158.212 Application Software Development

Suriadi Suriadi

Lecture 4

Object-Oriented Paradigm

Visual Studio IDE

- Form-based Applications
- Solution Explorer
- Properties Window
- Toolbox
- Basic Controls: TextBox, Label, Button

Object Oriented Programming

- Object Oriented Programming (OOP) is a programming paradigm

- Key feature: the grouping of ***data and procedures*** together into entities, commonly known as ***classes.***

- A ***class*** contain variables (data members) and functions/subroutines (methods).

This enables developers to map the programming logic closer to the real-world problem that is being solved.

What is a Class?

- For example: a car
  - Data members (properties)
    - Windscreen, seatbelt, trunk, etc.
  - Methods:
    - Accelerate, brake, park, etc.

- The essence is to *encapsulate* related data and methods together into a single entity (known as a class)

Class and Object

- A class forms a **template** or a **blueprint** of a certain entity.

- In OOP, your codes are structured as classes (i.e. you develop the blueprint for the entities that you want to use in your application).

- Loosely speaking, to use a class in your program, you need to:
    - instantiate the class to generate an ***object***

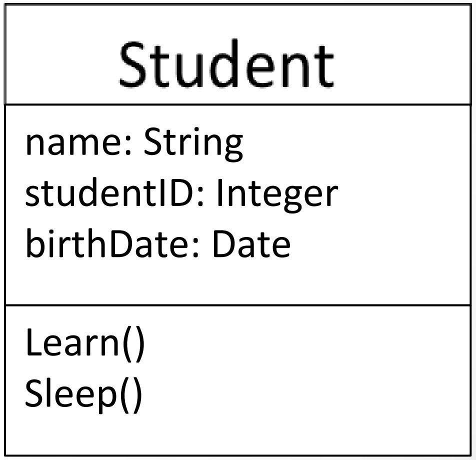
- A software application is thus designed as interactions between objects

Class and Objects

- For example, to model a student, we may define that a student must have the following properties: a name, a student ID, and a date of birth.

- Furthermore, we also need to encode the behaviours, e.g. Learn() and Sleep().

- Often, a class is expressed using a *class diagram*



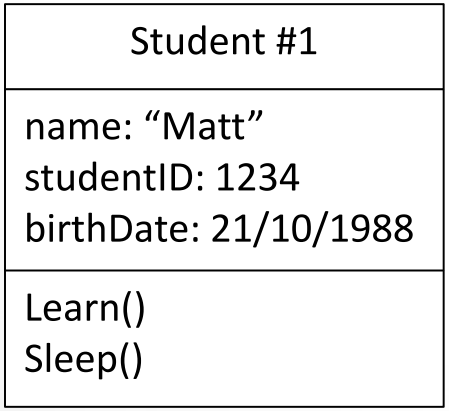
Student

name: String
studentID: Integer
birthDate: Date

Learn()
Sleep()

Class and Objects

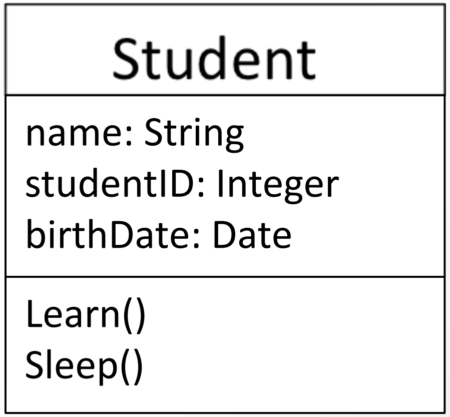
- A Class can be **instantiated** to create an **object**.
- The created object is the **concrete** representation of the class.
- The created object is also known as an **instance** of that class, e.g. Student#1 is an instance of the Student class..



Student #1

name: “Matt”
studentID: 1234
birthDate: 21/10/1988

Learn()
Sleep()



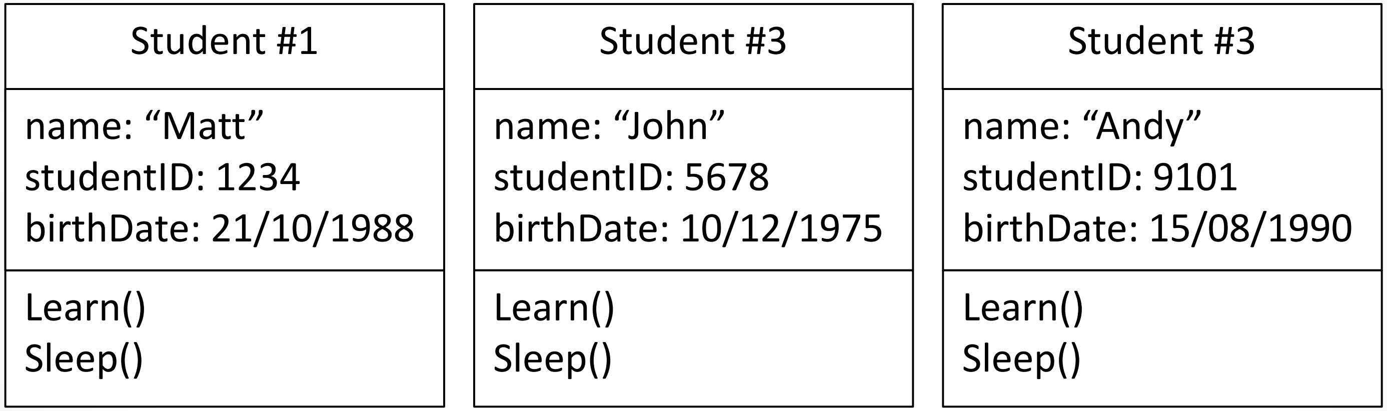
Student

name: String
studentID: Integer
birthDate: Date

Learn()
Sleep()

Class and Objects

We can then create a number of instances of this Student class. For example:



Student #1	Student #3	Student #3
name: “Matt” studentID: 1234 birthDate: 21/10/1988	name: “John” studentID: 5678 birthDate: 10/12/1975	name: “Andy” studentID: 9101 birthDate: 15/08/1990

Learn()
Sleep()
	Learn()
	Sleep()
		Learn()
		Sleep()

These objects are all of type Student but have different values.

Class and Objects

A Class defines:

- Members (properties), i.e. the data that describes an object.
  - The properties can be simple data types such as Integers, Strings, Doubles, as well as ***other objects.***

- Methods (behaviour) – describes what an object can do.
  - These are the functions/subroutines that the object can perform.

Class and Objects

An Object, as instantiated from a class, is defined by three things

type

- State → the present condition of the object, i.e. the values in the objects properties.
- Behaviour → the functions or operations an object can do.

identifier

- Identity → the identity of the object within the program, i.e. the identifier used to access an object.

Visibility and Access

- Generally, visibility of objects is comparable to variables (Lecture 3) – similar concepts (code blocks and scope)

- …… but, in OOP, there is an added `access control’ mechanism
    - known as the ***access modifiers***

- Access modifiers govern who can access the properties and methods of an object

- Three types of modifiers: *public, private*, and *protected.*

Visibility and Access

- **Public** properties and methods can be called by any part of the program in which the object is visible.
- **Private** members and methods are only accessible by methods of the class.
- **Protected** members and methods - accessible by the methods of the class or subclasses.

Why do we need access modifiers?

Encapsulation

- Real-world behaviours can be complex, e.g.
  - The mechanics of a car
  - The inner working of a CPU

- As users, we often *do not care* about the complexity.
  - *“I just want to be able to use it!”*

- ***Encapsulation*** mimics such concerns in OOP

Encapsulation

- Behaviours that are of *no concerns to users* (e.g. the drum rotating mechanism of a washing machine) should be *hidden*
  - Use *private* or *protected* modifier

- Behaviours that users *need to know or use* (e.g. the model of a washing machine and the ability to instruct the machine to wash clothes) need to be *exposed*
  - Use *public* or *protected* modifier

- The public properties and methods represent the **class interface**
  - i.e., those properties and methods that are exposed to (and hence, usable by) the public

Encapsulation

If a property or a method is accessible, it can be accessed using the "." operator.

```
Console.WriteLine(“Hello world”)
i = Console.ReadLine()
Integer.Parse(i)
```

Shared/Static Modifier

- Members and methods of a class can also be declared as **shared** (VB) or **static** (C#).

- In this case, that member or method will belong to the class and not to an individual instance.

- Every instance of the class will be able to access this member or method.

Shared/Static Modifier

- Static/shared modifiers also allow members to be accessed or methods to be used without creating an instance of a class.

- Revisiting code samples from previous lectures:
  - e.g. in the C# console applications, the functions/subroutines had to be declared as static in order to be visible from the static Main() function.

More OOP Concepts

Object Oriented concepts:

user defined classes

constructors

inheritance

method overriding

polymorphism

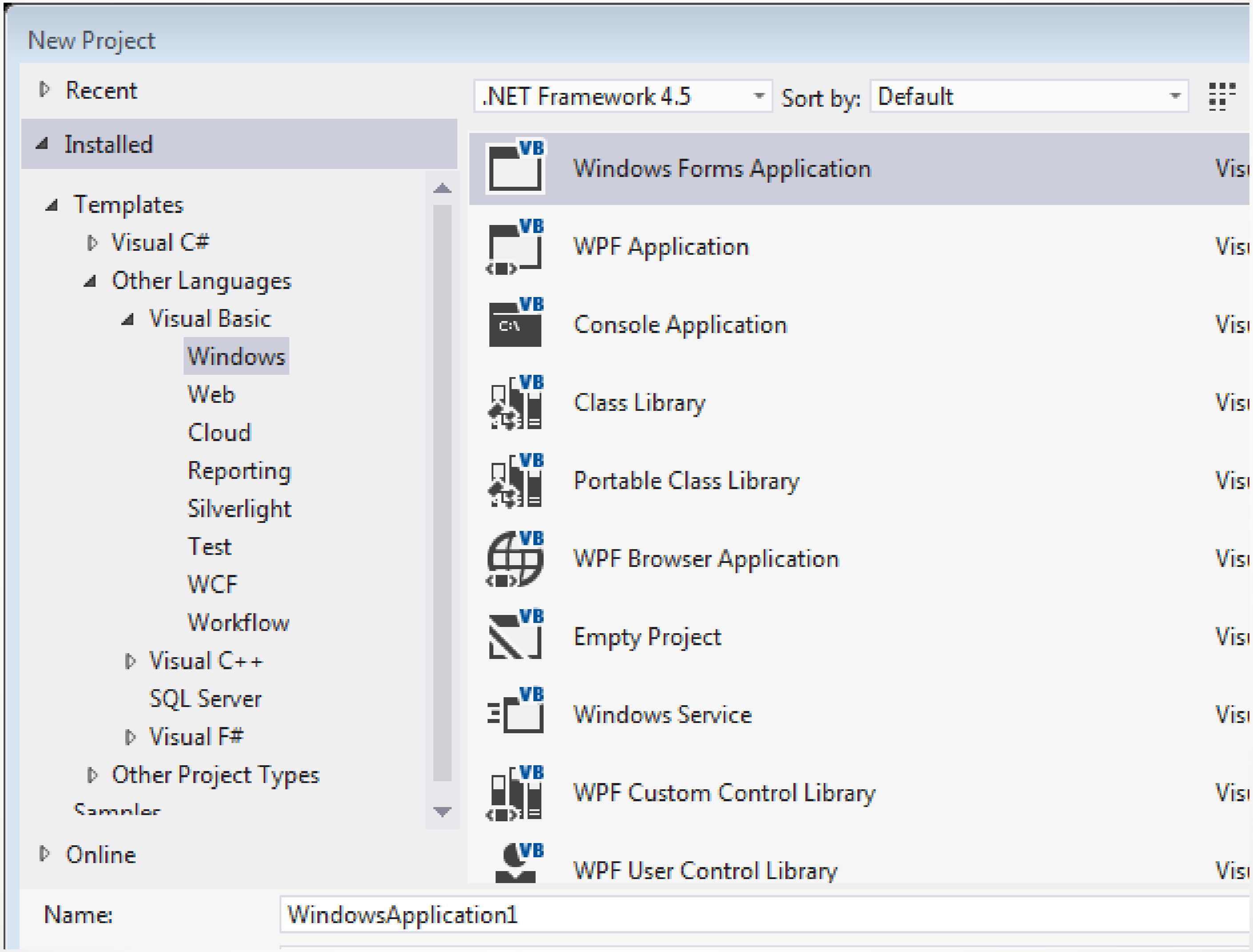
abstract classes

multiple inheritance

interfaces

The above will be covered in more detail later in the course.

Form-based Project



New Project

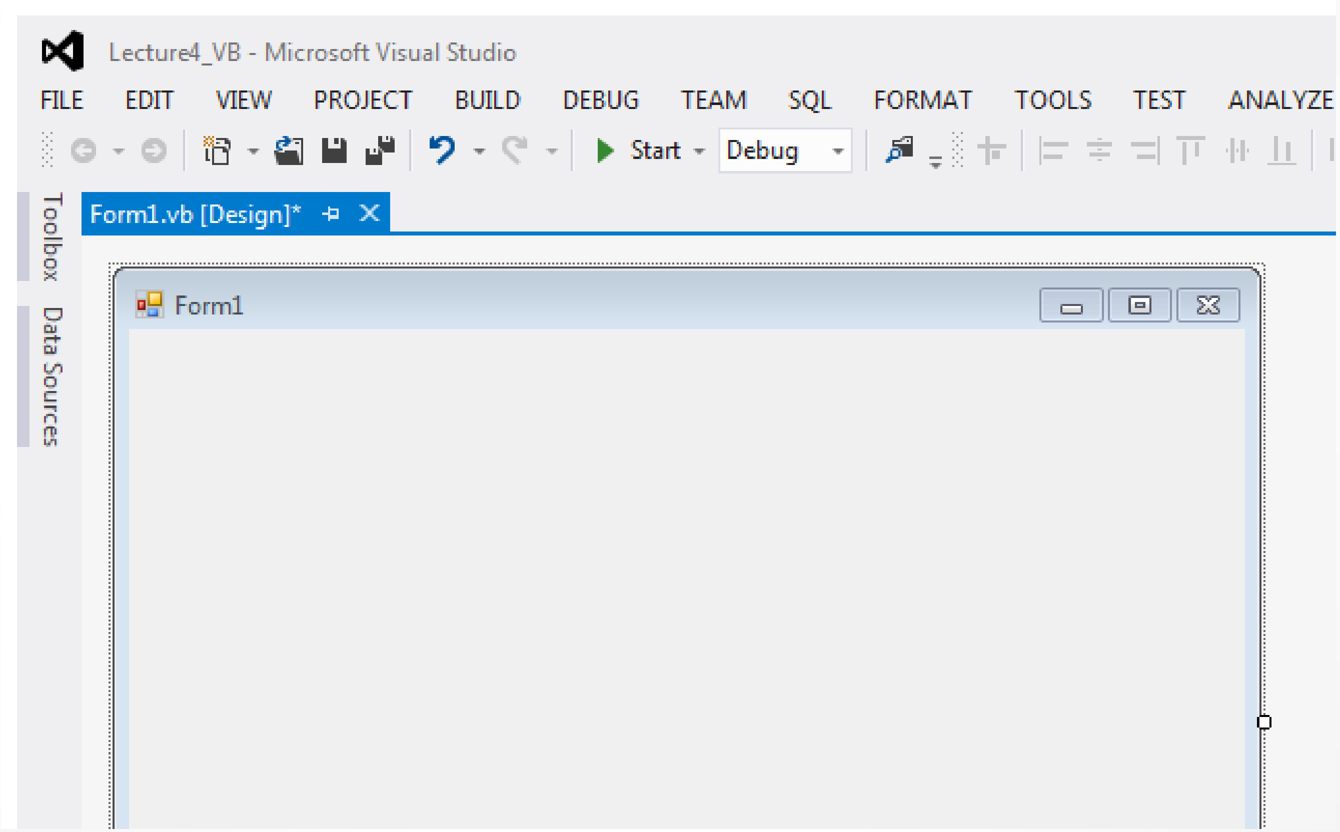
.NET Framework 4.5

Sort by: Default

Name: WindowsApplication1

Form-based Project

You should get a screen like this

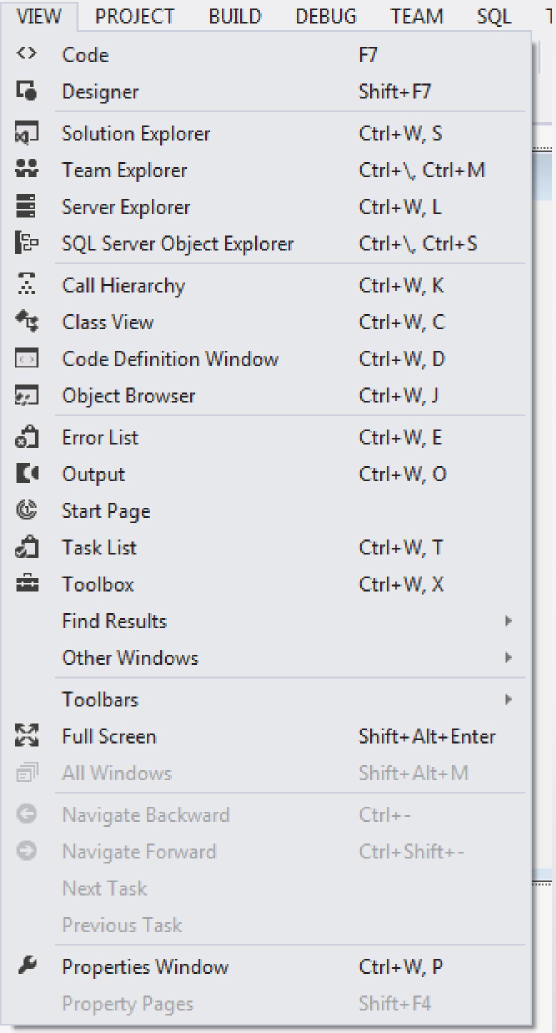


Configuring the Project

There are three important browsers we want to use in Visual Studio. These are:

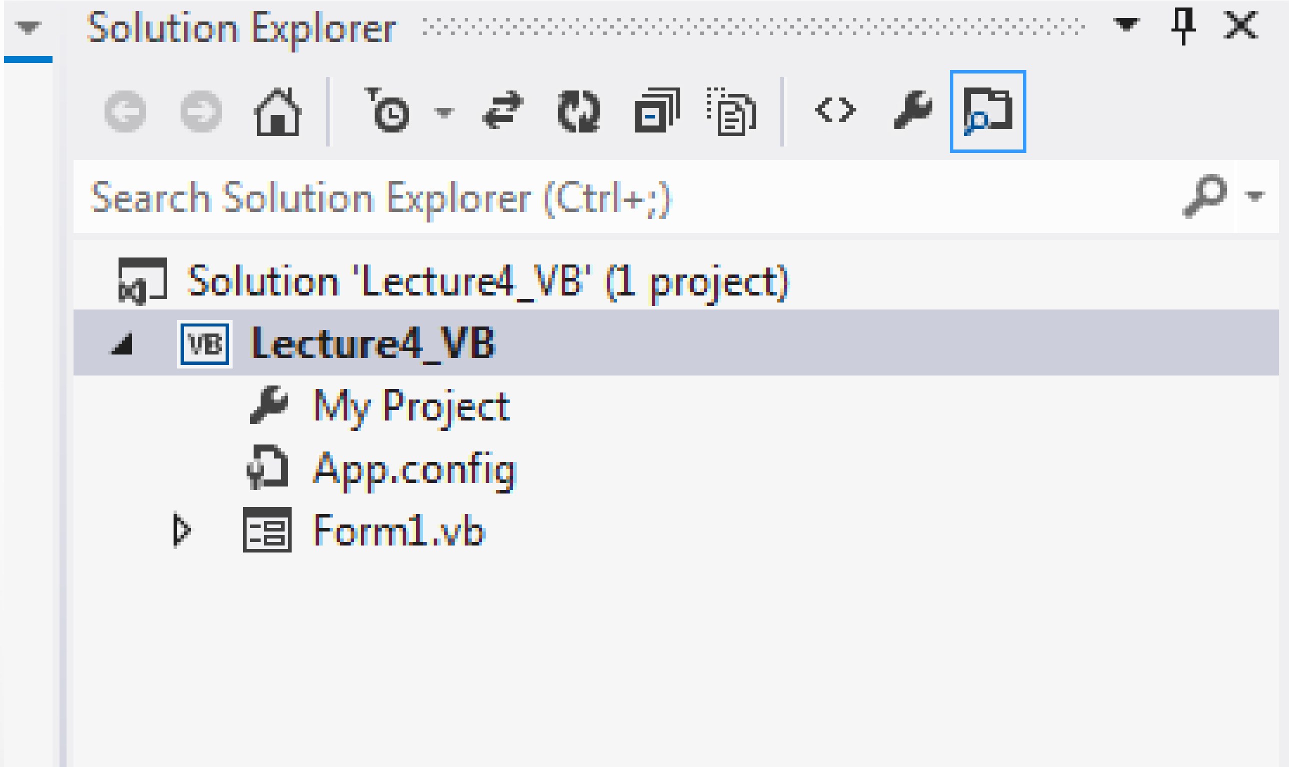
- Toolbox
- Properties Window
- Solution Explorer

If these are not visible on your screen you can add them from the View menu.



Solution Explorer

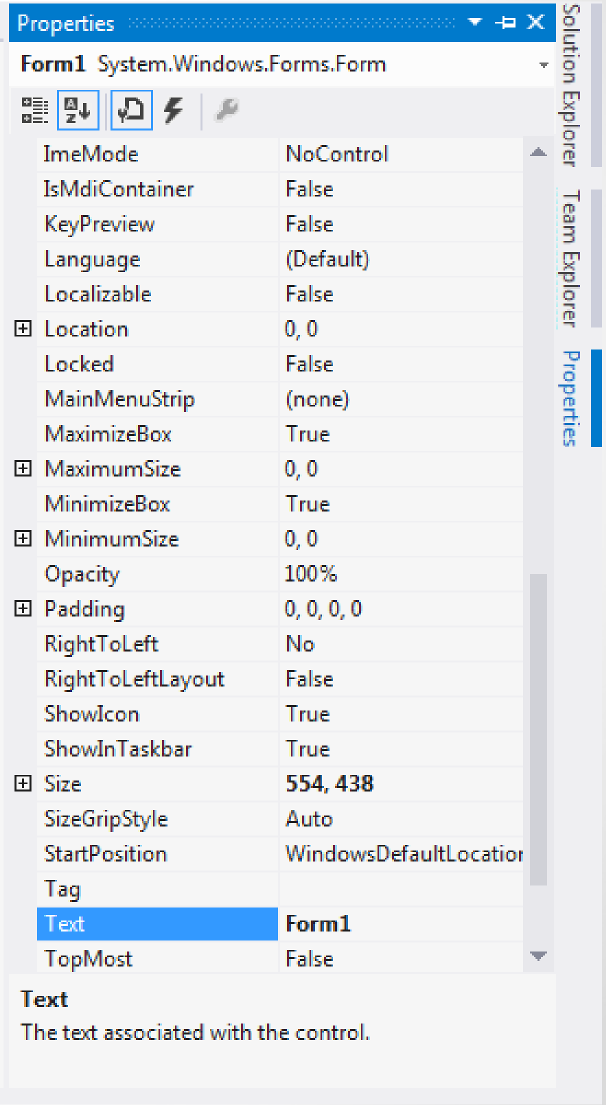
The Solution Explorer shows all of the files in your project and allows you to manage resources for you project (ie. images).



Solution 'Lecture4_VB' (1 project)
Lecture4_VB
My Project
App.config
Form1.vb

Properties Window

The properties window allows you to edit the properties of the form and the controls you place on it.



Form1 System.Windows.Forms.Form

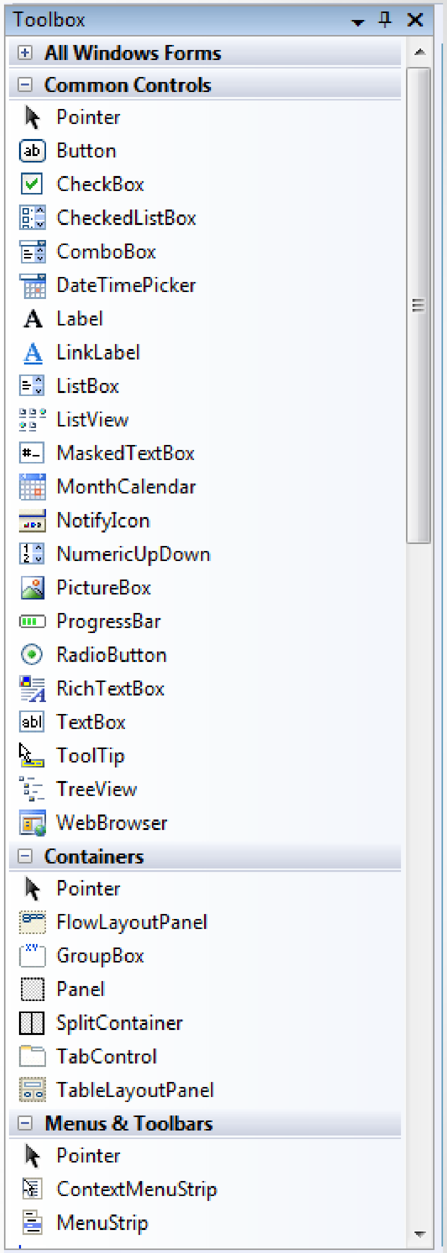
ImeMode	NoControl
IsMdiContainer	False
KeyPreview	False
Language	(Default)
Localizable	False
⊞ Location	0, 0
Locked	False
MainMenuStrip	(none)
MaximizeBox	True
⊞ MaximumSize	0, 0
MinimizeBox	True
⊞ MinimumSize	0, 0
Opacity	100%
⊞ Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
ShowInTaskbar	True
⊞ Size	**554, 438**
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocatior
Tag	
Text	**Form1**
TopMost	False

Text

The text associated with the control.

Toolbox

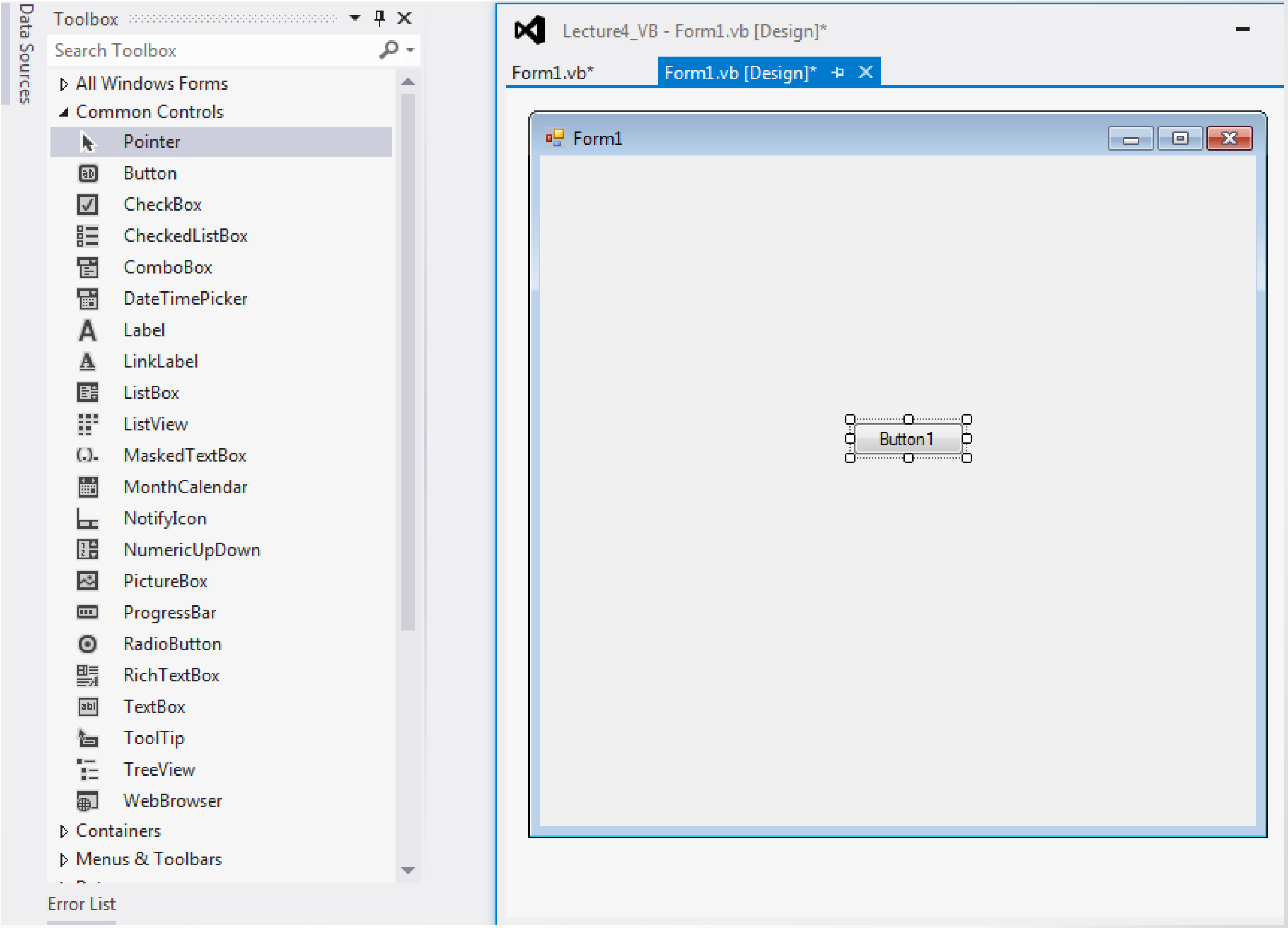
- The toolbox contains a set of *controls* (e.g. textbox, labels, buttons)
- You can drag and drop onto the form.
- For the assignment we are going to use three of these controls
- Label, TextBox and Button



Adding Controls

- Simply click on the desired control in the ToolBox and click on the form.

- An *instance* of that control is created and added to the form.



Adding Controls

- When you add controls to a form, VS automatically names them for you, e.g.,
    - if you add labels, the names will be ‘label1', 'label2', 'label3' etc.
    - If you add textboxes, the names will be 'textBox1', 'textBox2‘

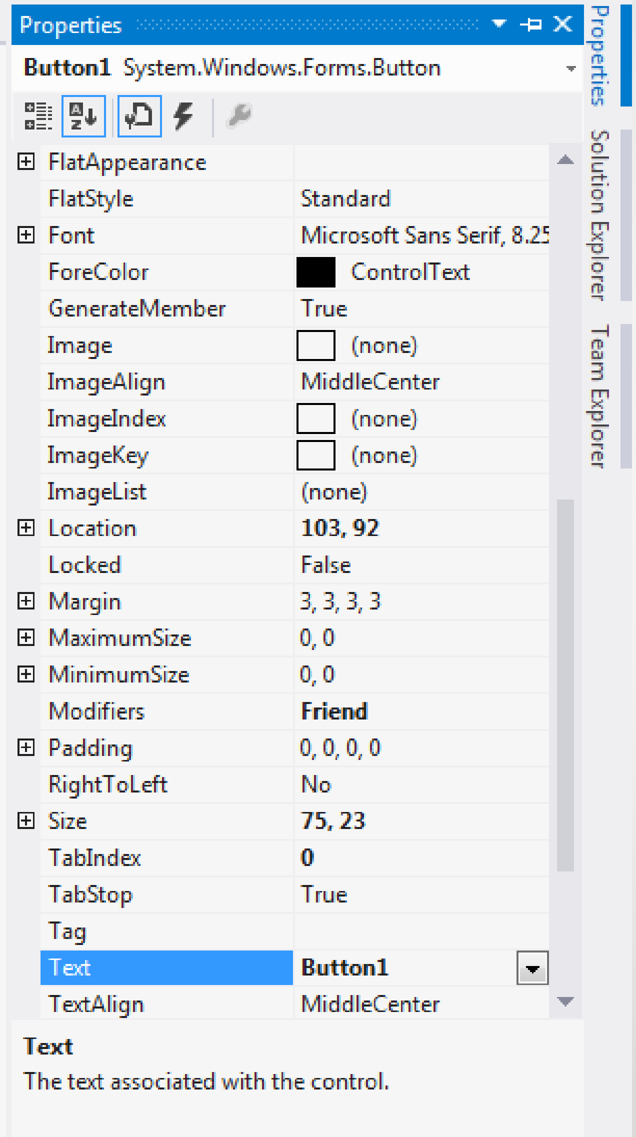
- You can use these names to access these controls within the code.

- Of course, you can also change them to have more useful names.

Control Properties

- If you click on the control, all the public members (properties) of that control show up in the property window.

- This shows all the data members of the object that you can access and change.



Button1 System.Windows.Forms.Button

⊞ FlatAppearance	
FlatStyle	Standard
⊞ Font	Microsoft Sans Serif, 8.25
ForeColor	■ ControlText
GenerateMember	True
Image	□ (none)
ImageAlign	MiddleCenter
ImageIndex	□ (none)
ImageKey	□ (none)
ImageList	(none)
⊞ Location	**103, 92**
Locked	False
⊞ Margin	3, 3, 3, 3
⊞ MaximumSize	0, 0
⊞ MinimumSize	0, 0
Modifiers	**Friend**
⊞ Padding	0, 0, 0, 0
RightToLeft	No
⊞ Size	**75, 23**
TabIndex	**0**
TabStop	True
Tag	
Text	**Button1**
TextAlign	MiddleCenter

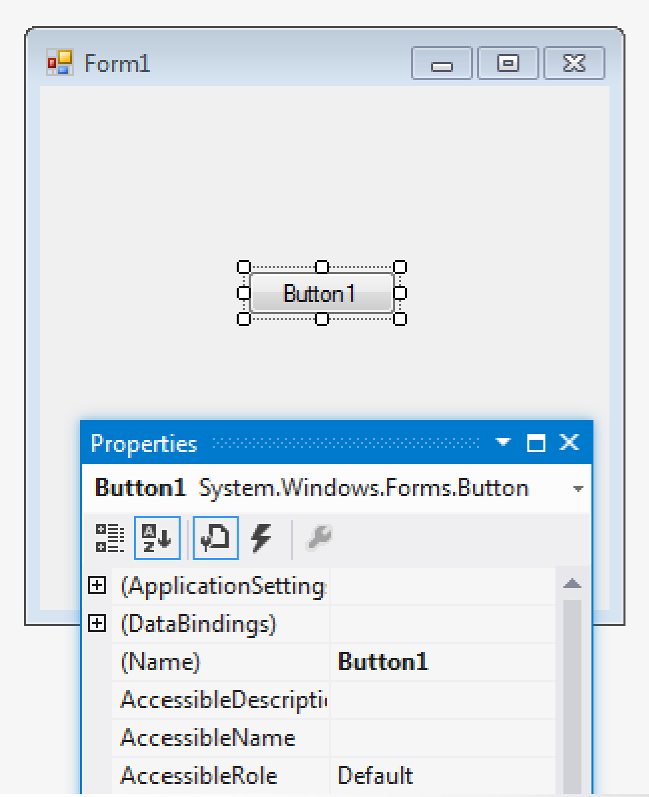
Text
The text associated with the control.

Commonly-used Properties

- The *name* property – the identifier

- To change the name: select the control, click on the *name* property, and type in new name

- The code will now use the new name to refer to that control



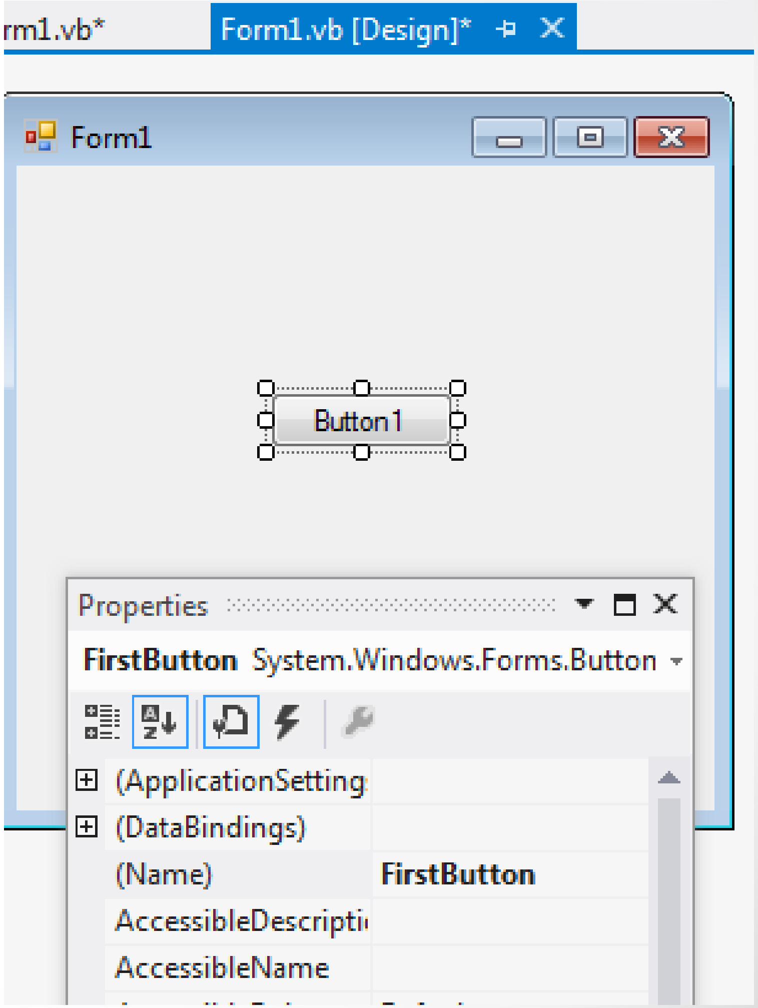
Button1 System.Windows.Forms.Button

⊞ (ApplicationSetting	
⊞ (DataBindings)	
(Name)	**Button1**
AccessibleDescripti	
AccessibleName	
AccessibleRole	Default

Changing Properties

- This doesn't change how the label appears on the form, just the name used to identify it.

- To change what the label displays, the Text property must be changed.

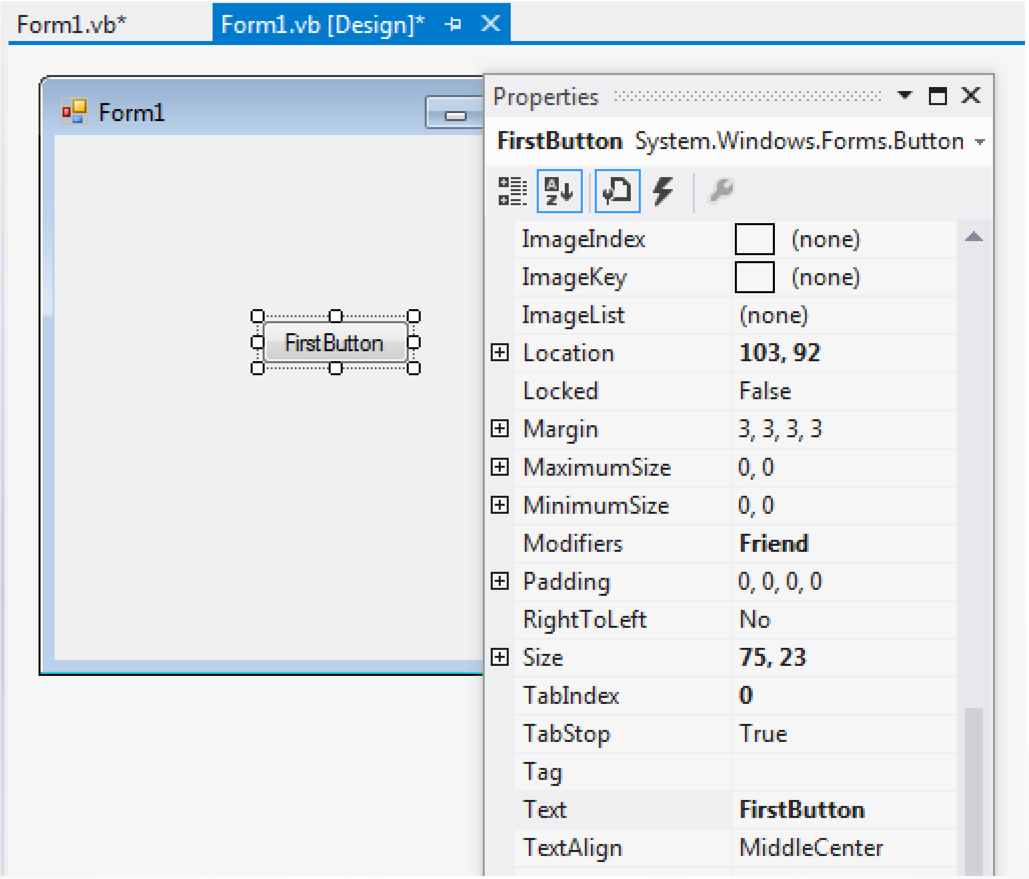


FirstButton System.Windows.Forms.Button

⊞ (ApplicationSetting	
⊞ (DataBindings)	
(Name)	**FirstButton**
AccessibleDescripti	
AccessibleName	

Changing Properties

Now the control *FirstButton* displays "FirstButton"

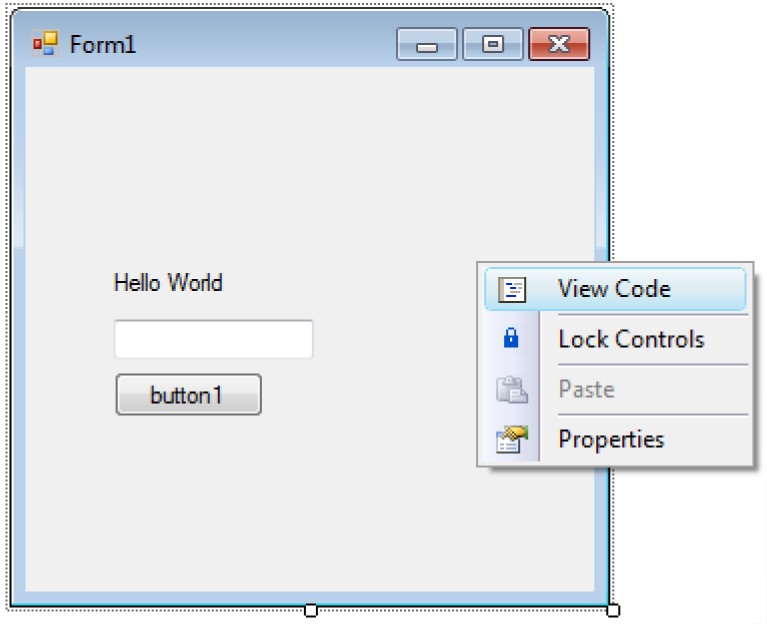


FirstButton System.Windows.Forms.Button

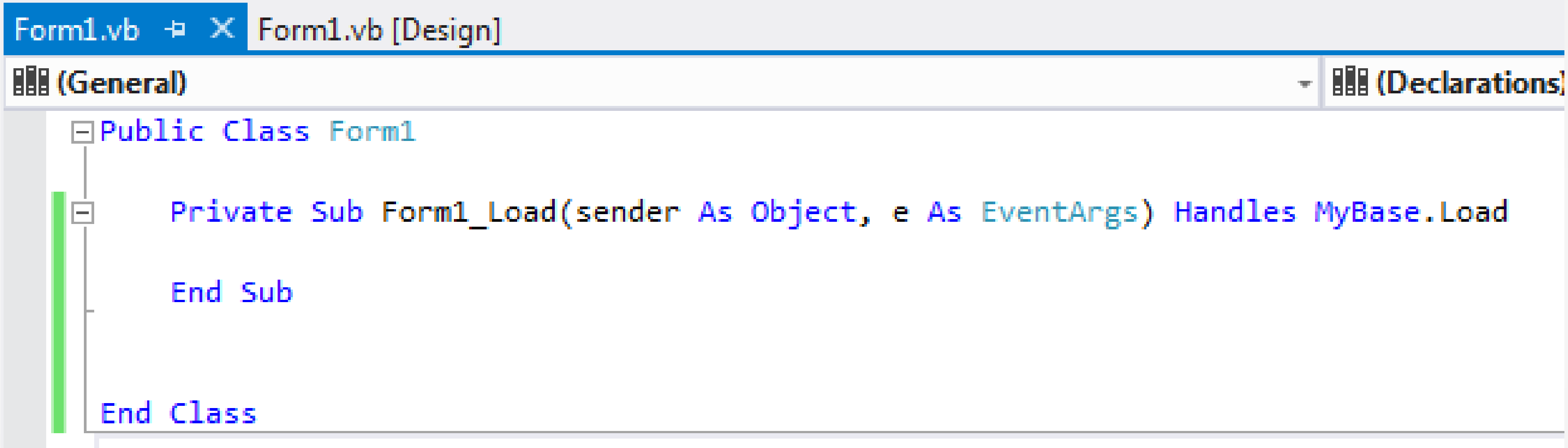
	ImageIndex	(none)
	ImageKey	(none)
	ImageList	(none)
⊞	Location	**103, 92**
	Locked	False
⊞	Margin	3, 3, 3, 3
⊞	MaximumSize	0, 0
⊞	MinimumSize	0, 0
	Modifiers	**Friend**
⊞	Padding	0, 0, 0, 0
	RightToLeft	No
⊞	Size	**75, 23**
	TabIndex	**0**
	TabStop	True
	Tag	
	Text	**FirstButton**
	TextAlign	MiddleCenter

Code-view and Design-view

- We have so-far seen the *design view*
  - So far all the editing we have done has been through the design view – WYSIWYG
- However, there are codes behind it to control the behaviours better (right click → View Code)



Viewing the Code

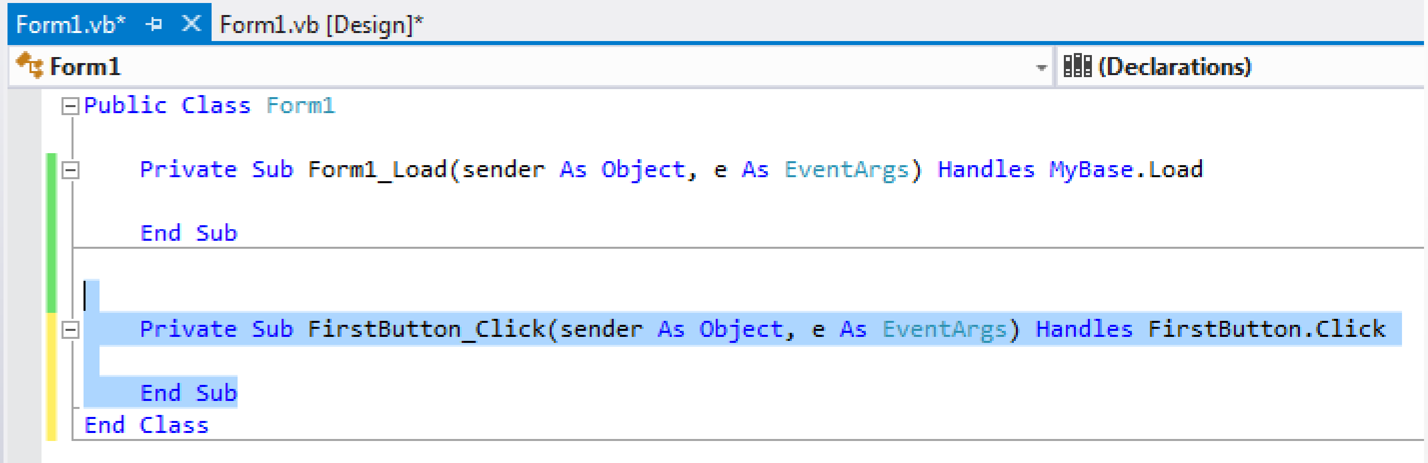


```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub


End Class
```

- The code you see is ***not*** all the code to make the form and controls appear.
- That code is managed by the designer and so you cannot edit it directly.

Viewing the Code

- The code View is useful for adding behaviour to the controls. I
- f you go double click on the Button, it will take you back to the code window but you will see a new subroutine.
- This subroutine will now be called every time the user clicks on that button.

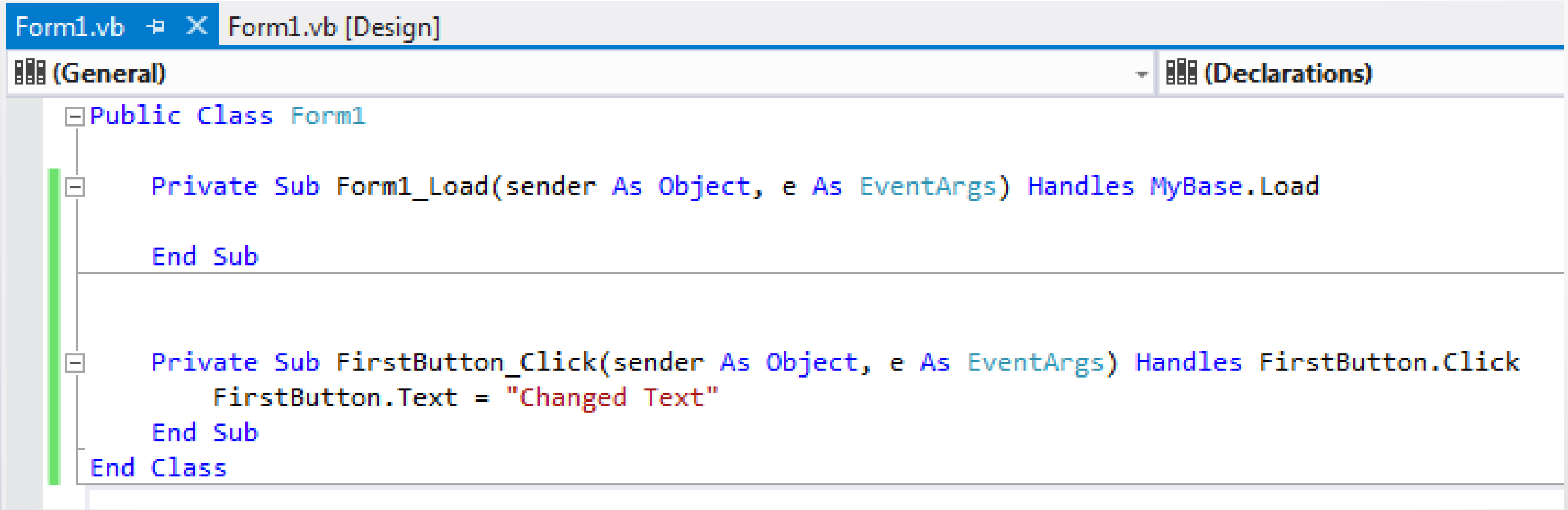


```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub


    Private Sub FirstButton_Click(sender As Object, e As EventArgs) Handles FirstButton.Click

    End Sub
End Class
```

Viewing the Code

- From this subroutine, the other controls that have been added to the form can be accessed, together with their public methods and members.

- For instance the Text property of ‘FirstButton‘ can be changed as follows:

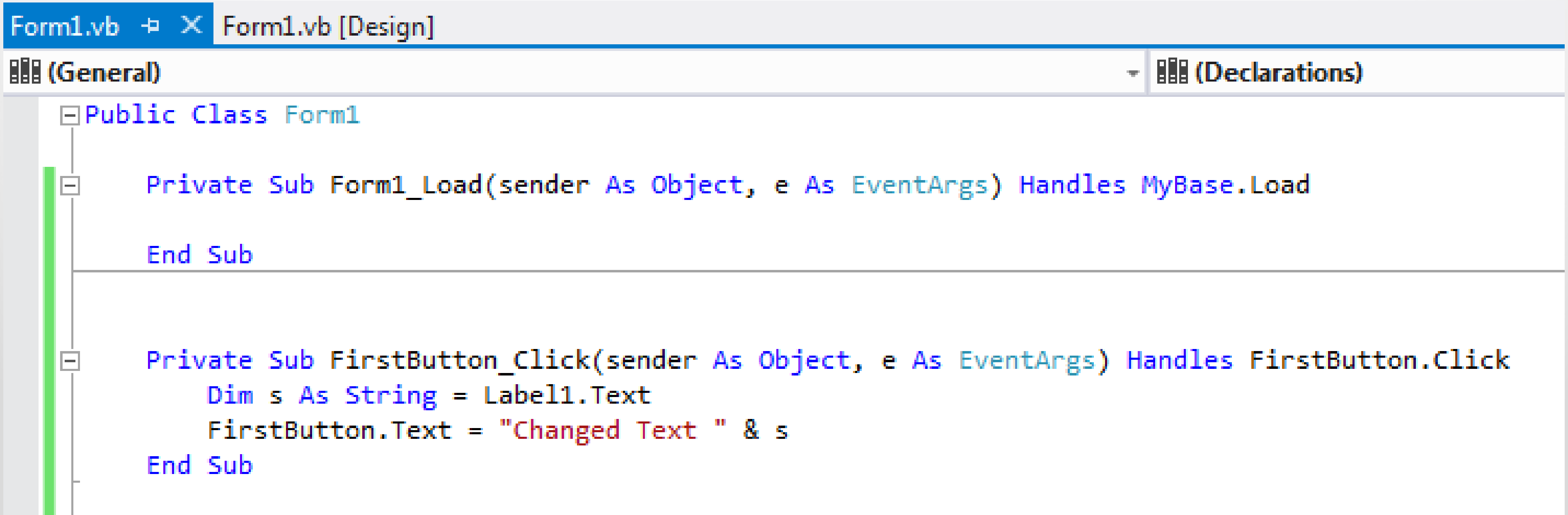


```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub


    Private Sub FirstButton_Click(sender As Object, e As EventArgs) Handles FirstButton.Click
        FirstButton.Text = "Changed Text"
    End Sub
End Class
```

Viewing the Code

- This code can also read from other controls, e.g.
  - The text value of a label

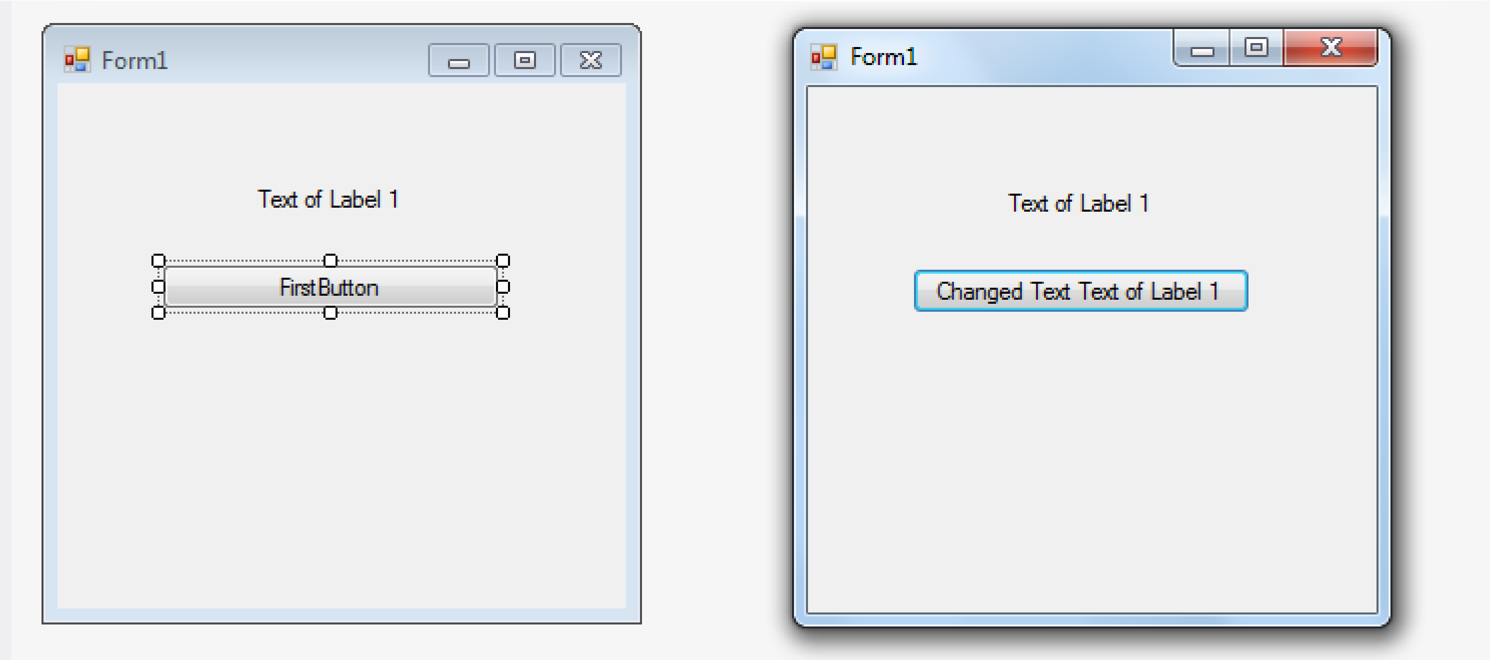
- The button click subroutine could read this value and change the button text to a concatenated string



```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub


    Private Sub FirstButton_Click(sender As Object, e As EventArgs) Handles FirstButton.Click
        Dim s As String = Label1.Text
        FirstButton.Text = "Changed Text " & s
    End Sub
```

Running the Program

- When this program is first run, the label of the label still displays the text “FirstButton”
- But upon a click of the button, it changes the text as programmed



Assignment 2

- You may need to use more controls for your assignment.
- This lecture gives you the knowledge to get you started
- Explore available controls and properties!

Summary

- The OOP groups data and behaviour into classes
  - Model the real-world closer in the programs

- OOP manages complexity through *encapsulation*

- Visual Studio IDE allows visual programming and development of GUIs

- Windows forms are one of such technology
- Provides controls such as text boxes, labels, and buttons.