.NET and C#

ADJUSTING TO CODING

Visual Studio .NET

- Microsoft states Visual Studio .NET is the only development environment built from the ground up to enable integration through XML Web services.

Can be used to:

- Build the next-generation Internet.
- Create powerful applications quickly and effectively.
- Span any platform or device.

Multiple Languages

- Visual Basic .NET
- Visual C++ .NET
- Visual C# .NET
- Visual J# .NET.

Why C# over others?

- Introduced in 2001, C# offers a familiar syntax, that is attractive to C++ and Java developers
- Unique language constructs that offer *code-focused* developers a more elegant experience when developing applications for the .NET Framework.

Migrating from JBuilder to VS.NET

- Easy as 1,2,3
- A Few new hot necessary hot keys need to be remembered
- Small language changes from Java to C#

Visual Studio .NET Hotkeys

- **Start Without Debugging**
  Control + F5
- **Start**
  F5
- **Stop Debugging**
  Shift + F11

C#.NET	Java
Namespace 1.use {}	Packages 1. use as first line in file
Using	Include
XML Tags 1.use "///" followed by <summery>, <remarks>,<value>,<exception>, <param> a."tools->build comment web pages" will let you extract the xml comments, create a structured, hyperlinked set of HTML documents based on the XML	// or /* */ Comment
Int32.Parse	Ingeter.parse
bool	boolean
Array 1.must place square brackets before the variable name 2.multidimensional array int[,] table	Array 1.multidimensional array int[][] table
Inheritance 1.Methods have to be declared virtual to overwrite them completely 2.sealed	Inheritance 1.Methods are automatically declared virtual 2.final

Classes

- Changing a class does not change the name of the .cs file. Its is common practice to name the .cs file after the class
- When you write your own class constructor, remember the fields you don't initialize are still implicitly initialized to 0, False, or null
- **Naming**
  - Identifiers that are public should start with a capital letter
  - Those that aren't should start with lowercase letter

Naming Variables

- Don't use underscores
- Don't create identifiers that differ only by case
- Start name with lowercase letter
- Start subsequent word with an uppercase letter
- Don't use Hungarian notation
- If you leave the mouse pointer over a variable a ToolTip appears telling you the type of variable so there is no reason for Hungarian notation

Declaring Variables

- Must explicitly declare all variables before you can use them
- Can't assign one type of value to a variable of another type
  - Use float away = 0.42F;
  - Others are L for long and M for monetary
- Date type decimal holds monetary values
- Variables declared as part of a class rather than a method (function) are called fields

Naming Methods (Functions)

- C# does not support global methods (note to C, C++ and VB programmers)
- Must specify a return type of void
- If you don't feel comfortable with writing methods you can use the C# Method Wizard

Overloading Identifiers

- Only allowed to overload when the two methods have different number of parameters or the types of the parameters differ.
  - You can't overload the return type of a method.
- C# does not support default arguments however you can mimic default arguments using overloaded methods (note to C++ and VB programmers)

Exception handling

- try/catch is used in the same way as in Java, C, C++
- System.Int32.MaxValue or System.Int32.MinValue lets you determine the max or min of int.
  - "checked" can be used and an OverflowException will be thrown if there is an overflow instead of silently overflowing.
  - "unchecked" can be used if you don't want it to throw an exception.
    - Floating point (non-integer) arithmetic cannot be controlled by checked/unchecked, not even when you divide by 0.0
- You can use "throw" to throw your own exception.
- If you need a segment of code to run even if there is an exception you can use "finally"
  - You can only write a finally block after the try block or immediately after the last catch handler after the try block.

ref and out parameters

- when you prefix parameter with "ref" the parameter becomes an alias for the actual argument rather than a copy
  - when you pass an argument to a ref parameter, you must also prefix the argument with the ref keyword
- out is similar but with out the method has to assign a value to an out parameter

Declaring Arrays

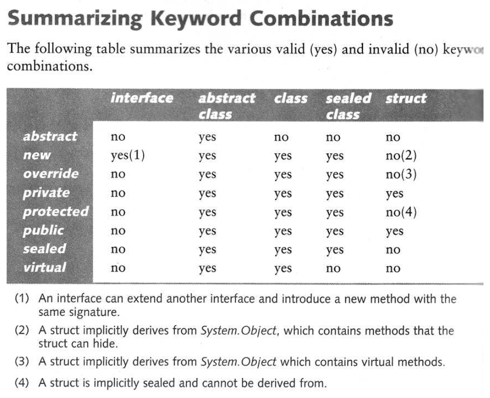
- Size not part of array declaration (note to C and C++ programmers)
- You must place the square brackets before the variable name (note to Java programmers)
- int[] numbers = new int[5] {1, 2, 3, 4, 5};
- int[,] numbers = new int[3, 2] {{9, 99}, {3, 33}, {5, 55}};

Arrays

- **ArrayList**
  - **Very useful when you need array that expands when you need to insert/remove from list**
    - **Same as <std:vectors> in C++**
    - **First-in First-out(FIFO)**
- **Stack**
  - **Push/pop same as in assembly**
    - **Last-in First-out(LIFO)**
- **SortedList**
  - **Elements held inside array must be comparable**
  - **No duplicates**
- **params prefix**
  - **allows you to set array parameter in method by using arguments**
- **c and c++ programmers may recognize params as a type-safe equivalent of the varargs macros from the header file stdarg.h**

Inheritance

- class DerivedClass:BaseClass, Interface, Interface etc
  - :base(parameters) calls base class constructor
    - only works if base class has a public base constructor



Summarizing Keyword Combinations

The following table summarizes the various valid (yes) and invalid (no) keywo combinations.

	interface	*abstract class*	*class*	*sealed class*	*struct*
abstract	no	yes	no	no	no
new	yes(1)	yes	yes	yes	no(2)
override	no	yes	yes	yes	no(3)
private	no	yes	yes	yes	yes
protected	no	yes	yes	yes	no(4)
public	no	yes	yes	yes	yes
sealed	no	yes	yes	yes	no
virtual	no	yes	yes	no	no

(1) An interface can extend another interface and introduce a new method with the same signature.

(2) A struct implicitly derives from *System.Object*, which contains methods that the struct can hide.

(3) A struct implicitly derives from *System.Object* which contains virtual methods.

(4) A struct is implicitly sealed and cannot be derived from.

Forms Application Suggestion

- Never modify the contents of the InitializeComponent method with the Code and Text Editor window, use the properties of components in Design View or in the Properties window or your code changes will be lost.

Quick Guide to Programming Languages

- TASK: Shoot yourself in the foot.
- C: You shoot yourself in the foot.
- C++: You accidentally create a dozen instances of yourself and shoot them all in the foot. Providing emergency medical assistance is impossible since you can't tell which are bitwise copies and which are just pointing at others and saying, "That's me, over there."
- FORTRAN: You shoot yourself in each toe, iteratively, until you run out of toes, then you read in the next foot and repeat. If you run out of bullets, you continue with the attempts to shoot yourself anyways because you have no exception-handling capability.
- Pascal: The compiler won't let you shoot yourself in the foot.

Continue..

- LISP: You shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds...
- Prolog: You tell your program that you want to be shot in the foot. The program figures out how to do it, but the syntax doesn't permit it to explain it to you.
- BASIC: Shoot yourself in the foot with a water pistol. On large systems, continue until entire lower body is waterlogged.
- Visual Basic: You'll really only appear to have shot yourself in the foot, but you'll have had so much fun doing it that you won't care.

Continue…

- Unix:
- % ls
- foot.c foot.h foot.o toe.c toe.o
- % rm * .o
- rm:.o no such file or directory
- % ls
- %
- Assembler: You try to shoot yourself in the foot, only to discover you must first invent the gun, the bullet, the trigger, and your foot.