A – Starting Classes

# LOG Review

These are some very brief points regarding the concepts and skills listed in LOGs for this topic.

**OOP Basics**

* **Define the term "class" as used in OOP**
  + A class is a template that acts as the basis for creating objects and describes a) What it “looks like”, and b) How it “works”.
* **Explain the purpose of classes in computer programs**
  + Classes allow programmers to define their own complex data types, complete with their own internal data (fields) and behaviours (methods).
* **Describe the primary keywords used in declaring a class**
  + public/private – identifies whether the class is visible to other code; public means all other code can see this class, while private means
  + class – declares that this is a class definition (tells the compiler to interpret this as a new “classification” or “kind” of data type; that is, a complex )
* **Define the term "method" and give an example**
  + A method is a set of instructions that manipulate information.
* **Create simple classes with methods only** 
  + *Demonstrate your ability to meet this learning outcome guide by completing the related assignments.*
* **Explain what is meant by "overloading"**
  + Overloading describes creating more than one method with the same method name (but different parameters) in the same class. Overloaded methods allow for alternate ways to perform the same general behaviour of an object or a class.
* **Describe the syntax of a method declaration (aka, method implementation)**
  + …
* **Explain how classes with identical names can be distinguished from each other** 
  + Different classes can have identical names provided that they exist in different namespaces. (Namespaces help to prevent a “collision” or confusion on the part of a compiler when confronted with two or more classes with the same name.)

**General Programming Concepts and Terms**

* **Define the term "keyword" as it applies to programming languages**
  + A keyword is a special command or instruction that is part of (that is, built in to, or “intrinsic” to) a programming language. Keywords are “reserved” words, meaning that they cannot be used for other purposes, such as being a name of a class, a variable, or a method.
* **Define the term "identifier" as it applies to programming languages**
  + An identifier is a programmer-supplied name for some item, either a class name, a field or local variable, or a method.
* **Define the term "program statement" and identify how the computer recognizes a program statement**
  + A program statement is combination of keywords, identifiers, punctuation, and/or literal values that, together, represent a single “instruction” or “command” that a compiler can translate into executable code.
* **Define the term “syntax” as it applies to programming languages.**
  + Syntax refers to the rules of a programming language that define how to combine keywords, identifiers, literal values, and other symbols to create specific instructions for a computer program.
* **Identify the entry point for every computer program**
  + Every computer program must supply a method called “main” as the method that the operating system will call (execute) to begin running the program.
* **Perform simple output to the console using System.Console**
  + *Demonstrate your ability to meet this learning outcome guide by completing the related assignments.*
* **Identify the basic structure of a simple C# program**
  + A C# program must consist of at least one class and have a single “main” method as the program’s entry point. Typically, a C# program will consist of a “driver” (the class that has the main method) and other classes used to represent objects (data) in the program.
* **Explain what is meant by a "driver"**
  + A “driver” is a method or set of methods that “run” or control the execution of a program; it’s responsibility is to direct and maintain the “lifetime” of a program, and makes use of classes and objects to achieve the purpose of the program. Every driver begins its execution in a method called “main” (which is the name given for the entry point for every computer program).
* **Explain what is meant by a “case-sensitive” programming language**
  + A case-sensitive language is one where the compiler regards “identical” words as different if they differ only in their case. In other words, a case-sensitive language will regard two identifiers are distinct if the only difference is whether they are spelled with upper- or lower-case characters. For example, HelloWorld and Helloworld will be regarded as different because one has a capital “W” and the other has a lower-case “w”.
* **Explain what is meant by a "strongly-typed" programming language**
  + A “strongly-typed” programming language is one that requires all variables or objects to always match or conform to whatever data type they were originally declared as. Some programming languages (such as C#Script) allow variables or objects to “change” their data type whenever a programmer wishes to use it for other purposes; strongly-typed languages do not allow variables to “change” the type of data they can store. Strongly-typed programming languages provide better security and make it more difficult for programmers to inadvertently corrupt, mangle or expose important data.
* **Explain what "string concatenation" is and how it works**
  + String concatenation is where two or more strings are “combined” to create a longer string; this is represented in C# by “adding” strings together. For example, given two literal strings whose values are “One” and “word”, the result of “One” + “word” would be the string “Oneword”.
* **Define and distinguish the terms “argument” and “parameter”**
  + A parameter is a variable declared in the parameter list of a method declaration; parameters are used to hold or capture information (values) that are sent or passed into the method. An argument is some value (either a literal value, a value stored in a variable, or some other value that comes from an “expression”) that is sent (passed) into a method; arguments appear in the syntax of method calls.
* **Use single-line, multi-line and XML comments as a means of documenting code**
  + *Demonstrate your ability to meet this learning outcome guide by completing the related assignments.*
* **List the four pieces of information to include in comments at the top of every source file**
  + The file name, author, creation date, and purpose of the file.

**Intro to IDEs**

* **Define the term "IDE" as it relates to software development tools**
  + The acronym “IDE” stands for “Integrated Development Environment”. An IDE brings together separate, but complementary, software development tools that aide developers in creating computer programs. For example, most IDEs bring a sophisticated text editor together with file management tools, a compiler, a debugger, and a program execution environment, as well as other useful tools.
* **Define the terms "project" and "solution"**
  + A solution is a folder in which one or more projects can be managed and “kept together”; it is the “area” in which a developer can create projects. A project is as set of classes designed to achieve a particular purpose; a project may represent a complete program in and of itself, or may be a single component of a larger, more complex computer program.
* **Identify the various parts of the IDE (Integrated Development Environment) used in this course**
  + [-- image here --]
* **Create a new project in the IDE for this course**
  + *Demonstrate your ability to meet this learning outcome guide by completing the related assignments.*
* **Create new source files in the IDE for this course**
  + *Demonstrate your ability to meet this learning outcome guide by completing the related assignments.*
* **Add existing files to a project**
  + *Demonstrate your ability to meet this learning outcome guide by completing the related assignments.*