2015

Software Design & Build Fundamentals

LEARN TO DESIGN & BUILD A WINDOWS C# APPLICATION

STEPHEN O'CONNOR

Contact me:
stevo.joc@gmail
github.com/stevo5o
Stackoverflow Stevo5o

Developing Software is Fun!
Don't be intimidated … take a little at a time

Exercise files github
All exercise files are here https://github.com/Stevo5o/Software-Design-Fundamentals
This document location github.com/Stevo5o/Software-Design-Fundamentals.pdf

Tips on learning C#:
You learn by doing. As you watch the demos in class, try to follow what has been done and try re-create the project and the code. After the demo look back over this lesson plan or talk to class mates or do some research in regards to any questions you might have. Don't just mindlessly copy source code from a website or from this lesson plan. If something is not completely clear, take the time to use the Help feature or google to do some extra research.

Be patient. I began to teach myself programming back in September 2010. I didn't get my first real job using C# until June 2013. It took a lot of nights and weekends to learn it well enough to get a job. How long will it take you to learn? That depends on your level of commitment and your background in software development. However, after going through these lessons and demos, talking to class mates, as well as all the videos on LearnVisualStudio.NET, you should have a pretty rich understanding of how to develop software using C# and the Visual Studio.NET environment.

Find a project to do for a friend, a charity, a small local business. If you have a project in mind as you begin to learn, it will force you to learn things you otherwise might overlook. Once you are finished, you will inevitably feel ambivalent: you'll be so proud of what you've accomplished, and at the same time you will look back and realize how utterly pathetic that first attempt at programming was. Although I'm not a golfer, I understand there are similarities. If you are competitive and passionate, you'll constantly seek ways to improve your game. But even Tiger Woods is never completely satisfied, although he may be one of the best who has ever played the game.

Taken from stackoverflow
How long to learn C#?
That's like asking how long it would take to learn French:

- 1 day to learn what it is
- 1 week to learn it to an infant/elementary level
- 1 year to be considered a beginner by professionals
- Several years to be considered an experienced professional
- Plus there's "deep" knowledge of those subjects which a mere mortal such as you or I will never learn

Then again, plenty of people (most normal people, non-programmers) never learn those subjects, so if you're like "most" people then the answer would be "it would take forever" or "it will never happen".

Lessons

Workflow, Visual Studio Interface
Design Best Practices
Variables and Datatypes
Iteration and Selection Statements
Object Oriented Programming Fundamentals

The Set-up

1. Download and install Visual Studio Community 2013
2. Create GitHub account and create a new repository called Csharp-Basics
3. Download and install the latest version of GitHub for windows version 7, 8, 8.1
4. On your PC in users\username\documents create new vsprojects folder
5. Open PowerShell type following text

```
C:\Windows\system32> cd c:\
C:\>
```

Change directory (cd) to vsprojects folder and git clone into folder

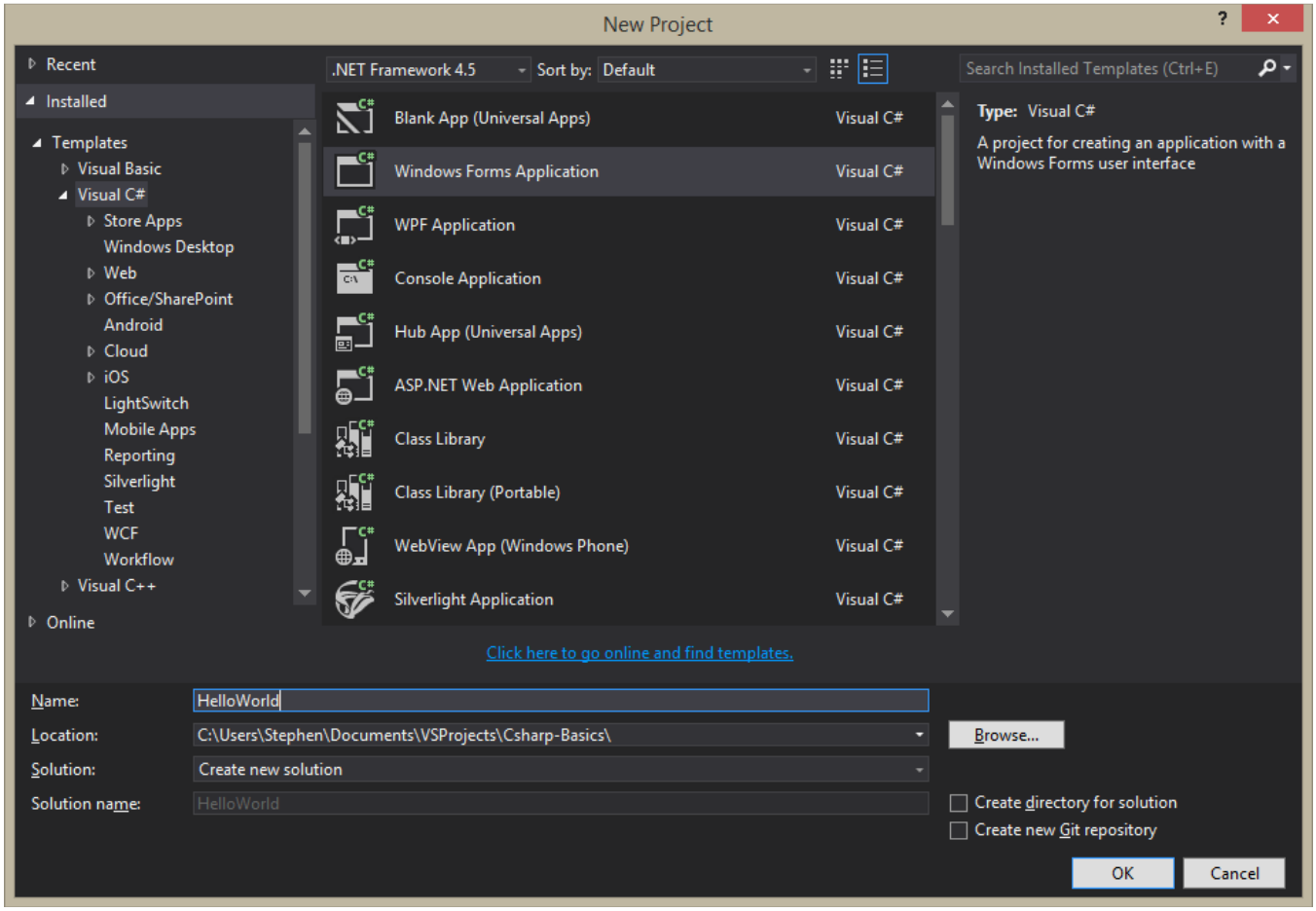
```
C:\> cd c:\users\username\documents\vsprojects
C:\users\username\documents\vsprojects> git clone "HTTPS clone URL"
```

```
C:\users\username\documents\vsprojects> dir
C:\users\username\documents\vsprojects> cd csharp-fundamentals
C:\users\stephen\documents\vsprojects\csharp-basics [master]> git version
git version 1.9.5.msysgit.0
```

Lesson 1

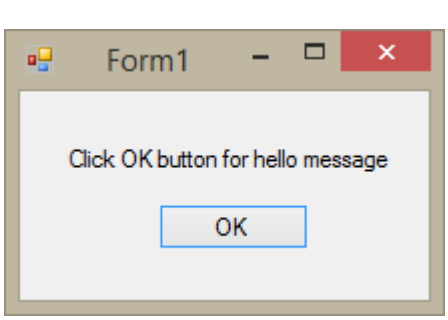
HelloWorld windows application. Open Visual Studio select new project. Select Visual C# and Windows form application. In the Windows dialog change name and location to:

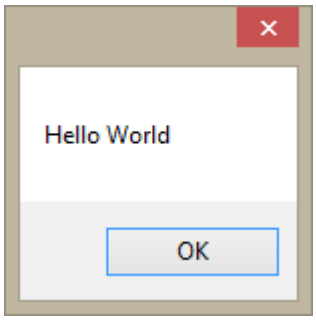
- Name: HelloWorld
- Location: click Browse button navigate to vsprojects\Csharp-Basics folder – GitHub clone
- Create new folder Lesson1
- Click OK button



Run debug click start button, this is a basic windows form click x and close. This is a complete windows app, however it does not do anything. Go to C:\Users\username\Documents\VSProjects\ Csharp-Basics\Lesson1\HelloWorld\bin\Debug and click HelloWorld.exe

Create a button that once clicked displays a message box that displays Hello World. Go to toolbox select button. Double click button and into Form1.cs type `MessageBox.Show("Hello World");`
Two tabs code design view and debug app using the start button.





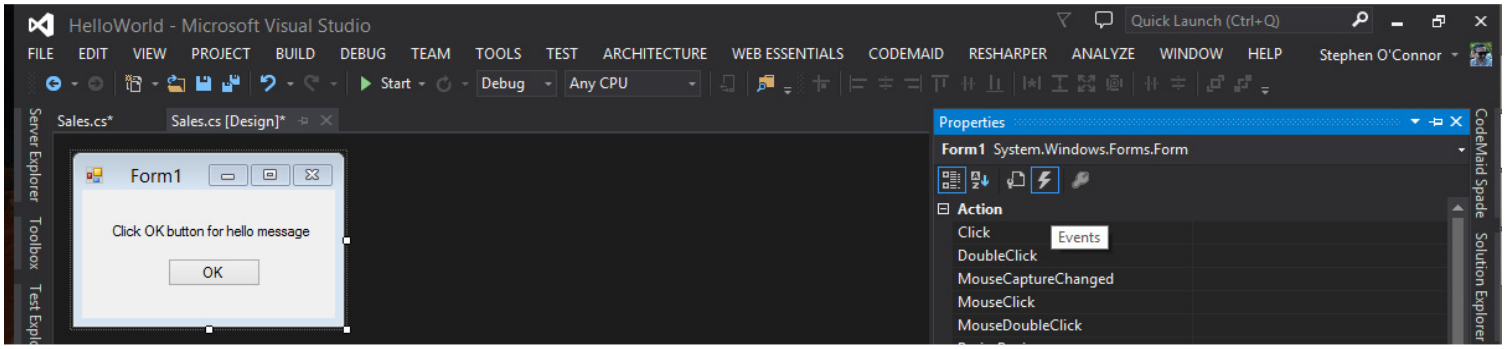
Solution explorer file description. Shut down reopen visual studio. Looking at Debugging. Set break points to step through each line of code.

What are Events? "Event Driven Programs"

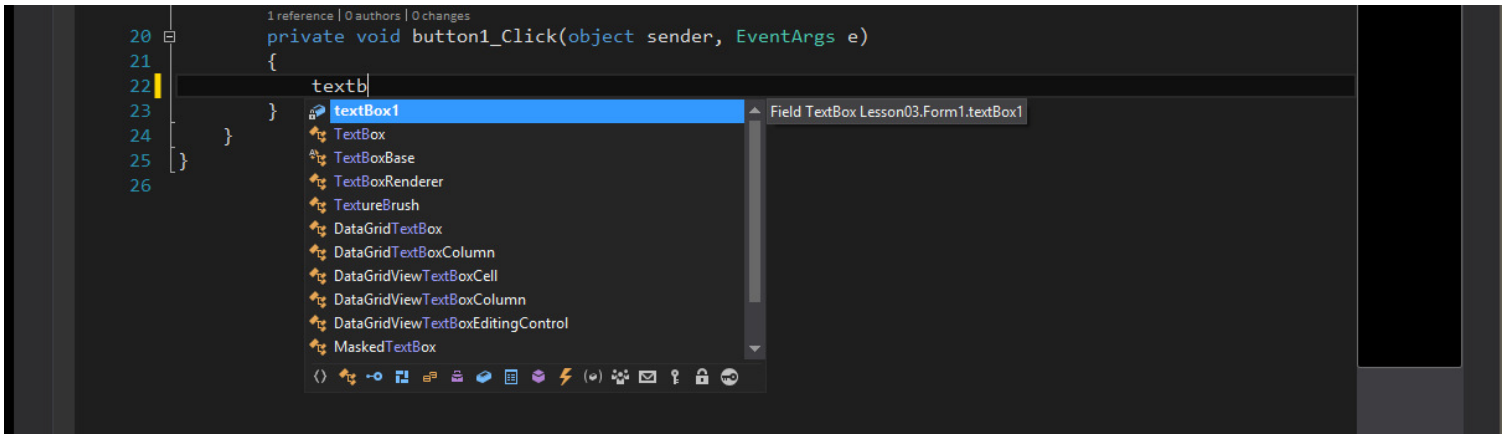
Response to events like open file, exit, new file, print file. Hundreds of events that an app can react to. Double click on the OK button an event handler is created and given a default name. When the event is triggered the end user clicks the button. Event handler event default name button1_Click Curly braces … { } ... define a bock of code. Events are triggered in an app. App can respond or ignore those events. Write code in Event Handlers to handle events. Code must be written inside of code block defined by curly braces. Methods are the basic building blocks of writing code. An event Handler is a more specific type of a method.

```
private void button1_Click(object sender, EventArgs e) // event handler
{
    MessageBox.Show("Hello World"); // event
}
```

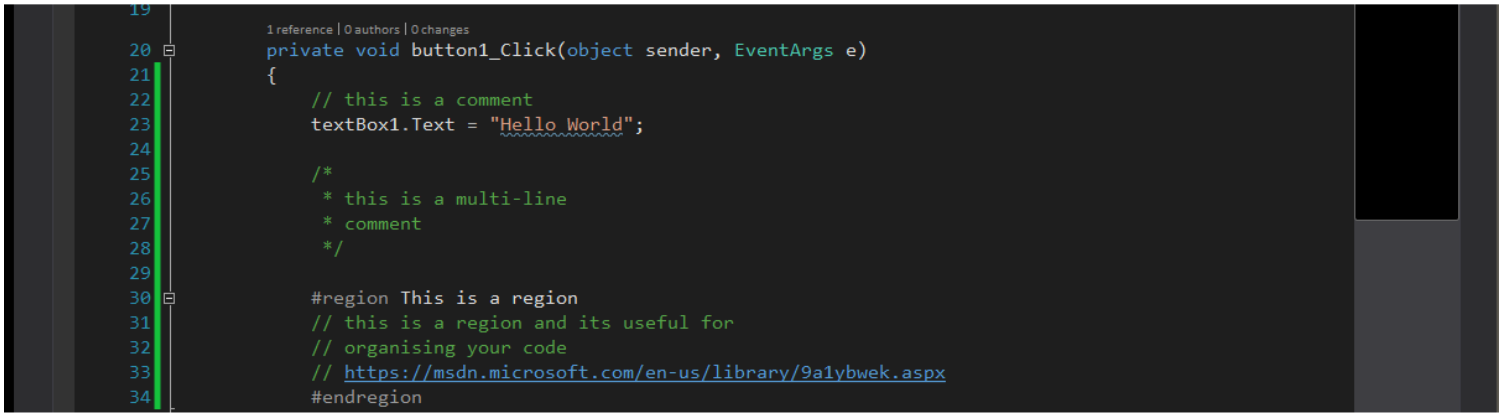
Events



IntelliSense



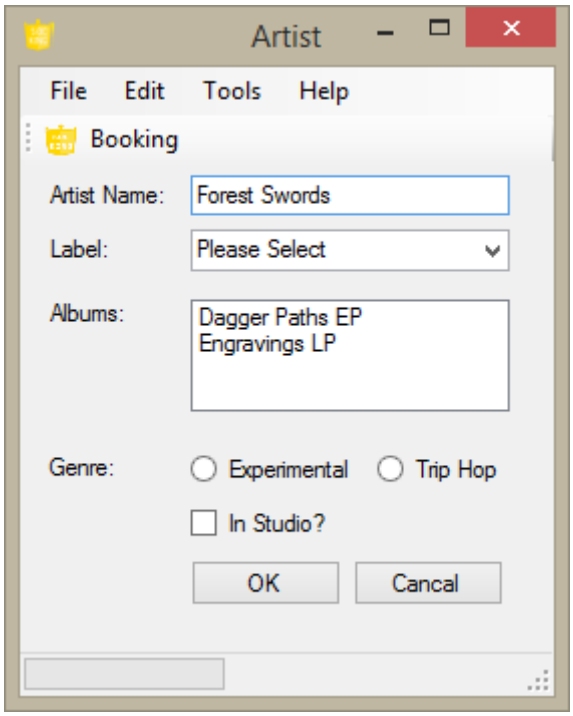
Comments and regions



Lesson 2

Design Best Practices

Arrange controls in columns and rows. Labels left input boxes, data entered on the right. Ok Cancel buttons on the right. Use standard fonts and colors. Keep it simple. Use standard, succinct descriptions (names) for controls. Don't make the user "think", easy for the user to use.



Buttons
Allow user to communicate a decision or to trigger some action.

Labels
Non-interactive descriptions or text usually displayed on the left-hand side of other controls.

Text Boxes
Allow for unstructured user input.

Check Boxes
Allow for yes /no or on /off type user input. Used together to allow off "Check all that apply"

Combo box
Text and list box combination. User can select item in the list, or type in a selection that is not in the list.

Menu strip
Add menu to app, select Insert Standard Items.

Tool strip
Add toolbar to app. Includes progress bars, textboxes, combo boxes & more

Status strip
Add status bar to app to provide feedback to the user

Tool Strip Container
Hosts other controls like the menu strip, Toll Strip and the Status Strip to provide user app customization.
Arrange toolbars top left bottom right

Tips
Make sure the right control is selected before making any changes in the Properties window.
Set tab order. Click VIEW and Tab Order click item to make first.
Tab Order starts with the control from the left-hand corner logically to the lower right-hand corner.

Lesson 3

Variables and Datatypes

x = 4
y = x + 6
What does y equal?

Variables represents a space in a computer's memory that is assigned to store a value. The name of the variable is then used in code to reference the value that is stored in that memory space. Declaring a variable is the act of allocationg space in he computer's memory for valus of a specific data type and giving the variable name.

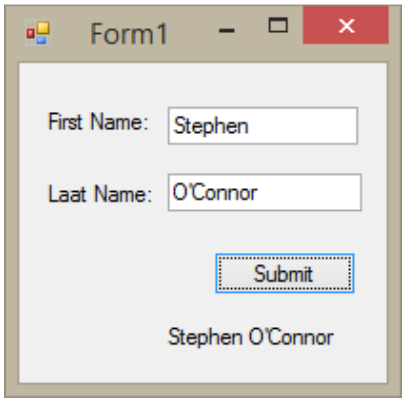
There are many available data types in C#, including ones that can store strings, dates, numbers, and more..

Three Basic Numeric data Types ..

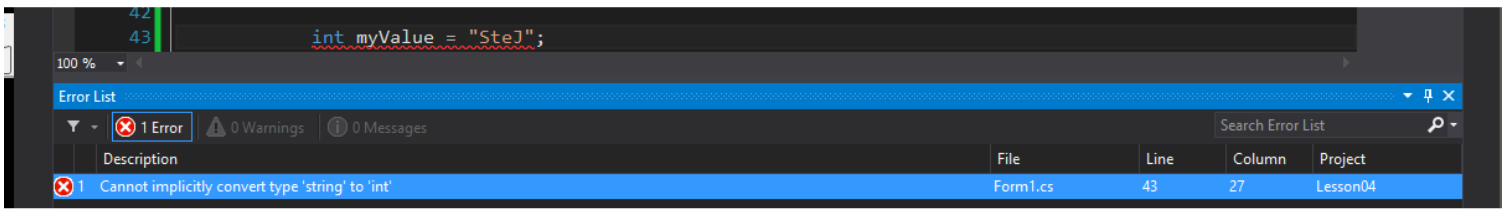
- Interger (int): -2,147,483,648 to +2,147,483,647
- Double (double): $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ up to 15 decimal places
- Boolean (bool): true or false

When writing C#, C# is case sensitive. Consider a variable a bucket and the string literal can go into the bucket.

```
string hello; // hello is the bucket
hello = "hello world"; // string literal goes into the hello bucket
MessageBox.Show(hello); // hello var
MessageBox.Show("hello"); // string literal

// declare two vars
string firstTextBox = textBox1.Text;
string secondTextBox = textBox2.Text;

label1.Text = firstTextBox + " " + secondTextBox;
```



Declare a Variable with the wrong a data type error with discription.

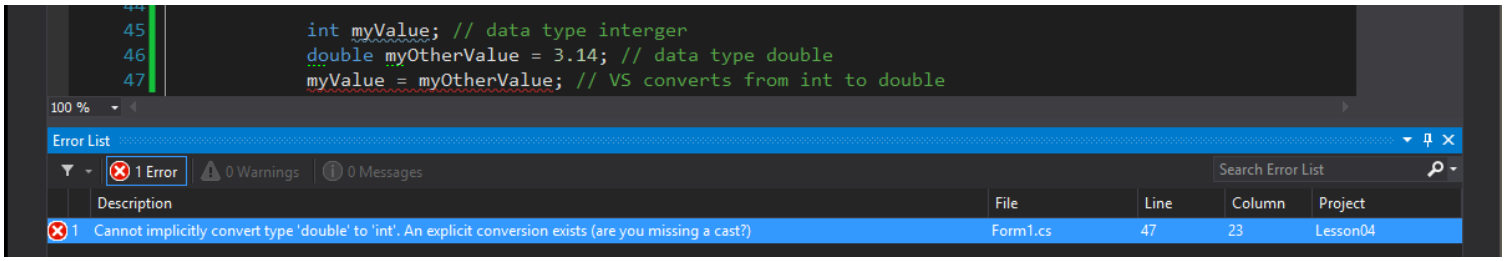


VS implictyly converts interger to double no loss of data

```
int myValue = 3; // data type integer
double myOtherValue; // data type double
myOtherValue = myValue; // VS converts from int to double
```

VS cannot convert double to interger loss of data

```
int myValue; // data type integer
double myOtherValue = 3.14; // data type double
myValue = myOtherValue; // VS cannot convert
```

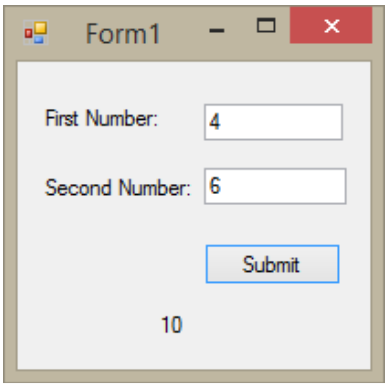


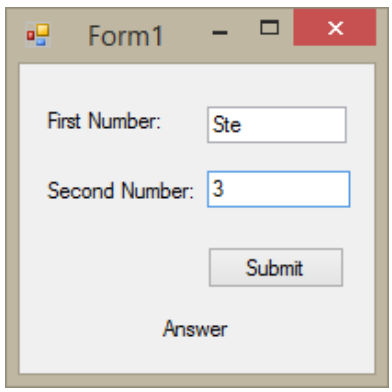
Create a Basic Calculator

To explicitly convert data types, lets look at the following example. With an explicit cast, either you are telling the compiler that **you know more than it does** - "please believe me, but check anyway": TextBox.Text cannot add two numbers and display the result in label1.Text

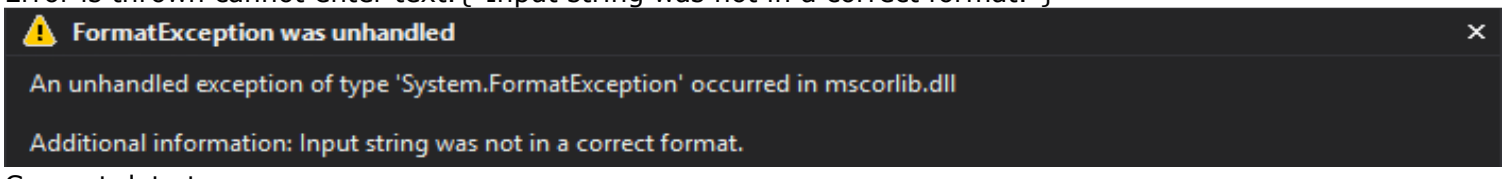
```
int firstTextBox = 0;
int secondTextBox = 0;
int result = 0;

firstTextBox = int.Parse(textBox1.Text);
secondTextBox = int.Parse(textBox2.Text);

result = firstTextBox + secondTextBox;
label3.Text = result.ToString();
```

Basic testing of the calculator.





Error is thrown cannot enter text.{"Input string was not in a correct format."}



Convert data types
For int → int.Parse(myString);
For double → double.Parse(myString);
For bool → bool.Parse(myString);

Expressions Versus Statements

Expressions can be evaluted. This is a very basic expression

```
int x;
x + 3; // this is not a statement, it's an expression
```

Statements, this is a statement

```
x = x + 3; // this is an assignment
```

Valid Statements Consist of

- Assignment → myInterger = 3;
- Call → MessageBox.Show("Hello World");
- Increment → x++;
- Decrement → --x;

"Expressions can be evaluated.. Statements can be excuted."

```
label1.Text = firstTextBox + " " + secondTextBox; // statement
firstTextBox + " " + secondTextBox; // expression not a statement
```

Evalutating Expressions

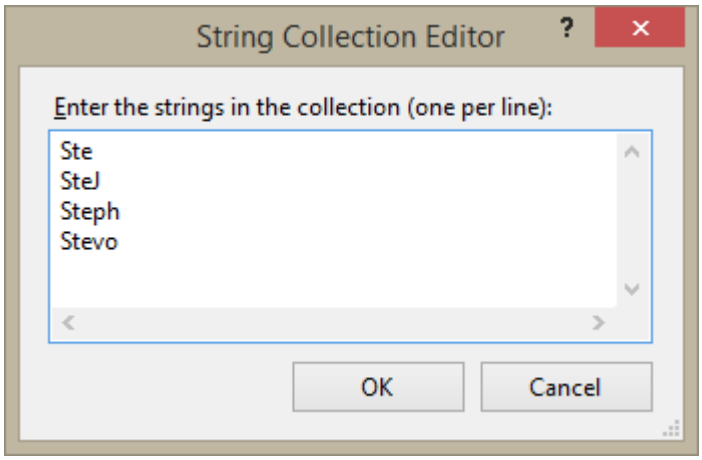
- (3 < 2) .. true or false?
- (3 > 2) .. true or false?
- Given: int x = 3;
  - (x == 3) .. true or false?
  - (x != 3) .. true or false?

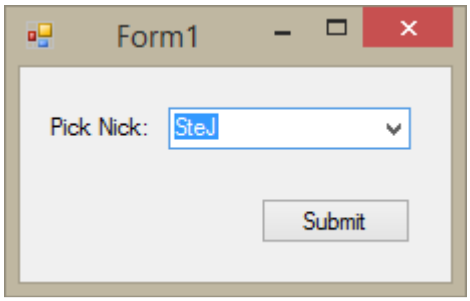
= Assignemnt telling: x is 4
== Evaluation asking: is this equal?
!= evaluates "not equal" 3 is not equal to 4: true

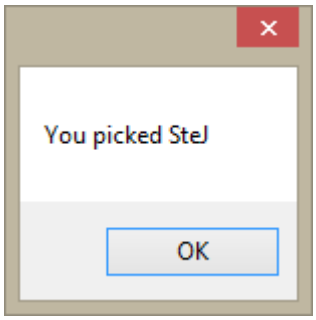
Lesson 4

Iteration and Selection Statements

Selection Statements decide whether or not to execute a block of code based on the evaluation of an expression. If condition else do this. Toolbox select and drag comboBox onto form1, select comboBox in design view click arrow select Edit Items. Type nicknames into dialog box, click OK







If and if else

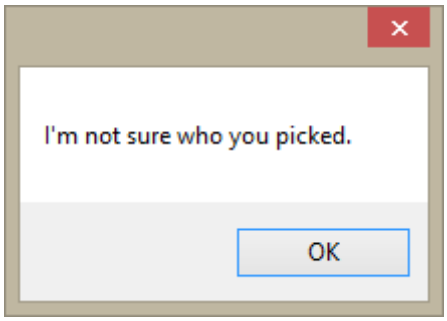
First and second examples of 'if' statements.

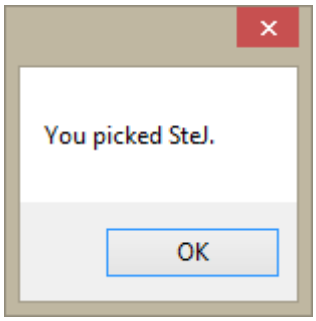
```
// 1. basic 'if' statement
if (comboBox1.Text == "SteJ")
{
    MessageBox.Show("You picked SteJ");
    comboBox1.Text = ""; // clears comboBox if selected
}

// 2. 'if' statement curly braces removed, one line of code to be executed
if (comboBox1.Text == "Steph")
    MessageBox.Show("You picked Steph"); // one line of code after 'if'
```

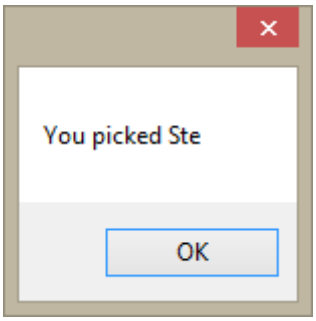
Third example of 'if' statement.

```
// 3. nested 'if' statement
if (comboBox1.Text != "Ste") // if not equal to Ste false go to else
{
    if (comboBox1.Text == "SteJ")
    {
        MessageBox.Show("You picked SteJ."); // SteJ is selected
    }
    else
    {
        MessageBox.Show("I'm not sure who you picked."); // Stevo or Steph selected
    }
} // end if
else
{
    MessageBox.Show("You picked Ste");
} // end else
```

In combox Select Stevo, SteJ and then Ste.

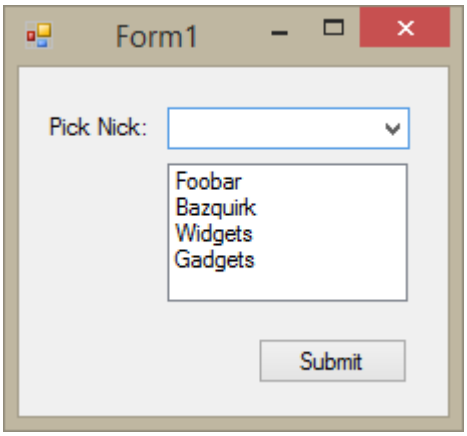


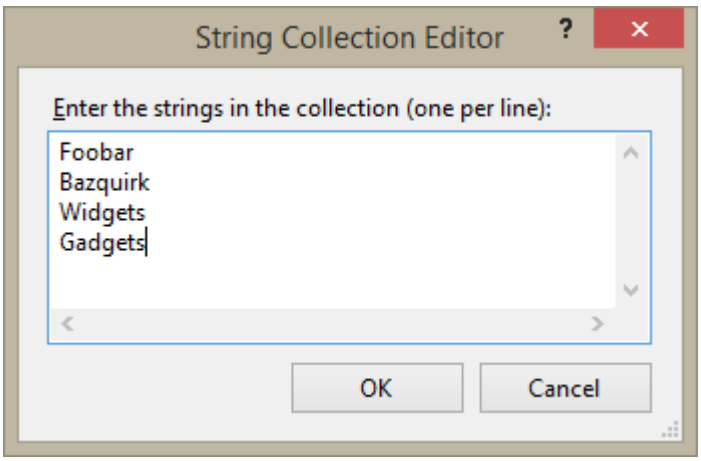




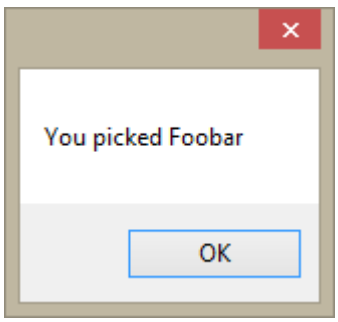
Switch

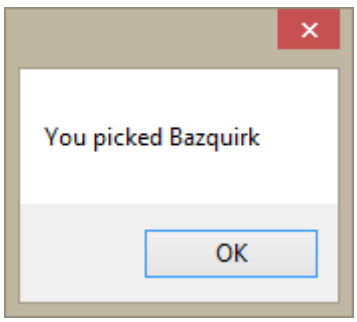
Drag and drop listBox from toolbox onto form1. Select listBox and click arrow. Type Foobar Bazquirk, Widgets & Gadgets into dialog box.

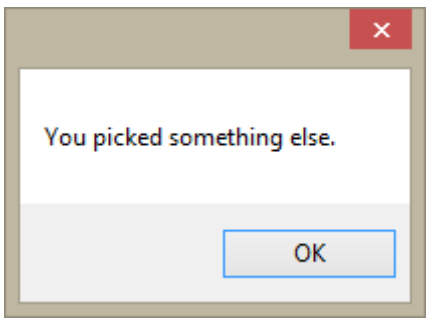





```
// 'switch' statement
switch (listBox1.SelectedItem.ToString())
{
    case "Foobar":
        MessageBox.Show("You picked Foobar");
        break;

    case "Bazquirk":
        MessageBox.Show("You picked Bazquirk");
        break;

    default:
        MessageBox.Show("You picked something else.");
        break;
}
```







Arrays

Type of collection that allows you to group together a bunch of values that are related in some way. All items in the array must be the same date type. Step add break point, debug and step in.

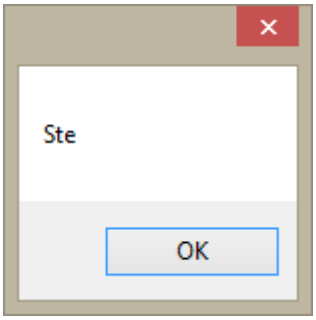
```
// arrays
// 1. sized array, set the size
string[] myArray = new string[2];
myArray[0] = "SteJ";
myArray[1] = "Steph";
// myArray[2] = "Stevo"; // causes an out of bounds exception
MessageBox.Show(myArray[1]);
```

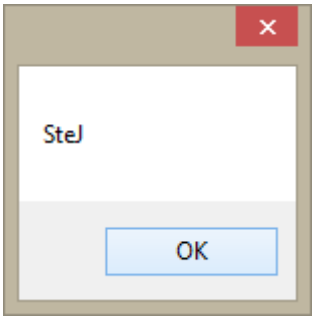
Iteration statements

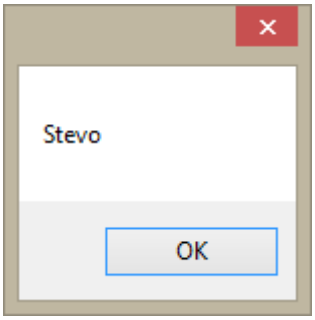
Loop through, or navigate through each item in an array one at a time.

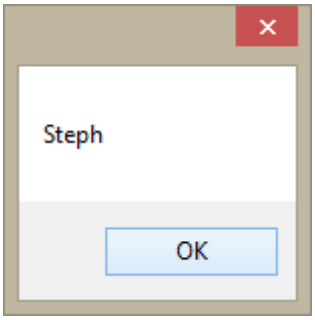
Foreach item (nickname) in array (myArray) MessageBox item (nickname, Ste).

```
// 2. intialized array [0] = Ste, [1] = SteJ, [2] = Stevo, [3] = Steph
string[] myArray = {"Ste", "SteJ", "Stevo", "Steph"};
// MessageBox.Show(myArray[1]); // test array

// create temp var with value of ncikname
foreach (var nickname in myArray)
{
    MessageBox.Show(nickname);
}
```



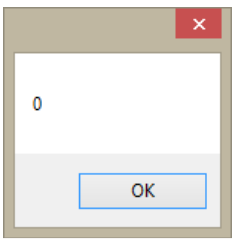


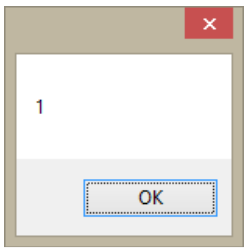


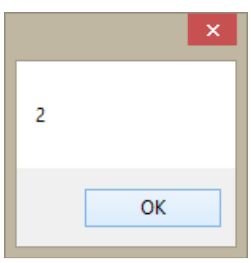


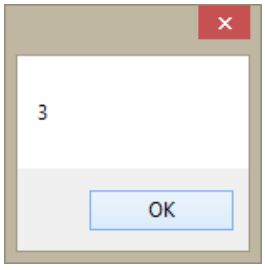
For Loop is index of myArray loop and display message I to a string myArray until length of array is greater than array length.

```
string[] myArray = { "Ste", "SteJ", "Stevo", "Steph" };
for (int i = 0; i < myArray.Length; i++)
{
    MessageBox.Show(i.ToString());
}
```





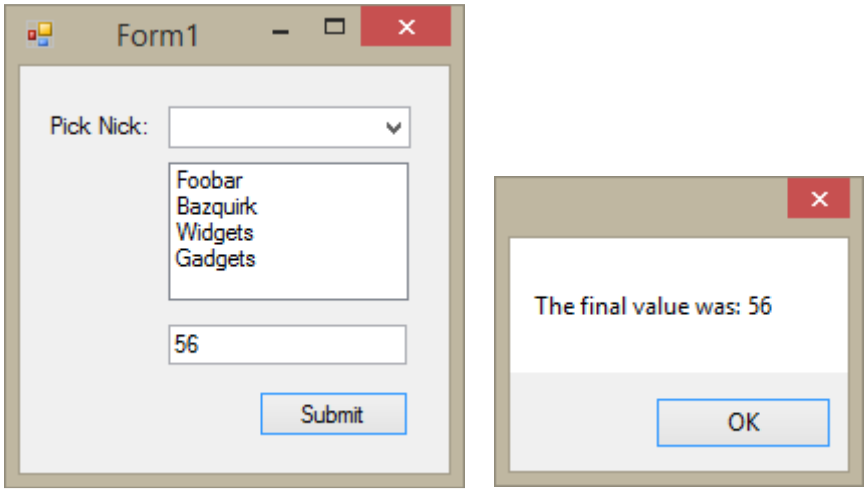




While loop, Drag and drop textBox onto Form1.

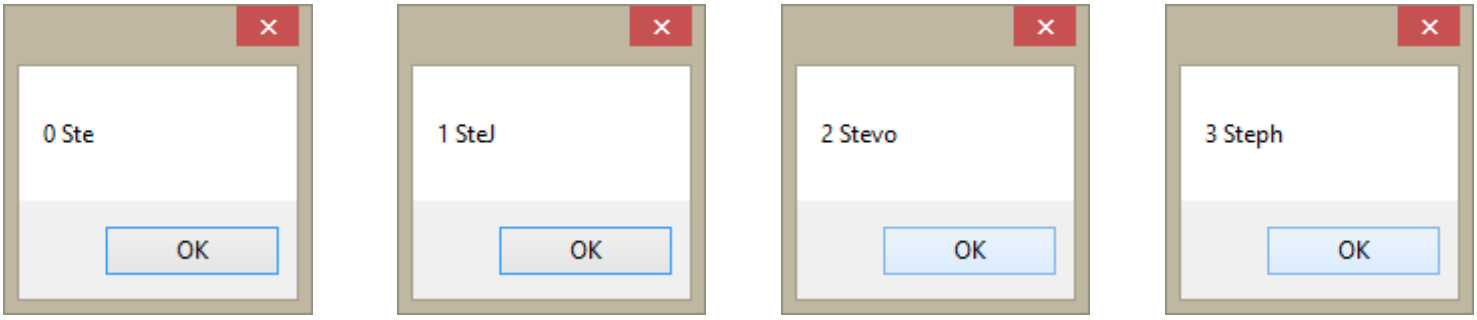
```
int i = 0;
while (i < int.Parse(textBox1.Text))
{
    i++;
}
MessageBox.Show("The final value was: " + i.ToString());
```

Debug and enter number into textBox



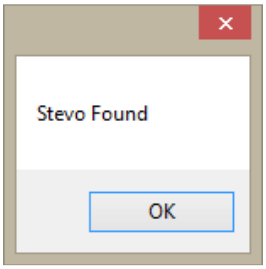
Two nested for loops, to get both index and name a two demensional for loop is created. A nested for loop, the second loop is a foreach.

```
// array [0] = Ste | [1] = SteJ | [2] = Stevo | [3] = Steph
string[] myArray = { "Ste", "SteJ", "Stevo", "Steph" };

// for and foreach loop
for (int i = 0; i < myArray.Length;)
{
    // temp value nickname
    foreach (var nickname in myArray)
    {
        MessageBox.Show(i++ + " " + nickname);
    }
}
```



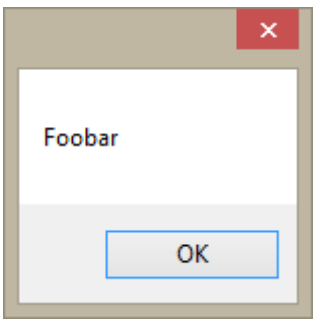
Loop through myArray if index = Stevo. Message "Stevo Found"

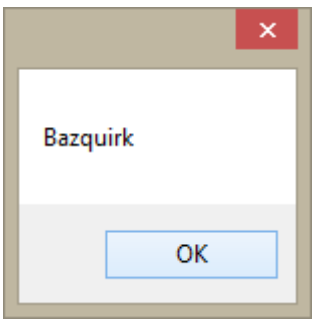
```
string[] myArray = {"Ste", "SteJ", "Stevo", "Steph"};

// combine for with if on array
for (int i = 0; i < myArray.Length; i++)
{
    if (myArray[i] == "Stevo")
    {
        MessageBox.Show("Found Stevo");
    }
}
```

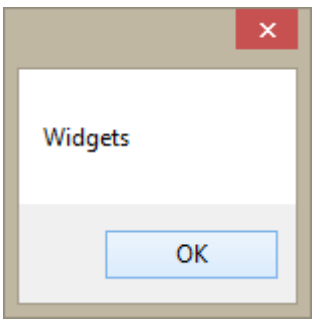


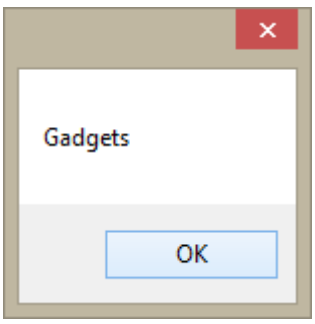
For loop with switch. Select listbox properties SelectionMode and slect from the drop down MultiSimple.

```
for (int i = 0; i < listBox1.SelectedItems.Count; i++)
{
    switch (listBox1.SelectedItems[i].ToString())
    {
        case "Foobar":
            MessageBox.Show("Foobar");
            break;

        case "Bazquirk":
            MessageBox.Show("Bazquirk");
            break;

        case "Widgets":
            MessageBox.Show("Widgets");
            break;

        case "Gadgets":
            MessageBox.Show("Gadgets");
            break;
    }
}
```









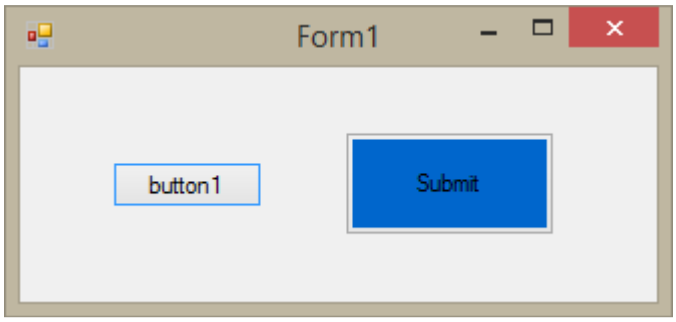
Lesson 5

Object Oriented Programming Fundamentals

Object Oriented Programming seeks to reduce the complexity of creating large applications by breaking the application down into smaller, manageable classes of code. Each class represents an idea, whether tangible and concrete (such as a Product or Employee) or conceptual (such as Inventory or Order). C# is a OOP language months or years just scatching the surface. Simlify application development. Relates code objects to real world objects, creates more flexible applications.

Months years learning OOP
Classes the "music sheet", "blueprint" or "recipe" for an object
Making a distinction between:
A blueprint and a house, create several houses from the same blueprint. The
A recipe and a meal many meal from one recipe – meals can be slightly different every time its cooked.
A sheet of music and a preformance

A template and an object or instance. Each instance of the button has it's own properties. A new instance of the buton class is created.




```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
}
```

Properties

Are attriubutes of our class. Each instance of our class (object) can have different property values, but they all share the same property definitions.
Properties consist of

- Fields variables that hold our property values `private string _make;`
- Property Get and Set statements that allow for accessing and modifying the underlying field.

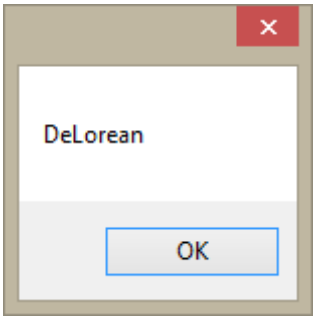
Private filed public property Encapsulation
Value generic parameter generated by C#
Define a vehicle class and create an instance of the vehicle class.

```
Vehicle myCar = new Vehicle();
```

Create Vehicle Class

```
class Vehicle
{
    // defined fields
    private string _make;
    private string _model;
    private int _elapsedMilage;

    // properties
    public string Make
    {
        get { return _make; }
        set { _make = value; }
    }

    public string Model
    {
        get { return _model; }
        set { _model = value; }
    }

    public int ElapsedMilage
    {
        get { return _elapsedMilage; }
        set { _elapsedMilage = value; }
    }
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    //Vehicle Car;
    //Car = new Vehicle();

    Vehicle myCar = new Vehicle(); // new keyword, break point /debug here
    myCar.Make = "DeLorean";
    myCar.Model = "Time Machine";
    myCar.ElapsedMilage = 9999969;

    MessageBox.Show(myCar.Make); // break point /debug here
}
```



Methods

Code inside an event handler Event handlers = Method. Functions procedures. Actions that can be performed by our class

A Dog class
Properties: height, weight, breed
Methods: run, jump, Bark.

Lets create two method one that returns a value and one that is void does not return a method.

```
int result = AddTwoNumbers(3, 5); // returns a value
// MessageBox.Show(result.ToString());
displayMessage(result.ToString()); // does not return a value
displayMessage(AddTwoNumbers(3, 9).ToString());
```

A Method

- Can accept Parameters
- Can return a value or not return void method

```
public string Drive(int kilometers)
        {
            _elapsedMilage += kilometers;

            // could preform some additional calculations here
            // for determining fuel consumption and wear and tear
            // costs

            // friendly string
            string result;
            result = "The " + Make + " " + _model + " now has " + _elapsedMilage +
" kilometers.";
            return result;

}
```

```
string result;
result = myCar.Drive(30);
MessageBox.Show(result);
```

Best practice have a method do one task /function.

Varibale scope
Scope means that the var can only be used with the context of the code block which it was defined, or any sub-code blocks that are also defined within the same code block.

Move the myCar variable into the form scope notice intellisense working now, making it a field of the form1 class. Global code block.

```
private void button1_Click(object sender, EventArgs e)
{
            MessageBox.Show(myCar.ElapsedMilage.ToString());
}
```

```
if(myCar.Make == "DeLorean")
{
    string myModel;
    myModel = myCar.Model;
}

MessageBox.Show(myModel);
```

Basic steps of Microsoft Solutions Framework MFS

- Envision a solution
- Plan the solution
- Develop the solution
- Test the solution
- Deploy the solution

Each “pass” through these steps (to implement additional functionality) is called an “iteration”.

What do we wanna build?
A course booking plan.

Constructors
Methods that excute a new instance of a class (an object) is created.
Constructors are used to initialize the state of a new object.

Overloading
Providing more than one method with the same name, but different ‘method signatures’ as a convenience to calling the method. A method signature is the number and data type of the parameters that one version of an overloaded methos accept. Convince two differnece wyas to od the same thing.

Hierarchy
Classes ‘own’ methods and properties and aredesignated with indentation in the code window.
Classes
Can reference other classes

Static versus Instance Objects

Encapsulation
Public and private – black box programming, take conrol the assigning of the values. Tennet of OOP that suggests that it is better to only make available thoise properties and methods of a class that are absolutely necessary for the consumer of the class to access. Example remote contolr for your T.V. Hide the complxity.

Inheritance
A tennet of an OOP that allows a class to derive its atttributes and its functionality from a bvase class. The derived class can specialize the base class by adding properties and methods, override properties and methods etc.

Private Methods = “Helper Methods
Break down large methods into smaller methods

There is no one write way to write code

Benefits of OOP

1. Provides a process for simplifing the design of an application
2. Keeps changes manageable by keeping the code modular
3. Encapsulation allows your class to change its interface
4. Reusability

How does it work?
Creat an .exe bin directory
C# source code is complied into a .NET Assembly which contains Intermediate Language (IL)

When the Assembly is executed for the first time on a new computer, the IL will be complied into machine language optimized for that computer's configuration. (Just-In_Time Compliation)

After complilation on the end usr=er's machine, the Assembly will be loaded into the .NET Framework Runtime Host

- Permissions, security violoations, etc
- Provides access to the FCL
- Cleans memory
- and more

Important!
Your applications depend on the .NET Runtime to be installed on the end user's computer.
An application can inclued and install the runtime you are using

Namespaces
Allow you to avoid naming conflicts between two clasess that have the same name. Two car classes GreatVehicles.Finance.Car GreatVehicles.Enginerring.Car. Names Stephen O'Connor full quafiy name Stephen J O'Connor
To Use Classess in the FCL

Some important Namespaces in the FCL
System.Windows.Forms
System.Data
System.Net
System.Web
System.IO
System.XML
System.Text
And Loads more

Using the Help system
Dynamic help
Slect keyword and clpres f1 button
Browse through the help

Namespaces and classes in the Framwork Class library
Navigate through Help to learn about the available namespace and classes
.NET Framework Calss Library

What is the .NET framework?

1. Framework Class Library
    a. A series of classes with methods that encapsulating common system or application related functionality.
    b. Contains hundreds of classes available to appllcaitions written to utilize a .NET language lie C#
2. .NET Runtime Host
    a. The 'sandbox' or protective bubble' where your applications run
    b. Maages permissons granted to applications
    c. Provides a common system of data types across all .NET languages (C#, F#, ect.)
3. Utilities (Compliers, Code Generators, etc.)
    a. Various and sundry applications that perform a wide varity of tasks
    C:\Windows\Microsoft.NET\Framework\v4.0.30319

Links

GitHub
http://git-scm.com/book/en/v2/Getting-Started-Git-Basics
https://windows.github.com/
https://help.github.com/articles/set-up-git/
https://github.com/blog/674-introducing-organizations

Microsoft
https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx
MSDN microsoft.com
http://www.learnvisualstudio.net/
https://msdn.microsoft.com/en-us/library/hcw1s69b.aspx
https://msdn.microsoft.com/en-us/library/jj161047.aspx

C# Classes
https://msdn.microsoft.com/en-us/library/x9afc042.aspx

MS SQL Server Database
https://www.youtube.com/watch?v=kWKllVyozOg

SQL Server naming conventions
http://www.isbe.state.il.us/ilds/pdf/sql_server_standards.pdf

https://processing.org/tutorials/2darray/
Two-Dimensional Arrays

Design
SDLC Overview

Inheritance, Polymorphism, and Abstract Classes
http://math.hws.edu/javanotes/c5/s5.html

https://projecteuler.net/