B – 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 "object" as used in OOP**
  + An object is an instance of a class.  
    Each object has its own “state” (values) for its fields, and all objects based on any given class are distinct from each other. The internal values (state)for an object’s fields can be changed independently of other objects of the same class.
* **Describe the syntax of static and non-static method declarations**
  + Static methods are “shared” by all instances (objects) of a class, and can be called without even having to have an instance of the method’s class.  
    Non-static methods, on the other hand, can only be called for a given object (instance of a class) and will “act on” (or “act on behalf of”) that object (and any other objects are not affected by the given object’s method call).
* **Compare and contrast the syntax for method declarations (implementations) vs. method calls (for both static and non-static methods)**
  + …
* **Define the term "field" as used in OOP and give an example**
  + A field is a variable that belongs to a class and is shared by all the methods of a class.   
    private static int sharedCount; // a static field  
    private String name; // an instance field
* **Compare and contrast instance and class members, for both methods and fields**
  + Static methods are shared by all instances of a class, and can be called independently from instances of a class, whereas instance methods can only be called if you have an object (an instance of a class).  
    Static fields are shared by and fully accessible to all methods, but non-static fields, also called “instance” fields, are only shared by other non-static methods. In the following example, the first field is a static fields and the second is an instance field.
* **Identify the "member access operator" and explain it's purpose**
  + The member access operator (also called the “dot” operator) is the period. It is used to access a field or method of an object or a class.
* **Create (instantiate) objects**
  + *Demonstrate your ability to meet this learning outcome guide by completing the related assignments.*
* **Distinguish between classes and objects in OOP**
  + A class is merely a “template” for creating objects and describes a) what the object will “look like” (that is, what fields or data it will contain) and b) how the object works (that is, what methods can be called and what those methods will do).
* **Explain what is meant by the phrase “type extensibility” and how classes represent “complex” data types in OOP**
  + “Type extensibility” is where a programming language allows developers to create or design their own “data types”.  
    All programming languages support the “simplistic” data types (for numeric information, such as whole and real numbers, as well as simple text or character representation). These simple data types are intrinsic (“build into”) the programming language. A language that supports “type extensions” simply allows the programmer to create or design new data types; these new types are typically “complex” types made up of zero or more other (either simple or complex) data types.

**General Programming Concepts and Terms**

* **List the three categories of simple information**
  + Textual, numeric, and conceptual
* **Define the term "intrinsic data type"**
  + An “intrinsic data type” is one that is built into the programming language.
* **Explain what an assignment statement is and how it works**
  + An assignment statement is an instruction to the computer to store a value into a specific variable (or object). The variable (or object) is placed on the left-hand side of the equals sign, and the value to be stored in the object is placed on the right-hand side of the equals sign.  
    The left-hand side of an assignment statement must be a single variable, whereas the right-hand side can be any expression as long as the resulting value of the expression is the same (or compatible) data type as the variable on the left-hand side.
* **Compare and contrast the three categories of simple information with the complex types of information which classes represent**
  + Simple information is either textual (such as a name), numeric (such as a total or count), or “conceptual” (that is, representing a more “abstract” value that is neither textual or numeric). An example of a textual type in C# is the **char** type. An example of a numeric type in C# is the **double** type. An example of a “conceptual” type is the **boolean** type. A variable whose type is one of the simple types can only hold a single, simple value at any given time.
  + Complex types of information can be constructed from many simple types or from simple and complex types. A “class” is used to describe or define what the complex type looks like and how it works. Complex types are the basis for creating objects, and while any given “object variable” can only reference a single object, that object may consist of many distinct “values”, each one being stored in a field that is “hidden” (or “encapsulated”) inside the object.
* **Explain the advantages of using classes and objects instead of only using intrinsic data types**
  + By using objects, the programmer is no longer limited to having to only deal with simple values. By designing and using complex types (classes), the programmer is able to create complex programs using less code in a way that is simpler and more reliable than if the programmer were only allowed to use the simple types built into the programming language.