# Things I'm Likely to Forget (double check)

## JavaDoc

Place before public classes and methods to:

- **Definition:** Explain what the class or method represents or does
- **@param**: to list parameters the method accepts
- **@return**: description of what the method returns and what it represents

```
/**
* This method encodes a given String to its consecutive ASCII values
* and returns these values stored in an integer array.
* @param: nameString The String to be converted
* @return: An integer array containing the ASCII values of each character
* in the given String
*/
```

## Method Signatures

### Visibility

- **Public**: Any other class may access the method
- **Private**: Method is private to this class (it is hidden from other classes)
- **Protected**: Method can **only** be used by subclasses

### Static and non-Static

- **Static**: The parent class does not have to be instanciated to call this method
- **Non-Static** (not declared): This method can only be called on a live instance of its parent class

### Return type

String, int, double, String[], int[], bool, etc.

### Parameters

Consist of the type and identifier.

## Scope

There are two types of Scope in Java: Block Scope and Method Scope.

### Block Scope

Variables in the body of Java code have block scope. They can only be accessed inside the block (the `{curly braces}`) in which they were **created**. To give a variable "global" scope, therefore, it has to be initialised inside the class.

> In the exceptional case of `for` statements, variables declared in the expression itself are also available inside the block's scope.

### Method Scope

A variable declared inside a method can be used anywhere in that method.

## Pass by Reference vs. Pass by Value

They like to confuse you with this one! Make sure you take time to double check which variables are local to which code blocks. If a piece of code is trying to access a variable that it cannot see, the compiler will throw an exception and the code will not compile!