9.8 SYNTAX QUICKGUIDE

9.8.1 BEFORE YOU GO ON

If you are reading this *before* you have done objects and classes (chapters **10 (https://canvas.swansea.ac.uk/courses/44525/modules/301393)** onwards) you should be putting the keyword **static** in front of all these examples.

9.8.2 Method Syntax

The basic syntax of a method is:

```
returnType name(type argname1, type argname2,...) {
        ..code
}
```

where:

returnType is the type of data returned by the method - e.g. String, int, etc. If the method does not return anything this should be the keyword void.

name is the name chosen for the method - the naming style is the same as for variables (i.e. start with lowercase letter).

in the (..) following the name are zero or more *arguments*

arguments are a pair - type representing the type of the argument (String, int etc.) and name is the name that will be used in the body of the method.

9.8.3 return and void

- Methods that are not void need to return a value of the correct type - e.g. a method String methodName(...) {... must contain at least one statement like: return value; where value is a String (or variable of type String).
- Methods can contain multiple return statements (don't overdo it - it's harder to read).
- Methods that are void do *not* need a return statement - if they have one it should just be the keyword return; (with no value). There can be multiple return statements (again don't overdo it).

9.8.4 Guidelines for Methods

More or less the same as the list in **section 9.6 (https://canvas.swansea.ac.uk/courses/44525/pages/9-dot-6-guidelines-for-methods)** :

- **Repeated Code** – if you have repeated code, make it a method (but this is NOT the only reason to have methods!)
- **Single Task** – *methods should do one thing*: if your explanation of a method is "it does X and Y" then it should probably be two methods.
- **Keep Small** – methods should be *short*: a rule of thumb is a maximum of 25 lines of about 80 columns of text (i.e. fits on the screen) - many good programmers write lots of *very* short methods.
- **Use Parameters** – methods that work only with data passed as parameters are easier to understand, more flexible and less prone to errors.
- **Not Too Many Parameters** – slightly contradicting the above, not *too* many parameters.
- **Logical Breakdown** – as you think about the problem you are trying to solve, and break it into parts, you can't go far wrong if you start by making each logical part a method at least as a starting point.