



# Visual C Sharp Fast-track

Sample manual - first two chapters



## TABLE OF CONTENTS (1 of 6)

1	VISUAL STUDIO PRIMER	Page
1.1	Windows Forms	8
1.2	Customising Visual Studio	8
	<i>Setting the Default Start-up Page</i>	9
	<i>Creating Appropriate Settings</i>	9
1.3	Creating Projects	9
1.4	Saving and Closing Files	10
	<i>Closing One Window</i>	11
	<i>Closing All or Nearly All Windows</i>	11
1.5	Auto-hiding windows	11
1.6	The Three Most Useful Windows	12
	<i>Displaying Windows</i>	13
	<i>Properties Window</i>	13
	<i>The Toolbox</i>	13

4	FORM EVENTS	Page
4.1	Events	21
	<i>Attaching Code to a Control's Default Event</i>	21
	<i>Creating a New Event-Handler for an Event</i>	21
	<i>Handling an Event with an Existing Routine</i>	22
4.2	Switching Between Form Design and Code View	22
	<i>Using the Keyboard</i>	23
	<i>Using Solution Explorer</i>	23
4.3	Those Strange Event Arguments	23
	<i>Argument 1 – The Object that Called the Event</i>	24
	<i>Argument 2 – The Event Arguments</i>	25

2	DRAWING FORMS	Page
2.1	Creating a New Form	14
2.2	Changing form properties	15
2.3	Form Controls	15
	<i>Definition of Controls</i>	16
	<i>Adding controls</i>	16
2.4	Selecting Controls	16
	<i>Selecting a Single Control</i>	17
	<i>Selecting Several Controls</i>	17
	<i>Selecting All Controls</i>	17
2.5	Basic Formatting	17
	<i>Resizing Controls</i>	18
	<i>Changing how Controls Look</i>	18
	<i>Moving Controls</i>	18

3	RUNNING APPLICATIONS	Page
3.1	Running a Program	19
	<i>Setting the Default Form in Program.cs</i>	19
	<i>Running and Stopping Programs</i>	19
3.2	Dealing with Errors	19
	<i>Building a Project</i>	20
	<i>Dealing with Build Errors</i>	20

## TABLE OF CONTENTS (2 of 6)

<b>5</b>	<b>VARIABLES AND DATA TYPES</b>	<b>Page</b>
5.1	Why Use Variables?	26
5.2	Declaring Variables	26
	<i>Declaring Variables</i>	27
	<i>Creating Nullable Variables</i>	27
	<i>Using Modified Hungarian Notation</i>	28
	<i>Default Values for Variables</i>	28
	<i>Problems with Declaring Variables within Clauses</i>	28
5.3	Setting Values in Variables	29
	<i>Declaring Integer Variables and Adding/Subtracting</i>	30
	<i>Accumulating Text in String Variables</i>	30
5.4	Variable Data Types	30
	<i>Mapping C# Data Types to the CLR Runtime</i>	31
	<i>A Lazy Person's Data Types</i>	31
	<i>Logical Values</i>	32
	<i>Integers</i>	32
	<i>Decimal (Floating Point) Numbers</i>	32
	<i>Strings and Text</i>	32
	<i>Dates and Times</i>	33
	<i>Objects</i>	33
5.5	Converting Variables	33
	<i>Conversion Using Convert.To</i>	34
	<i>ToString() – Special Case for String Conversions</i>	34
	<i>Casting Data Types</i>	34
5.6	Variable Scope	34
5.7	Notes on Working with Specific Data Types	35
	<i>Working with Characters</i>	36
	<i>Working with Strings</i>	36
	<i>Escape Characters</i>	37
	<i>Verbatim Strings</i>	37
	<i>Splitting Strings</i>	37
	<i>Formatting Dates</i>	38
	<i>Working with Numbers – Possible Operations</i>	39
5.8	Constants	39
5.9	Testing Data Types	40

<b>6</b>	<b>CONDITIONS</b>	<b>Page</b>
6.1	Using IF for Conditions	42
6.2	Operators	42
6.3	The SWITCH statement	43
	<i>Limitations of SWITCH</i>	44
6.4	Ternary and Coalesce Operators	44
	<i>The Ternary Operator</i>	45
	<i>The Null Coalesce Operator</i>	45

<b>7</b>	<b>LOOPS</b>	<b>Page</b>
7.1	Looping in C#	46
	<i>Looping Over a Collection/Array</i>	46
	<i>Looping a Given Number of Times</i>	46
	<i>Looping While a Condition is True (While/Do)</i>	47
7.2	Breaking Out of Loops	48

<b>8</b>	<b>ENUMERATIONS</b>	<b>Page</b>
8.1	Creating and Using Enumerations	50
8.2	Customising Enumerations	50
	<i>Enumeration Aliases</i>	51
	<i>Changing Enumeration Integer Values</i>	51
	<i>Changing the Enumeration Data Type</i>	51
8.3	Looping over Enumerations	51

<b>9</b>	<b>ERROR TRAPPING</b>	<b>Page</b>
9.1	Try / Catch / Finally	53
	<i>Syntax of the Try / Catch Statement</i>	53
	<i>General error trapping example – validating an integer</i>	53
	<i>Catching Specific Errors</i>	54
	<i>The Finally clause</i>	55
	<i>A better alternative to finally – Using</i>	56
	<i>Throwing Exceptions</i>	57
	<i>Exceptions bubble up the call stack</i>	58

## TABLE OF CONTENTS (3 of 6)

10	FILES AND FOLDERS	Page
10.1	StreamReaders and StreamWriters	59
	<i>Our Example</i>	59
	<i>Referencing the System.IO Namespace</i>	59
	<i>Reading in the Customers</i>	59
	<i>Writing out the customers' details to file</i>	60
10.2	Using FILE	61
10.3	FileInfo and DirectoryInfo	62
	<i>Useful File Properties</i>	63
	<i>Getting at folders</i>	63
	<i>Looping over files in folders</i>	64
	<i>Recursively looping over all folders and files</i>	64

11	DESIGNING CLASSES	Page
11.1	Cats as Objects	65
	<i>Types, Classes and Objects</i>	65
	<i>Instantiation and Termination</i>	65
	<i>Properties</i>	66
	<i>Methods</i>	66
	<i>Encapsulation and Exposure</i>	67
	<i>Inheritance</i>	67
11.2	Our Example – Dating Agency Customers	67
	<i>Our Customer Class</i>	68
	<i>Envisaging how you will Consume a Class</i>	68

12	CREATING CLASSES	Page
12.1	Creating a Class	70
12.2	Namespaces	70
	<i>Example of a Namespace</i>	71
	<i>The Using Statement</i>	71
	<i>Removing Unused Using Statements</i>	71
	<i>Giving Aliases to Namespaces</i>	72
	<i>Our DatingAgency Namespace</i>	72
12.3	Creating a Constructor	72
	<i>Syntax of a Constructor</i>	73
	<i>Example of a Constructor</i>	73
12.4	Fields and Properties	74
	<i>Creating Fields</i>	75
	<i>Properties</i>	75
	<i>Refactoring (encapsulating) fields</i>	75
	<i>The Quickest and Best Way to Create Properties</i>	77
	<i>Properties which Perform Other Logic</i>	78
12.5	Methods	78
	<i>Void Methods</i>	79
	<i>Methods which Return Values</i>	79
	<i>Choosing between a Property and a Method</i>	80
12.6	Static Properties and Methods	80
	<i>Example of a Static Property</i>	81
	<i>Example of a Static Method</i>	81

13	STRUCTURES	Page
13.1	Overview of Structures	83
13.2	Differences between Structures and Classes	83
	<i>Structures are Value Types</i>	84
	<i>Other Differences</i>	84
13.3	Familiar Structures!	85

## TABLE OF CONTENTS (4 of 6)

14	VALUE AND REFERENCE	Page
14.1	Types of Memory (Stack and Heap)	87
14.2	Types of Variables	87
	<i>Direct Variables</i>	88
	<i>Indirect Variables</i>	88
14.3	Boxing and Unboxing	88
	<i>System.Object</i>	89
	<i>Boxing</i>	89
	<i>Unboxing</i>	89
14.4	Passing by Value and Reference	90
	<i>Arguments are passed by value by default</i>	91
	<i>Passing arguments by reference using REF or OUT</i>	91

15	OVERLOADING	Page
15.1	Overloading	92
15.2	Creating Overloaded Methods	92
	<i>Consuming the method</i>	93
	<i>Creating our FIND method</i>	93
15.3	Overloading Constructors	94

16	INHERITANCE	Page
16.1	The Concept	96
16.2	Existing Classes in .NET	96
16.3	Inheriting from Existing Classes	97
	<i>The code for this example</i>	98
16.4	Creating your own Hierarchy	99
	<i>Our example – arranging dates between customers</i>	100
	<i>How to inherit classes – syntax</i>	100
	<i>Calling the base constructor</i>	101
	<i>Creating protected members and fields</i>	101
16.5	Overriding Properties	102
	<i>The problem: duplicate member name</i>	103
	<i>Two possible solutions: NEW versus OVERRIDE</i>	103
	<i>Using NEW to solve our problem</i>	103
	<i>Using OVERRIDE and VIRTUAL to solve our problem</i>	104
16.6	Overriding Methods	104
	<i>Calling the BASE method</i>	105
16.7	Sealed Classes and Members	105
	<i>Sealing classes</i>	106
	<i>Sealing methods and properties</i>	106
16.8	Abstract Classes and Members	106

17	EXTENSION METHODS	Page
17.1	Overview of Extension Methods	108
	<i>Syntax of Extension Methods</i>	108
17.2	Examples of Extension Methods	108
	<i>Converting text to proper case</i>	109
	<i>Logging details for a customer</i>	109

18	INTERFACES	Page
18.1	Example .NET Interface	110
18.2	Creating Interfaces	110
	<i>Our simple example – drawing controls on a form</i>	111
	<i>Adding an Interface</i>	111
	<i>Syntax of an Interface</i>	112
	<i>Implementing an interface in a class</i>	112
	<i>The big question – what was the point?</i>	113
18.3	A Realistic Interfaces Example	113
	<i>Our Example</i>	114
	<i>The interface design and solution</i>	114
	<i>Starting to load the form</i>	115
	<i>Dealing with advertisement income</i>	115
	<i>Normal customers</i>	116
	<i>Life memberships</i>	117
	<i>Populating the DataGridView</i>	118

19	DELEGATES AND EVENTS	Page
19.1	Overloading Operators	120
19.2	Our Example	120
19.3	Creating Delegates	121
	<i>Declaring the delegate</i>	122
	<i>Creating the customer acceptance notification method</i>	122
	<i>Consuming the delegate</i>	122
	<i>Storing methods inside the class</i>	123
19.4	Using Adapters	124
	<i>Why you need adapters</i>	125
	<i>Coding adapters</i>	125
19.5	Anonymous Methods	126
19.6	Events	127
	<i>Built-in WinForms Events</i>	128
	<i>Creating your own event</i>	128

## TABLE OF CONTENTS (5 of 6)

20	LAMBDA EXPRESSIONS	Page
20.1	Passing Delegates to Methods	130
	<i>Jargon time!</i>	130
	<i>An example of a predicate delegate</i>	130
20.2	Anonymous Delegates and Lambda Expressions	130
	<i>Anonymous Delegates</i>	131
	<i>Lambda Expressions</i>	131
20.3	Query Syntax and Method Syntax	131
20.4	A Full Worked Example	132

21	ARRAYS	Page
21.1	Arrays	134
	<i>Creating single-dimensional arrays</i>	134
	<i>Populating arrays and retrieving items</i>	134
	<i>Looping over arrays</i>	134
	<i>Multi-dimensional arrays</i>	135

22	LISTS	Page
22.1	Overview of Lists	136
	<i>An Example of a List</i>	136
22.2	Working with Lists	136
	<i>Creating a List</i>	137
	<i>Adding Items to a List</i>	137
	<i>Counting the Items in a List</i>	137
	<i>Displaying All of the Items in a List (FOR EACH)</i>	138
	<i>Removing Items from a List</i>	138
	<i>Finding items in a list</i>	138
	<i>Lambda Expression Syntax for Find Methods</i>	139
22.3	Getting a Subset of a List	140
	<i>Method 1: Using FindAll</i>	140
	<i>Getting a Subset of a List – Method 2: Using GetRange</i>	140
22.4	Joining and Splitting String Lists	140

23	STACKS AND QUEUES	Page
23.1	Queues	142
23.2	Stacks	142

24	DICTIONARIES	Page
24.1	Key/Value Pairs	144
24.2	Our Example – the Customer Class	145
24.3	Working with Dictionaries	146
	<i>Creating a Dictionary</i>	146
	<i>Adding to a dictionary</i>	146
	<i>Removing from a Dictionary</i>	147
	<i>Accessing Dictionary Values</i>	147
	<i>Determining if a key exists</i>	147
24.4	Looping over dictionary items	148
	<i>Looping by Key Value Pair</i>	148
	<i>Looping by Key Only</i>	148
	<i>Looping by Value Only</i>	148
24.5	pe	148

25	DATA TABLES	Page
25.1	Overview of Data Tables	150
	<i>Referencing System.Data</i>	150
	<i>How data tables work</i>	150
25.2	Working with Data Tables	151
	<i>Creating a Data Table</i>	151
	<i>Adding rows</i>	151
	<i>Looping over rows to retrieve data</i>	152

26	GETTING STARTED WITH LINQ	Page
26.1	What is LINQ?	153
	<i>Referencing the LINQ Namespace</i>	153
26.2	Anatomy of a LINQ Query	153
26.3	Implicit and Explicit Variable Types	154
	<i>LINQ queries are compiled</i>	155
	<i>Implicit variable types</i>	155
	<i>The case for explicit variable types</i>	155
26.4	Examples for Different Enumerable Sets	156

## TABLE OF CONTENTS (6 of 6)

27	LINQ SYNTAX	Page
27.1	Our Example	159
27.2	The SELECT keyword	159
	<i>Transformations</i>	160
27.3	Projections using Anonymous Types	160
	<i>Creating anonymous types</i>	161
	<i>Using LINQ to project data onto anonymous types</i>	161
	<i>Using anonymous types to merge data</i>	162
27.4	Taking and Skipping	163
27.5	Forcing Query Execution	164
27.6	Ordering a Sequence (ORDERBY)	165
27.7	Filtering (WHERE)	166
	<i>Calling methods in where clauses</i>	168
27.8	Adding Expressions (LET)	168
	<i>Example: listing primes</i>	169

30	LINQ AND ENTITY FRAMEWORKS	Page
30.1	Getting Data with LINQ	182
	<i>Creating a Data Context</i>	182
	<i>Selecting Data using LINQ</i>	182
	<i>Using Relationships</i>	183
30.2	Changing Data	184
	<i>Inserting Rows</i>	184
	<i>Deleting Rows</i>	185
	<i>Editing Rows</i>	185
30.3	Returning Anonymous Types	186
30.4	Working with Stored Procedures	187
	<i>Changing Stored Procedures</i>	188
30.5	Partial Classes and Entity Frameworks	189
	<i>Creating a partial class</i>	189
	<i>Partial classes don't work with LINQ</i>	190
30.6	Joining Tables	191

28	TYPES OF DATA MODEL	Page
28.1	Our Example	170
	<i>The 3 Types of Data Model</i>	170
28.2	Code First Models	171
	<i>Creating the Table Classes</i>	171
	<i>Creating the Database Class</i>	171
	<i>Creating the Database</i>	172
	<i>Viewing the Database</i>	172
28.3	The Model First Approach	173
28.4	Database First	174
28.5	Thoughts on which Approach to Use	175
	<i>Do you have a Database?</i>	175
	<i>How do you Change your Database?</i>	175
	<i>Do you like Wizards?</i>	176
	<i>Our Recommendation: Database First</i>	176

31	GROUPING IN LINQ AND EF	Page
31.1	Basic Grouping	192
	<i>How grouping works</i>	192
31.2	Grouping into Ranges	193
	<i>Grouping films by their initial letter</i>	194
	<i>Grouping customers by their decade of birth</i>	194
31.3	Grouping Into and Ordering	195

29	DATABASE FIRST MODELS	Page
29.1	Creating a Model	177
	<i>Step 1 - Creating the Database</i>	177
	<i>Step 2 - Adding a New Item</i>	177
	<i>Step 3 - Choosing the Model Type and Connection</i>	178
	<i>Step 4 - Choosing the EF Version</i>	179
	<i>Step 5 - Choosing the Entities for your Model</i>	179
	<i>Step 6 - Saving your Model</i>	180
29.2	Updating Models	181

32	USEFUL SHORT-CUT KEYS	Page
32.1	The Best Short-Cut Keys in Visual Studio	197
	<i>Going to the definition of a variable or member</i>	197
	<i>Going forward and backward using the keyboard</i>	197
	<i>Auto-formatting text</i>	198
	<i>Adding a Using statement</i>	199

## CHAPTER 1 - VISUAL STUDIO PRIMER

### 1.1 Windows Forms

There are three main types of application you can develop using *Visual Studio* (Microsoft's development tool for .NET programmers):

Type of application	Use for
<i>WinForms (Windows Forms)</i>	Creating basic business standalone applications to run within Windows.
<i>WPF (Windows Presentation Foundation)</i>	Creating Windows applications with fancy graphics (WPF takes longer to learn but is more powerful than Windows forms).
<i>ASP.NET webforms</i>	Creating websites using forms-based ASP.NET.
<i>ASP.NET MVC / MVC Core</i>	Creating websites using the model-view-controller method.

This courseware uses WinForms exclusively, because it's the simplest of the 3 types of application (the aim of the course is to teach C#, not drawing!).

This is an application form for the *Wise Owl Dating Agency (WODA, not to be confused with YODA)*. Clicking on the button could show a message like this:

To create a Windows Forms application like this, you'll first need to learn to use Visual Studio, as shown in the rest of this chapter.



*This courseware uses Visual Studio 2017, but you should find that all functionality is pretty much identical, regardless of which version of Visual Studio you use.*

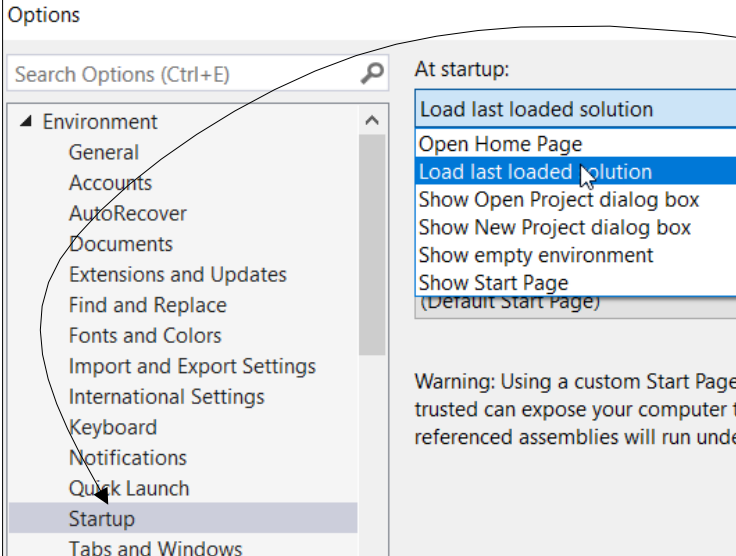


## 1.2 Customising Visual Studio

Before getting started using Visual Studio, it's a good idea to make a couple of changes.

### Setting the Default Start-up Page

To change what you see each time that you go into Visual Studio, from the menu select: **Tools Options...** and then:



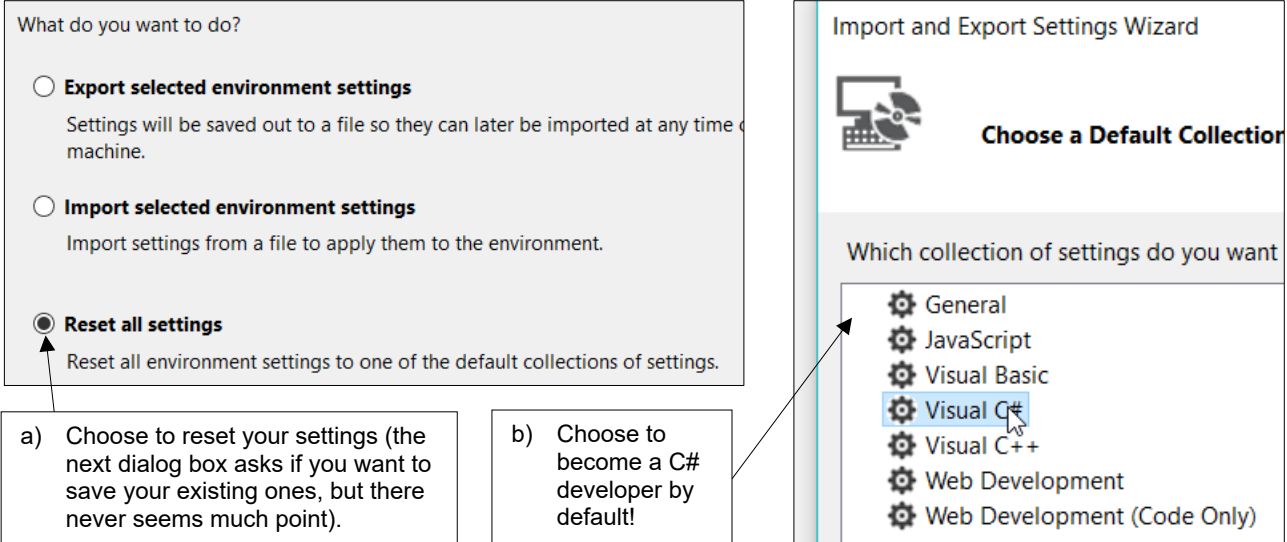
The screenshot shows the 'Options' dialog box with the 'Environment' category expanded to 'Startup'. The 'At startup:' section lists several options, with 'Load last loaded solution' highlighted. A warning message is visible at the bottom: 'Warning: Using a custom Start Page trusted can expose your computer to referenced assemblies will run under...'

a) Select the **Startup** option under the **Environment** section.

b) Choose the option you want (the Wise Owl preference is to show the last thing you were working with, as here).

### Creating Appropriate Settings

In Visual Studio you can save *settings*, letting you switch between different ways of working. From the menu choose **Tools** → **Import and Export Settings...** and then:



The screenshot shows the 'Import and Export Settings Wizard' dialog box. In the first step, 'Reset all settings' is selected. In the second step, 'Visual C#' is selected as the default collection.

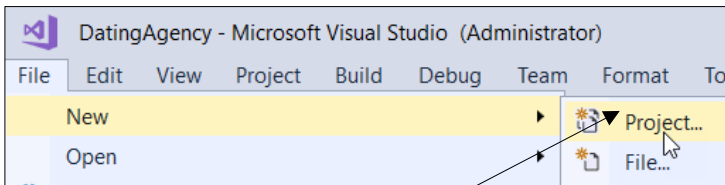
a) Choose to reset your settings (the next dialog box asks if you want to save your existing ones, but there never seems much point).

b) Choose to become a C# developer by default!

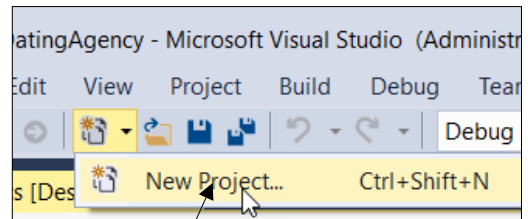
## 1.3 Creating Projects

A *project* is the term for the container for all of the files which make up your application. Projects can contain any number of folders and files. To create a project:

- 1) Choose one of the ways shown below to create a new project:



Either choose this menu option (you can also press **Shift** + **Ctrl** + **N** to do this) ...



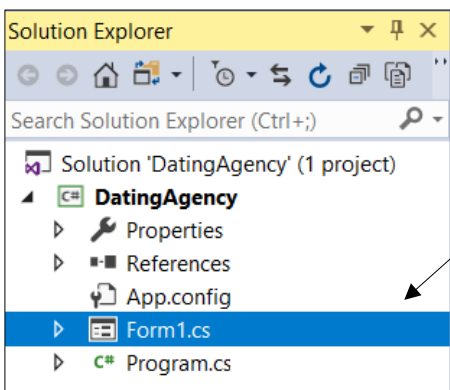
... or click on this tool to create a new project.

- 2) Complete the dialog box which appears as shown below, then select **OK**.

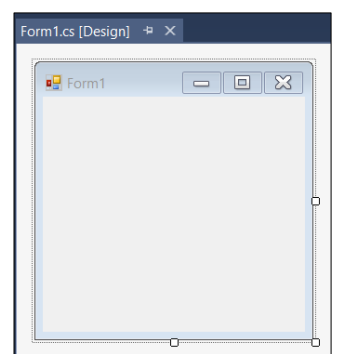
a) Choose to create a Windows application using C#.

b) Choose to create a WinForms application, not a WPF one.

c) Give your project a name, and choose where to store it.



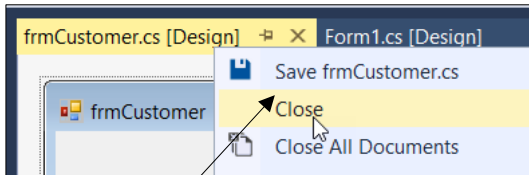
- d) Visual Studio automatically creates a new form called **Form1** in your new project, and shows this on screen:



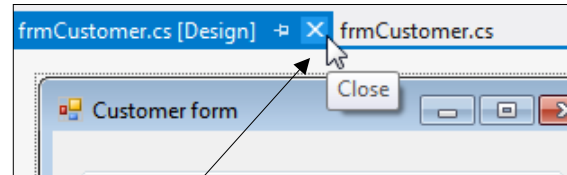
## 1.4 Saving and Closing Files

### Closing One Window

To close a single window, either right-click on its tab or use the cross:



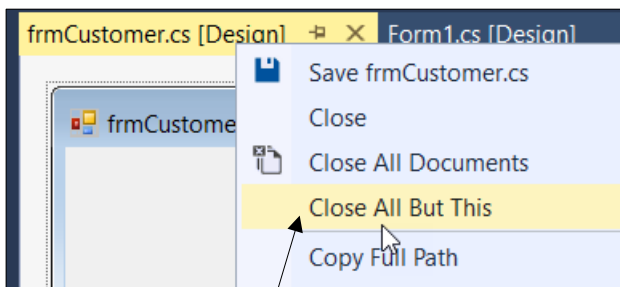
Either right-click on the window tab and choose the option shown to close it ...



... or click on this cross to close the window in question.

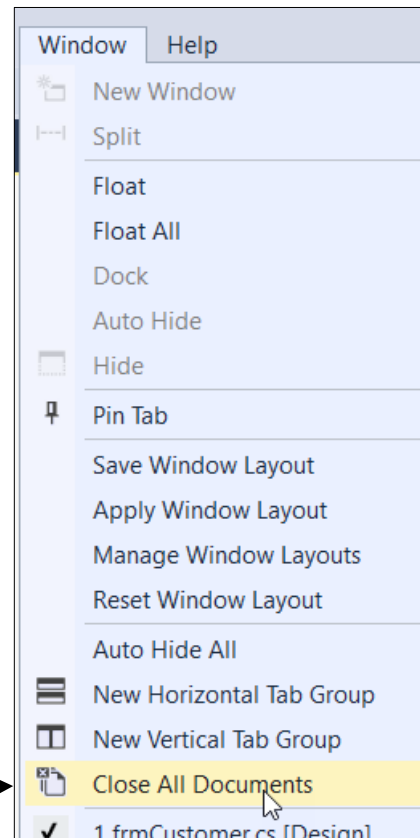
### Closing All or Nearly All Windows

Windows accumulate quickly in Visual Studio – before you know where you are you have 8 or 10 open! Two quick ways to close all or nearly all of your windows are shown below:



Right-click on a window and choose this menu option to close all the windows except this one.

Select this menu option to close every window that you have open.

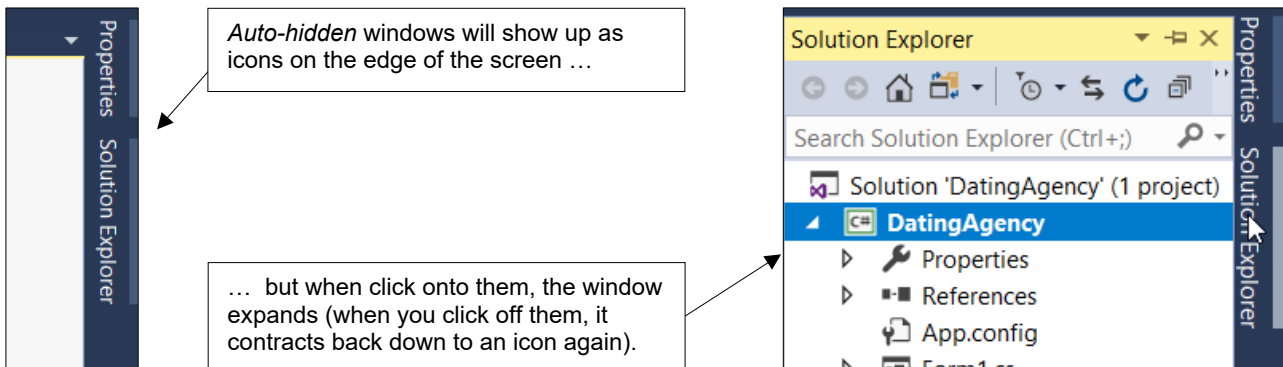


**Wise Owl's Hint**

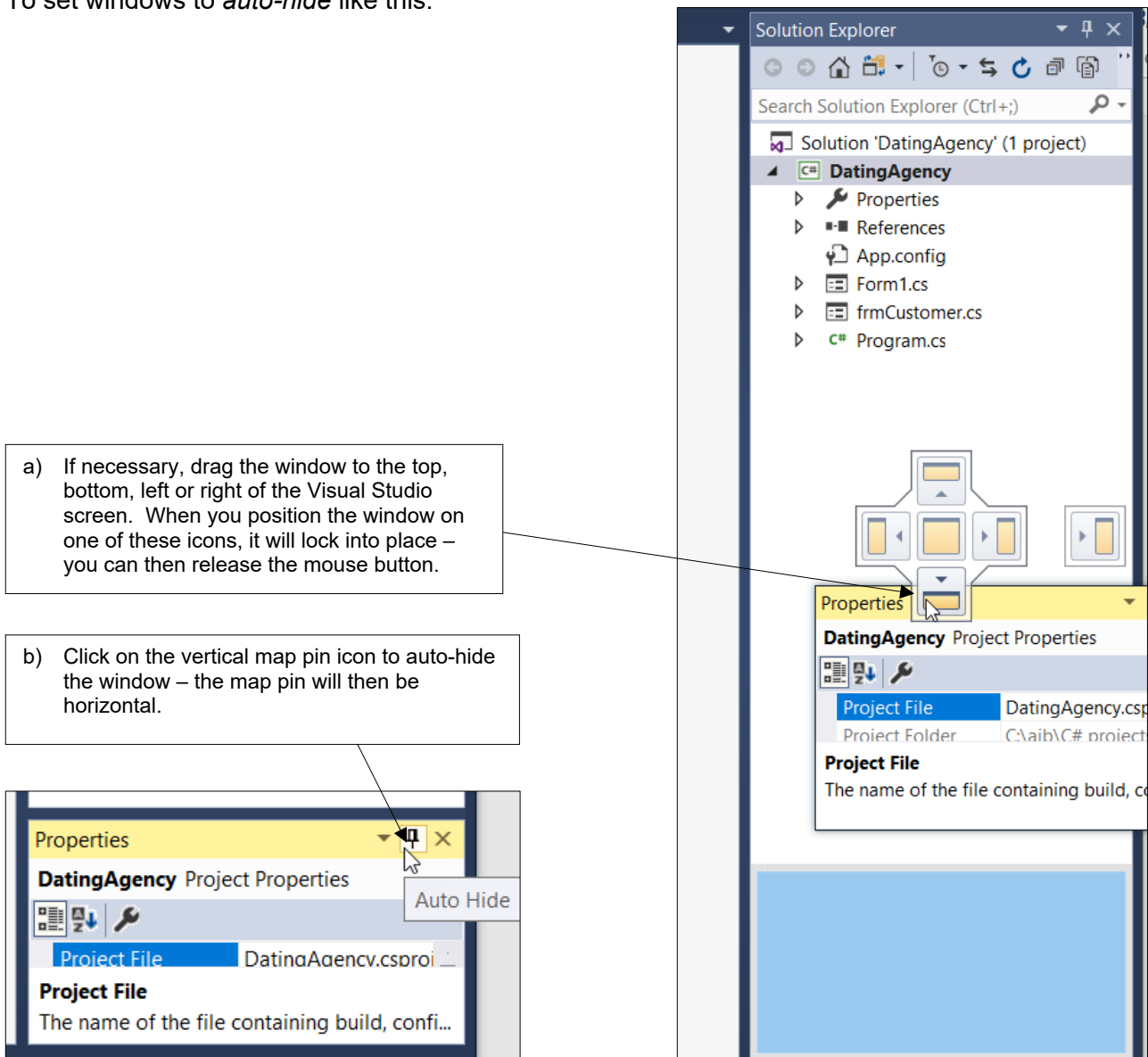
A useful short-cut is to press **Shift + Ctrl + S** to save every window that you have open – this means that you can then close all of your open windows without having to confirm that you want to save the changes for each.

## 1.5 Auto-hiding windows

There are a lot of windows in Visual Studio, and they can quickly clutter up the screen. The best thing to do is to *auto-hide* them, so that they show up as icons on the edge of the screen:



To set windows to *auto-hide* like this:



## 1.6 The Three Most Useful Windows

### Displaying Windows

Visual Studio contains 3 toolbars that you'll need to have open most of the time – here's how to display them:

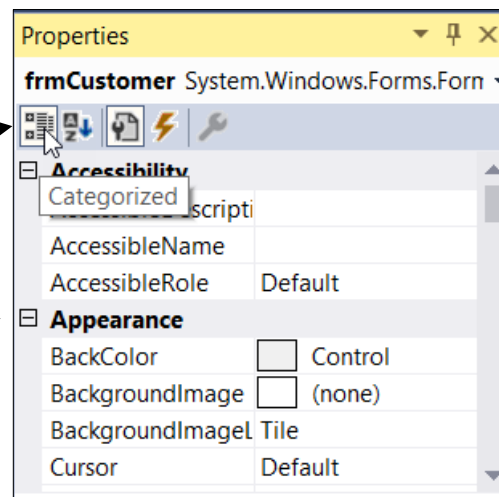
Windows	Menu option	Short-cut key
<i>Solution Explorer</i>	<b>V</b> iew Solution Explorer	Alt + Ctrl + L
<i>Properties</i>	<b>V</b> iew Properties Window	F4
<i>Toolbox</i>	<b>V</b> iew Toolbox	Alt + Ctrl + X

### Properties Window

You can choose to display properties by category or alphabetically:

If you choose to display properties by category (the best option, this owl thinks) ...

... you'll see the properties grouped into categories (**A**ccessibility, **A**pppearance, etc.).

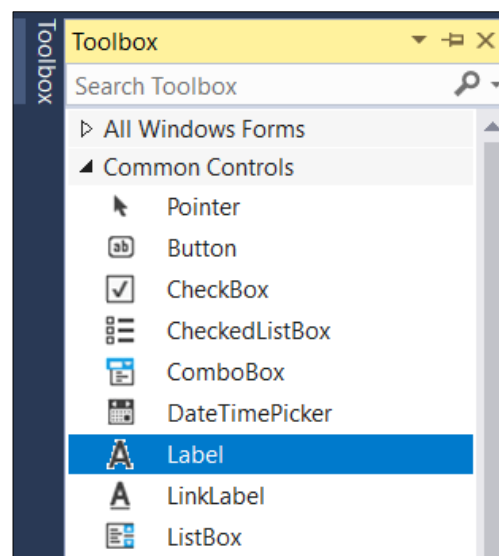


### The Toolbox

The *toolbox* allows you to add things to a form quickly:

Click on any tool and drag it onto the form to add it. The main tools we'll use are:

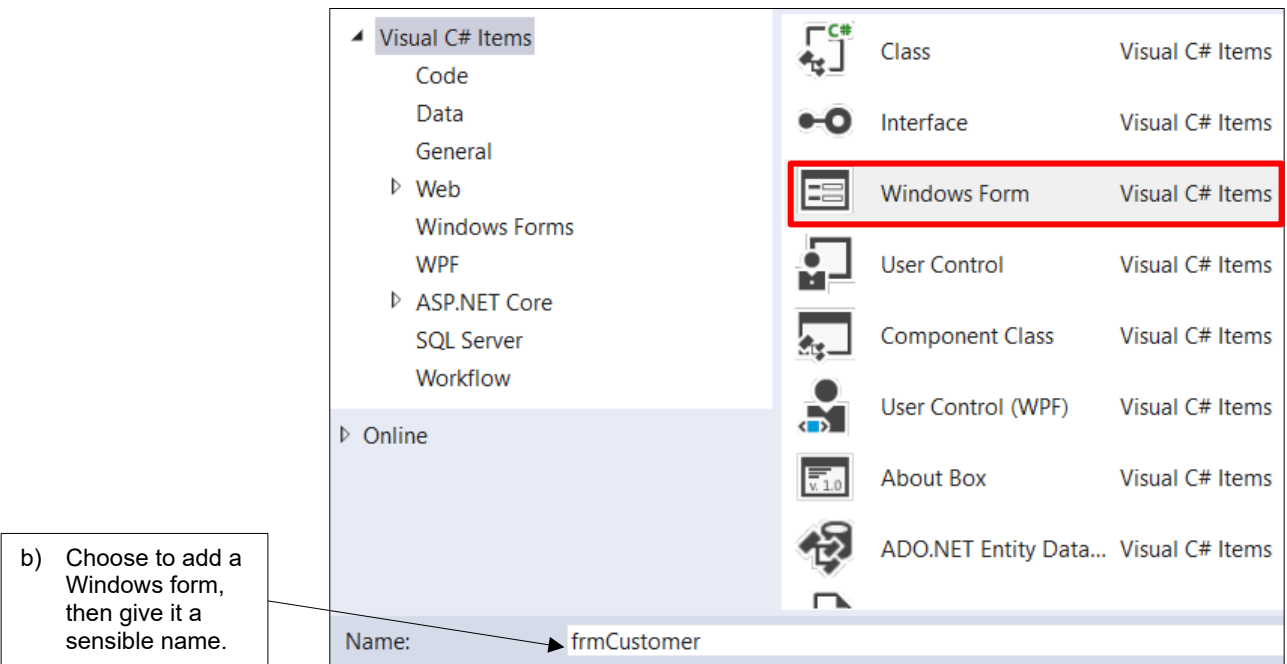
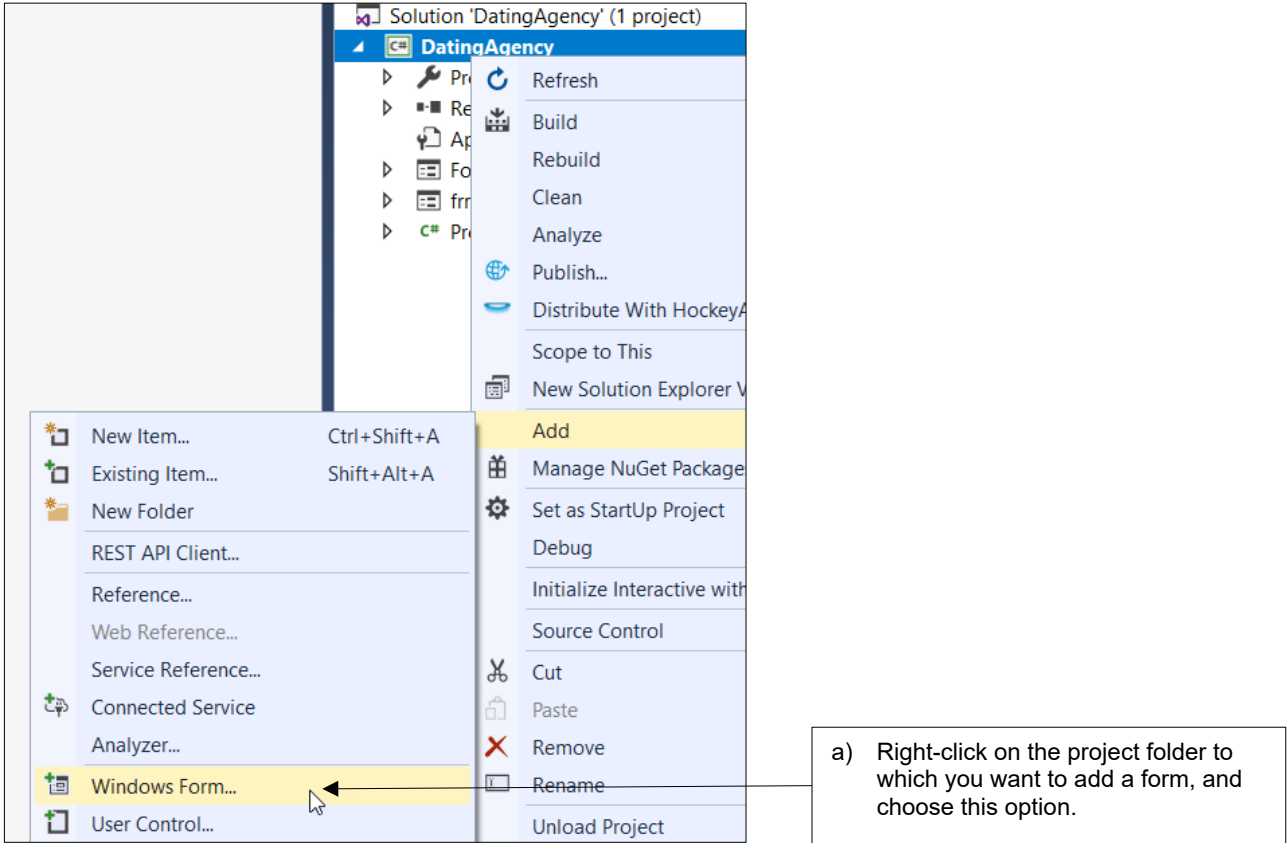
Tool	What it does
<i>Label</i>	Displays text on a form.
<i>TextBox</i>	Allows a user to type text into a box.
<i>Button</i>	A clickable button, with events attached.
<i>ComboBox</i>	A dropdown list.
<i>GroupBox</i>	Used to draw rectangles on forms.



# CHAPTER 2 - DRAWING FORMS

## 2.1 Creating a New Form

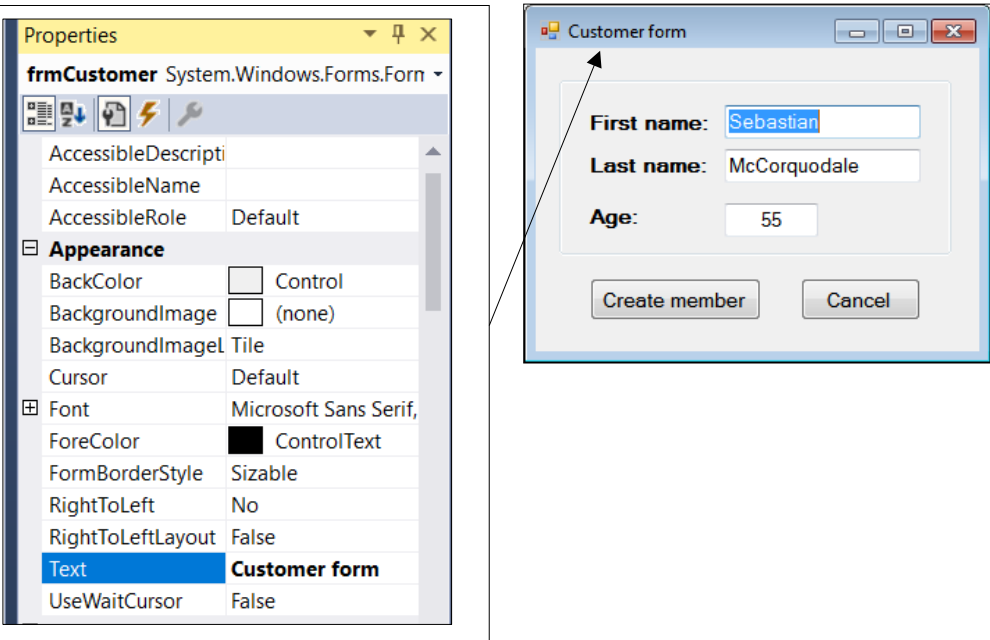
To create a new form, press **Alt** + **Ctrl** + **L** to bring up the Solution Explorer window, then:



## 2.2 Changing form properties

Click on the background of the form to make sure it's selected, then change any properties:

For example, the caption of the form comes from its **Text** property, which is in the **Appearance** category.



The image shows two side-by-side screenshots. On the left is the Visual Studio Properties window for a form named 'frmCustomer'. The 'Text' property is highlighted in blue and set to 'Customer form'. On the right is a screenshot of the 'Customer form' window. It has a title bar that says 'Customer form'. Inside, there are three input fields: 'First name:' with the text 'Sebastian', 'Last name:' with the text 'McCorquodale', and 'Age:' with the text '55'. At the bottom are two buttons: 'Create member' and 'Cancel'. An arrow points from the 'Text' property in the Properties window to the title bar of the form window.

Here are some useful properties to set for a form:

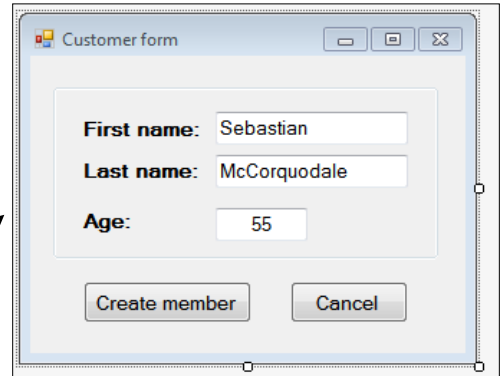
Category	Property	Notes
<i>Appearance</i>	<b>Text</b>	The caption for the form (as above).
<i>Layout</i>	<b>StartPosition</b>	Change to <b>CenterScreen</b> to make a form appear in the middle of the screen.
<i>Msc</i>	<b>AcceptButton</b>	Set to the button which you want to be selected by default if a user presses <b>↵</b> .
	<b>CancelButton</b>	Set to the button which you want to be selected by default if a user presses <b>Esc</b> .

## 2.3 Form Controls

### Definition of Controls

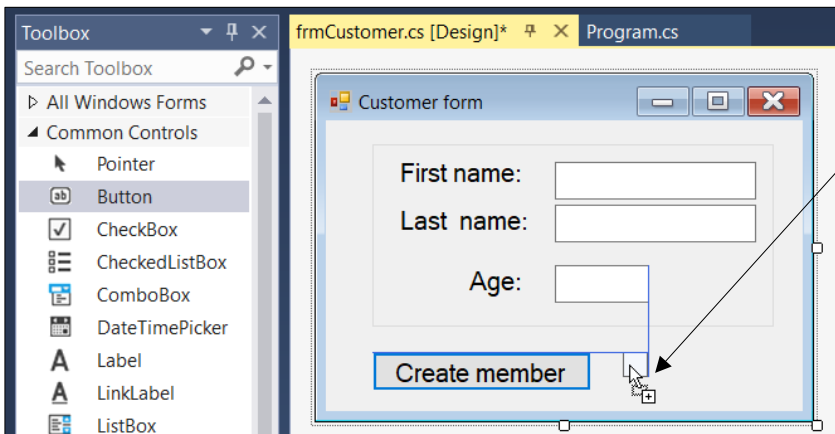
All the different widgets (labels, text boxes, command buttons, etc) that you add to a form are called *controls*:

This form contains 3 labels, 3 textboxes, 2 buttons and a groupbox (the thin rectangle).



### Adding controls

The easiest way to add a control to a form is to click and drag on it:



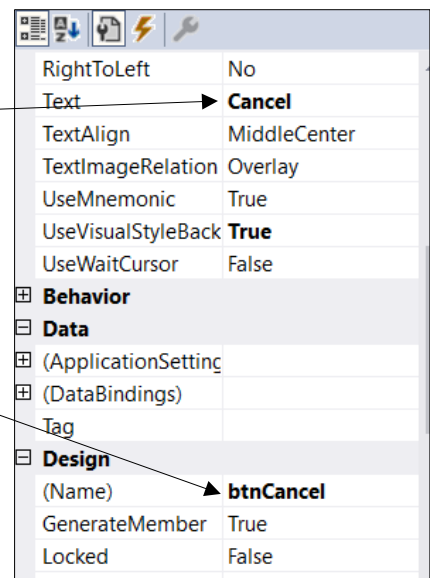
Here we're adding a clickable **Button** to a form. You can use the horizontal and vertical blue lines which appear to align controls.

You should then give your control a good name:

It's likely that you'll want to change the text displayed on a button.

The first **Design** category property of a control is always its name. Wise Owl use modified Hungarian notation (!) to name controls:

Type of control	Prefix to use
<i>TextBox</i>	<b>txt</b>
<i>Label</i>	<b>lbl</b>
<i>Button</i>	<b>btn</b>
<i>ComboBox</i>	<b>cmb</b>





## 2.4 Selecting Controls

Before you can move, copy, edit, delete, rename or format controls, you must first select them!

### Selecting a Single Control

Here are two ways to select a single control:

Click on a control to select it ...

.. or click on the dropdown at the top of the **Properties** window to change what you have selected.

### Selecting Several Controls

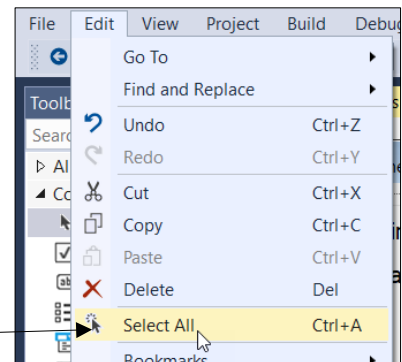
Suppose that you want to select the two buttons in the form above. Here are two ways to do this:

Method	Example diagram
Select one control, then hold down the <b>Ctrl</b> key and click on the others you want to add to/remove from your selection.	
Click and drag to draw a rectangle – anything it touches or encloses will be selected.	

### Selecting All Controls

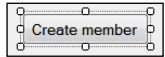
To do this, press **Ctrl** + **A** or select the menu option shown:

Select this option from the menu to select all of the controls on a form.



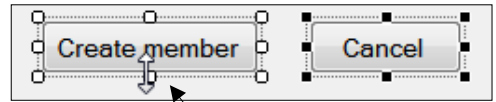
## 2.5 Basic Formatting

You can change how a form looks by *formatting* it. All of the examples below refer to the button, but the properties shown apply to most controls and to the form itself.



### Resizing Controls

You can change the size of one or more controls using the handles surrounding them:





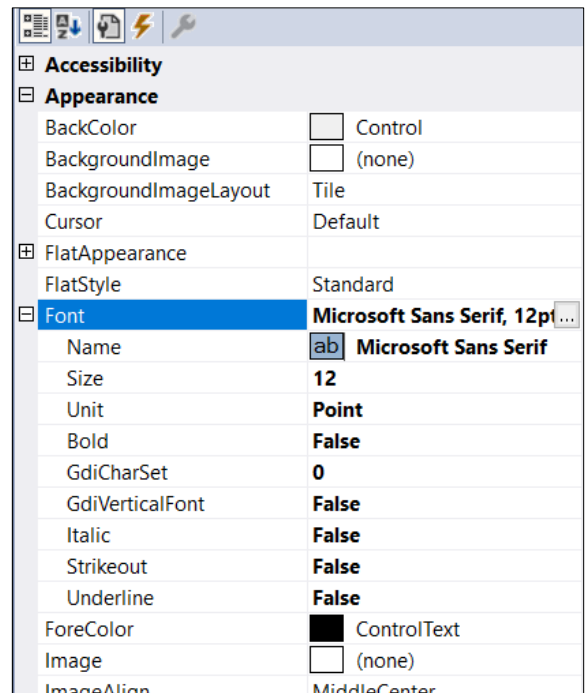
Clicking and dragging here, for example, would change the height of both buttons, since they are both selected.

### Changing how Controls Look

To do this, select the control(s) and change any of their properties – here are a couple of examples:

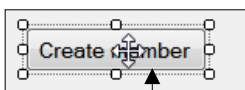
You can change the appearance of controls using these properties – here are some ideas!

Example	Notes
	This font is set to <b>MV Boli</b> – and looks the worse for it!
	Changing the <b>BackColor</b> , <b>ForeColor</b> and <b>FlatStyle</b> properties.



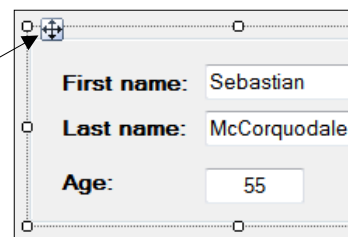
### Moving Controls

To move controls, select them and then click and drag on them with the  symbol:










































Click and drag to move this textbox.

The only way to move this **GroupBox** is using this icon.



## What we do!

		Basic training	Advanced training	Systems / consultancy
<b>Office</b>	Microsoft Excel			
	VBA macros			
	Office Scripts			
	Microsoft Access			
<b>Business Intelligence</b>	Power BI			
	Power Apps			
	Power Automate / PAD			
<b>SQL Server</b>	SQL			
	Reporting Services			
	Report Builder			
	Integration Services			
	Analysis Services			
<b>Coding</b>	Visual C# programming			
	VB programming			
	DAX			
	Python			



**WiseOwl**  
Training

