*If you are not following the posted WFL BOCES Code of Conduct and Classroom Discipline Policy you are doing something very wrong!!!

TEN STEPS TO SUCCESS:

*Use this checklist **EVERY DAY** to monitor daily progress*

WITHIN THE FIRST MINUTE OF CLASS DID YOU?

BEGINNING

1. Enter The Room QUIETLY and sit in your assigned seat (if applicable).
2. Get in the game. Follow the classroom rules and expectations always!
3. Log into the course Moodle at www.cyberLearner.com , check and work on ***TODAY'S*** agenda by starting with the warm-up (check-in).

FOR THE DURATION OF CLASS DID YOU?

MIDDLE

4. Listen to any instruction the first time and remain quiet.
5. Raise your hand if you have an appropriate question.
6. Complete all the day's activities in the order given staying on task at all times.
7. When this day's activities are complete work on pending items that also need to be completed. Examples: Meyers readings; TestOut; Glossary Entries.

BY THE END OF CLASS DID YOU?

END

8. Check your grades in our Moodle for any work that may be outstanding.***
9. Make sure to ask a classmate then your instructor if you need help! Make sure to respond to a classmate when they are seeking help.
10. If you complete the first nine steps work on your assigned job—if you have no assigned jobs see Mr. Calaman.

*****QUESTIONS YOU SHOULD BE ASKING YOURSELF ABOUT YOUR GRADES:**

- What is my average for the current marking period?
- Are all my quizzes and tests taken and graded?
- Have I checked my email for any important information (you need email to do this)?
- What work do I need to do (zero's or – that can still be completed)?
- **IS EVERYTHING I CAN DO DONE?**

If the answer to the last question is YES then you can do an extension project while you are ahead of the rest of the class! If you are ready to do extensions see Mr. Calaman. You will be responsible to be ***ON TASK*** with your extension assignment and doing only the things you are supposed to be doing!

Moodle

PROBLEM SOLVING PRIMER

You must do these things to be successful on a consistent basis in solving problems:

Be careful -- take care to pay attention to what you need to do and what you are doing.
Take your time -- don't rush. When people are in a hurry mistakes are made and important things are missed.
Verify your solutions -- use every opportunity to verify your solutions. Ask yourself "does it make sense". Make sure to look for concrete ways to verify. If there are no concrete solutions look for other identifiers that give a clue if you are on the right track.

Follow these steps to solve problems:

1. Read the entire problem if it is written. Take your time to fully understand the problem in its entirety. Make sure to identify the proper problem else you will have NO CHANCE of finding a solution that works. Having trouble? List the answers to these questions:
    a. What are the symptoms of the problem?
    b. What caused the problem?
2. List the words/concepts/ideas/functions you don't understand
3. Read/identify the problem again and answer the following questions:
    a. What is the problem asking you to do?
    b. What information do you need to know?
    c. Is there information you don't need to know?
    d. What materials do you need to solve the problem?
    e. What approach is appropriate to solve the problem?
    f. What is my desired outcome?
4. Read the problem again and again until you understand what is being asked and how you should go about solving it. You should be trying to answer:
    a. What are my options?
    b. What are the consequences of my options?
5. Choose and implement a solution.
6. Complete any verification steps you are able to complete to assure you have chosen the best solution.

PROCESS TO CREATE A C# PROGRAM/CLASS

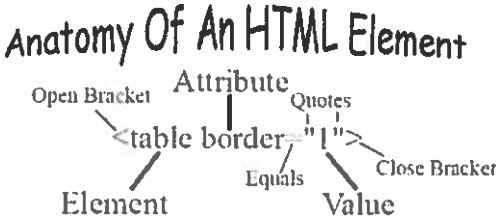
1. Create project/files.
2. Use your brain to think through and understand fully what you need to do. Use a graphical organizer and/or structured English if necessary.
3. Program instance variables.
4. Program properties.
5. Program constructors.
6. Program methods.
7. Create a test class with a Main() method and TEST, TEST, TEST everything you programmed in the class you need to test.

HTML Elements

An HTML element is everything from the start tag to the end tag:

HTML Element Syntax

- An HTML element starts with a **start tag / opening tag**
- An HTML element ends with an **end tag / closing tag**
- The **element content** is everything between the start and the end tag
- Some HTML elements have **empty content**
- Empty elements are **closed in the start tag**
- Most HTML elements can have **attributes**



HTML Attributes

These are used to modify the behavior of the elements and come in the form of name value pairs.

<element attribute="value">

Example: <table border="1">

Elements can have multiple attributes like this:

<element attribute="value" attribute="value">

Example: <table border="1" cellpadding="1" cellspacing="2">

HTML and Cascading Style Sheets

Style Precendence:

1. *Inline:*
```
<p style="color: sienna; margin-left: 20px"> This is a paragraph </p>
```
2. *Internal:*
```
<head> <style type="text/css">
<!–
hr {color: sienna}
p {margin-left: 20px}
-->
</style> </head>
```
3. *External:*
```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```
4. *Browser Default:* this is the set of rules the browser uses to display content.

Three Parts of Styles: selector {property: value}

Multiple Properties Separated By Semi Colon

```
selector { property1: value1;
           property2: value2;
           property3: value3 }
```

A Value With More Than One Word In "Quotes"

```
p {font-family: "sans serif";}
```

Group Selectors with Comma's:

```
h1,h2,h3,h4,h5,h6 {color: #ffffff;}
```

Class Selectors:

```
p.italic {font-style: italic;}
p.bold{font-weight: bold;}
```

Is called with:

```
<p class="italic">This paragraph will be in italics.</p>
<p class="bold">This paragraph will be bold.</p>
```

A Class Selector for any Element:

```
.center {text-align: center}
<h1 class="center"> This heading will be center-aligned </h1>
<p class="center"> This paragraph will also be center-aligned. </p>
```

IP Address Classes

Class A	1 – 127	(Network 127 is reserved for loopback and internal testing) Leading bit pattern	0	00000000 00000000 00000000 00000000 Network Host Host Host
Class B	128 – 191	Leading bit pattern	10	10000000 00000000 00000000 00000000 Network Network Host Host
Class C	192 – 223	Leading bit pattern	110	11000000 00000000 00000000 00000000 Network Network Network Host
Class D	224 – 239	(Reserved for multicast)		
Class E	240 – 255	(Reserved for experimental, used for research)		

Private Address Space

Class A	10.0.0.0 to 10.255.255.255
Class B	172.16.0.0 to 172.31.255.255
Class C	192.168.0.0 to 192.168.255.255

Default Subnet Masks

Class A	255.0.0.0
Class B	255.255.0.0
Class C	255.255.255.0

TCP/IP	OSI Model	Protocols
Application Layer	Application Layer	DNS, DHCP, FTP, HTTPS, IMAP, LDAP, NTP, POP3, RTP, RTSP, SSH, SIP, SMTP, SNMP, Telnet, TFTP
	Presentation Layer	JPEG, MIDI, MPEG, PICT, TIFF
	Session Layer	NetBIOS, NFS, PAP, SCP, SQL, ZIP
Transport Layer	Transport Layer	TCP, UDP
Internet Layer	Network Layer	ICMP, IGMP, IPsec, IPv4, IPv6, IPX, RIP
Link Layer	Data Link Layer	ARP, ATM, CDP, FDDI, Frame Relay, HDLC, MPLS, PPP, STP, Token Ring
	Physical Layer	Bluetooth, Ethernet, DSL, ISDN, 802.11 Wi-Fi

C# Course I Study Guide

Data Types: data types determine memory allocations to variables, return types and parameters. Know char, Boolean, int, float.

JScript value type	.NET Framework type	Storage size	Range
boolean	Boolean	N/A	**true** or **false**
char	Char	2 bytes	Any Unicode character
float (single-precision floating-point)	Single	4 bytes	Approximate range is -3.4E+38 to 3.4E+38 with accuracy of about 7 digits. Can represent numbers as small as 1E-44.
Number, double (double-precision floating-point)	Double	8 bytes	Approximate range is -1.79E+308 to 1.79E+308 with accuracy of about 15 digits. Can represent numbers as small as 1E-323.
decimal	Decimal	12 bytes (integral part)	Approximate range is -7.9E+28 to 7.9E+28 with accuracy of 28 digits. Can represent numbers as small as 1E-28.
byte (unsigned)	Byte	1 byte	0 to 255
ushort (unsigned short integer)	UInt16	2 bytes	0 to 65,535
uint (unsigned integer)	UInt32	4 bytes	0 to 4,294,967,295
ulong (unsigned extended integer)	UInt64	8 bytes	0 to approximately 1.8E+19
sbyte (signed)	SByte	1 byte	-128 to 127
short (signed short integer)	Int16	2 bytes	-32,768 to 32,767
int (signed integer)	Int32	4 bytes	-2,147,483,648 to 2,147,483,647
long (signed extended integer)	Int64	8 bytes	Approximately -9.2E+18 to 9.2E+18
void	N/A	N/A	Used as the return type for a function that does not return a value.

Naming conventions: as programmers we name things. The names we use are called 'identifiers'. Typically we use 'camel casing' to name things. For classes we name things in Pascal Case with all capital letters (such as 'CircleDisk'). For methods we name first words with lower case and then any subsequent words capitalized (such as 'moveToOtherWorld'). See the CamelCase poster for examples!

Variables: variables create memory space in RAM to hold values our program specifies. In this fashion the program can substitute values into things as represented by what value the variable holds currently in memory.

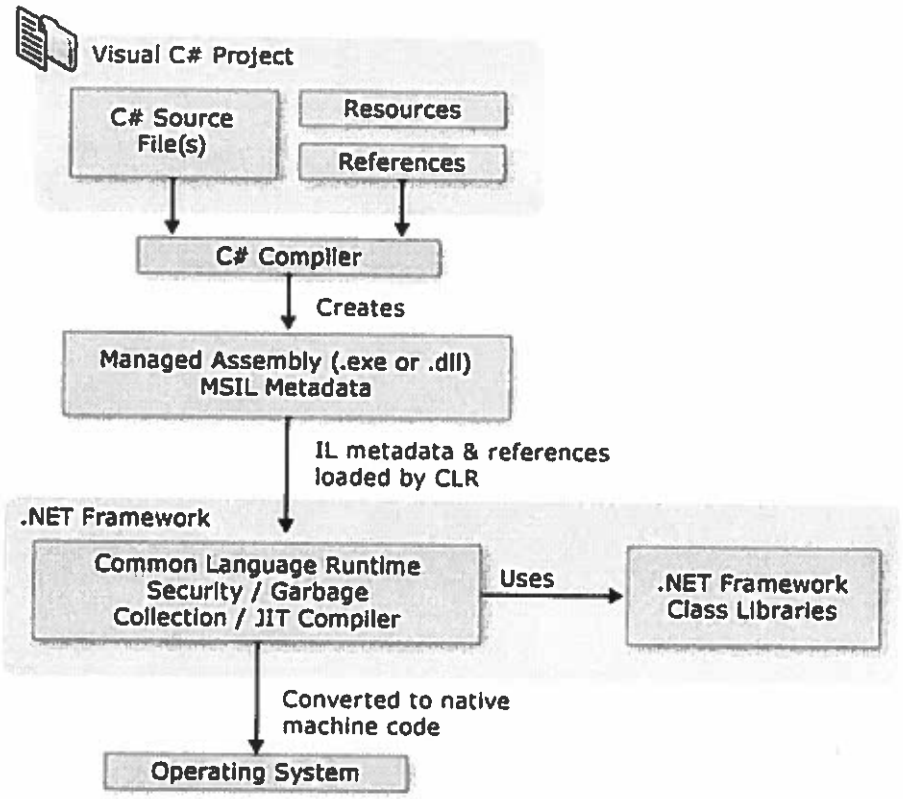
Constants: Constants are created with the const keyword. They are almost always static which means the constant is available to all classes in the program at all times. As a convention we usually name a constant with all capital letters (CONSTANT).

Software Versioning: Know that beta versions of software are the 'not ready for prime time' forefathers of officially released software versions.

Class: Classes are the definitions of objects in our program and contain fields (instance variables) and methods. There are also special methods—constructors-- that define how an object will be instantiated in memory. Constructors must be named the same as the class.

Method: Methods are groups of programming statements that define the behavior of the objects in a class.

C# Compiler: [1]



Errors: There are primarily two types of errors logic and syntax. Logic errors are when the program is really not doing what the programmer wants—so the program still works but it doesn't work the way the programmer wants it to work. Syntax errors, on the other hand, stop the compiler and throw build (compile) errors. These are things that must be fixed in order to get the program working at all!

Inheritance: inheritance is the way object-oriented sub-classes inherit components of the super-class such as methods.

IntelliSense: helper feature in Visual Studio that pops up tips, hints and/or suggestions when you are programming.

Console applications are ones that we operate from the commence line or console. There are special console commands. One of them is Console.WriteLine("Your Stuff Prints At The Command Prompt"). In the previous example Console is a Class and WriteLine is a method.

Windows applications have an entry point specified by the Main() method. This method appears once in the program and defines what instructions are executed when the program runs. When the last line of the Main() method is executed the program stops and awaits user input. Another key different between Windows applications and other applications is Windows Applications sit in a constant loop awaiting 'events' that trigger the program to behave in certain ways. This is why we say Windows is an 'event driven' model. How the program behaves in response to these events is programmed into special methods called 'event handlers'. Objects, such as buttons, menus and labels, that

[1] https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

can respond to user input via their respective events are called 'controls'. If a user clicks a button, for example, the code written into the buttons onClick event will be executed.

Escape Characters: All programming languages have an 'escape character' that tells the compiler to 'ignore' converting the character into anything other than the literal text it represents. In C# the escape character is '\'. If I want to print the escape character to the console I would have to type: Console.WriteLine("\\"); Can you see why this prints: \ ?

Integrated Development Environment (IDE): an IDE typically consists of an editor, compiler and debugger put together in a package to aid the programmer in writing code.

Properties: classes have a group of associated properties that can be read and mutated by the programmer. Some examples are:

- Text: What text displays with the class.
- Focused: if the object of the class is the currently active item on the screen.
- MulitLine: enables a TextBox to have more than one line.
- ReadOnly: disables a control from being modified by the user.

The increment operator (++) increments its operand by 1. The increment operator can appear before or after its operand: `++variable` and `variable++`.

The addition assignment operator (+=) adds the value after the += to the value before. So:

```
x += y
```

is equivalent to

```
x = x + y
```

Derived Class: The derived class inherits the properties of the base class, and you can add or override methods and properties as required.

Know how to convert decimal to binary and binary to decimal.

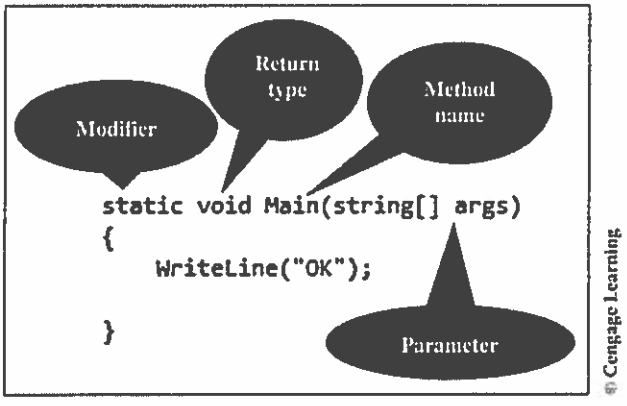


FIGURE 3-1 Method components

```
5     public class Student
6     {
7         //Data members, data fields, or characteristics
8         private string studentNumber;
9         private string studentLastName;
10        private string studentFirstName;
11        private int score1;
12        private int score2;
13        private int score3;
14        private string major;
15
16        //Default constructor
17        public Student()...
21
22        //Constructor with one argument
23        public Student(string sID)...
27
28        //Constructor with three arguments
29        public Student(string sID, string lastName, string firstName)...
35
36        //Constructor with six arguments
37        public Student(string sID, string lastName, string firstName,
38                       int s1, int s2, int s3, string maj)...
48
49        //Properties
50        public string StudentLastName...
61
62        public string StudentFirstName...
73
74        public string StudentNumber...
85
86        public string Major...
97
98        public int Score1...
```

FIGURE 4-3 Auto-generated code from Student class diagram