locate the javadoc.exe file on your system if it is not already specified - it should be in the Java jdk bin folder, within Program Files. When you press Finish, it will generate **HTML** Javadoc files for you and place them within a **doc** folder within your project.

1. You will see there are many HTML files - if you go to **index.html**, you will be able to choose any of your classes and look at the corresponding documentation that has been generated. There is no need to write Javdoc for your test files, but you will notice there is some documentation in the Counter class, as this was already written when you downloaded it from Blackboard.
2. Go back to the **Counter Java class file** and see if you can work out how it has generated the HTML Javadoc from the comments listed within the class.
3. If you have not already done so, try to write an appropriate Javadoc comment for the **decrement()** method you added to the original Counter class and then generate the HTML Javadoc again and look at the changes.
4. If you have attempted it, try to write some simple Javadoc comments for another class and then regenerate the Javadoc again to view the changes.