.NET and C# Programming Review

.NET and C#

- C# is used to create programs that run inside of the .NET Environment and can (optionally) interact with other .NET compatible programs easily.
- C# is a pure object-oriented programming environment where ALL data types are objects.
- Many of C#'s simpler data types are just .NET data types masquerading as basic C# types - i.e., "int" is the same as .NET's "Int32"
- C# programs are compiled into "Intermediate Language (IL)" - a low-level language which executes inside the .NET CLR (Common Language Runtime).
- Compiled C# code is stored in an "Assembly" file which typically has a name ending in "EXE" (for programs) or "DLL" (for libraries).
- The Stack is a region of memory where local variables and arguments live.
- Stack memory is automatically freed up when the corresponding variables go out of scope.
- The Heap is a (bigger) region of memory where objects live. Anything created by "new" lives on the heap.
- Periodically, the CLR's Garbage Collector discards objects that are no longer used from the Heap.
- The "using" statement lets you access classes from other class libraries in your program.
- Immutable objects cannot change after they have been created. Strings are immutable.
- Whenever you are assigning something of one type to a variable of a different type and information could be lost in the conversion, you have to include a cast.
- Use a "for" loop if you are counting the number of times thru the loop. Use a "foreach" loop if you doing something to all the items in an array/collection.
- Use a "do/while" loop if you have to go through the loop at least once, otherwise use "while/do" loop.
- In C#, all "case" sections in a "switch" statement must end with "break;"
- A method's name, return type, parameters and attributes are called its "signature."
- Method arguments can be passed "by reference" or "by value." If passed by reference, the original variable can be modified by the method it was passed to. If passed by value, a copy of the variable's data is sent.
- By default, all arguments are passed "by value." However, remember that a variable-that-represents-an-object actually contains a pointer to the location on the heap where the actual object lives. So when a variable-that-represents-an-object is passed into a method, the method actually gets a ***copy of that reference***. That copy points to the same object on the heap as the original variable and, if the object allows it, the method can use that copy of the reference to modify the object's properties and/or call its methods. And so, effectively, if you pass a variable-that-represents-an-object to a method, you are *effectively* passing it "by reference."
- If you have several methods with the same name but different parameter lists, those methods are called "overloaded" methods.

- Optional parameters have default values that are used if those parameters are not included in the method call. Optional parameters must be placed at the end of a method's parameter list.
- Enums let you create a type with a set of fixed, named values like Colors.
- If you want to store null in a type that doesn't normally take null as a value (like an int), add a question mark to the end of the type. (e.g., "int?")