Introduction to Computer Programming

Ronald P. Kessler
Santiago Canyon College

Developer Tools

- Visual Studio
- VB
- C#
- C++
- F#
- JavaScript
- Python
- Xamarin
- SQL Server
- ASP
- ASP MVC
- Universal Windows Apps
- Internet of Things
- Windows IoT Core
- .Net Micro Framework
- Arduino IDE for Visual Studio

People & Places

Weather

Photos

Mail

Music

Camera

Calendar

Learning to Code with Visual Basic©

Introduction to Computer Programming

Visual Basic©
Fall 2020

Hands-On Learning Series for my CMPR-105 Course

Ronald P. Kessler, Ph.D., M.C.S.E

Table of Contents

INTRODUCTION

Are you ready to learn computer programming?

Great! Let's get started. This book has been designed for the absolute beginner. If you want to learn how to use what Microsoft calls the .NET programming languages to build desktop, robotics, home automation, tablet, and phone applications then you are in the right place.

My name is Ron Kessler and I taught programming & robotics at Santiago Canyon College in Orange, CA until I retired in June 2018 (see www.RKessler.com). I have discovered that when my students need help with their projects, the majority of online tutorials and books are just too advanced and therefore not very helpful to them. Consequently, I created my own website, sample projects and videos for them and now you can learn from them also.

As you begin your programming adventure, please keep a couple things in mind. First, the main purpose of coding is to make a device do something useful or cool for us. Second, the devices you are communicating with have no built-in intelligence. Machines have no understanding of what you want it to do. It is made up of electronic components that cannot learn anything. Therefore, we have to learn to write programming instructions that can be interpreted or translated into some form that the machine *can* use. You must learn to tell the device exactly what you want it to do and it responds by processing each line of your programming code one miserable line at a time! The good news is that it can process millions of instructions per second.

To get started programming in this course, you will need a copy of Microsoft© Visual Studio. This is the program we will be using to create all types of applications. As I am writing this book, VS 2019 has just been released so you should use that unless you already have an older version on your computer. VS 2017 works just fine.

The easiest way to get VS is to download a free version called Visual Studio Community Edition. With this application you will be able to program in Visual Basic, C#, C++, JavaScript, & Python for Windows desktop, Windows 10 store apps and Windows Phone. It is a great program that will let you build projects for nearly any device. You can even create Apple iPad, iPhone, and Android apps all within Visual Studio! This is too cool.

You may be wondering why there are so many different programming languages. The reason is that software development started in the 1960's. Over time, people discovered more and more uses for computers and needed to solve different problems. Some programming languages work well when you want to create a game. Others work well when designing robots or home automation devices. The good news is that Visual Studio gives you several options so you can create apps in the language of your choice. That way, you can use the language you are most comfortable with and use it for almost all of your projects.

So, to summarize, you will be learning to program your computer using a language called Visual Basic which is included in Visual Studio. I like to start with this language because it was designed to let you focus on learning coding and not be overwhelmed by the syntax or complexities that other languages use.

How to Use This Book

I designed this textbook to accompany my CS105 course at my website at RKessler.com. From my home page you can find courses I teach from the top menu. When you navigate to CS105 you will see a series of lessons on the left and the actual topics and sample projects we will be exploring in the center column. Whether you are enrolled in my class or not, I want you to feel like you are. So, think of the videos as my lectures. Think of the projects as in-class demos. *If you are in the class, check the website for your schedule so you will get home work and other assignments in on time.*

Each project has a link so you can download it to your computer. You simply unzip it and open it in Visual Studio (VS). Each project also has a video. You can click on the link and watch me build the project! I strongly suggest you watch the video and build the project along with me. You can pause each of the videos so you can work at your own pace. I want you to feel like you are in the classroom with me as we build projects together.

Please understand that my book does not take the place of the recommended textbook. The Murach VB book I want you to read is going to introduce you to many topics you will need in your programming career. You should use the Murach book to expand your general understanding of programming and use my book like a guide to my specific class. *My* book is designed to help you navigate through my course and allows me to teach you the specific things I think you should know.

Getting Started

In this lesson you will learn to set up your programming environment and actually create your first Windows VB desktop application. Be sure to follow along with my videos and take good notes. I strongly recommend you create the projects I show in the videos along with me!

On my home page, there is a “How to Section”. Please look at the video which shows how to package up homework and send it to me. This must be done in a specific manner or I won’t get your entire project.

Lesson 1: Getting Started with Computer Programming

Textbook Chapters & Videos for Lesson 1

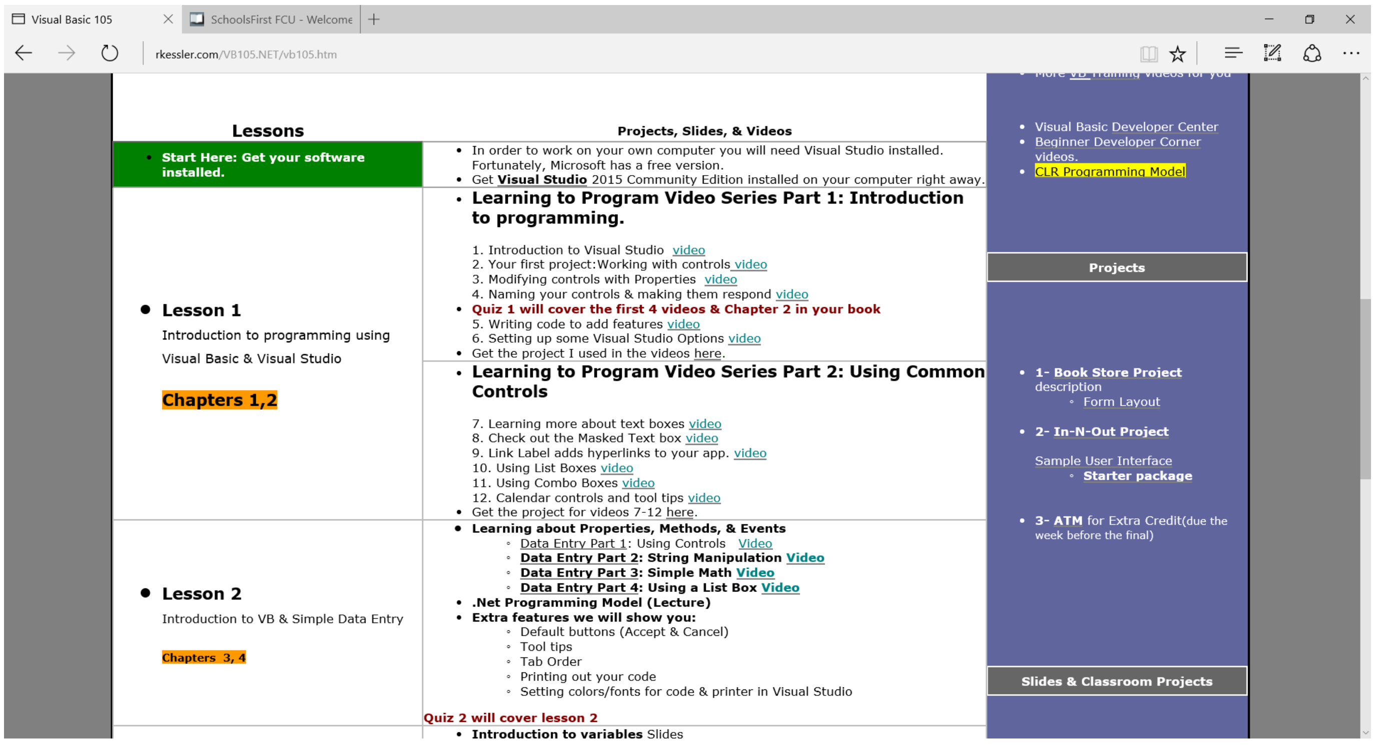
- Murach Chapters 1 & 2
- Videos 1-6 on your VB home page

Step 1
Getting your software & textbook

First of all, please order your Murach textbook so you can start working through the sections I assign as we move through the course.

Next, navigate to RKessler.com and choose “Courses” from the top menu. Then click on “Introduction to Programming Using VB”. This takes you to the CS 105 page. All the material you need is on this page. I have all my lessons, homework assignments, and a lot of resources for you to look at. Look at the Green box that says Get your software installed. In the center box on that row is a link to download VS community edition. Download that software and get it installed now.

Figure 1: My VB Lessons



Lessons	Projects, Slides, & Videos
• Start Here: Get your software installed.	• In order to work on your own computer you will need Visual Studio installed. Fortunately, Microsoft has a free version. • Get **Visual Studio** 2015 Community Edition installed on your computer right away.
• **Lesson 1** Introduction to programming using Visual Basic & Visual Studio **Chapters 1,2**	• **Learning to Program Video Series Part 1: Introduction to programming.** 1. Introduction to Visual Studio video 2. Your first project:Working with controls video 3. Modifying controls with Properties video 4. Naming your controls & making them respond video • **Quiz 1 will cover the first 4 videos & Chapter 2 in your book** 5. Writing code to add features video 6. Setting up some Visual Studio Options video • Get the project I used in the videos here. • **Learning to Program Video Series Part 2: Using Common Controls** 7. Learning more about text boxes video 8. Check out the Masked Text box video 9. Link Label adds hyperlinks to your app. video 10. Using List Boxes video 11. Using Combo Boxes video 12. Calendar controls and tool tips video • Get the project for videos 7-12 here.
• **Lesson 2** Introduction to VB & Simple Data Entry **Chapters 3, 4**	• **Learning about Properties, Methods, & Events** ◦ Data Entry Part 1: Using Controls Video ◦ **Data Entry Part 2: String Manipulation Video** ◦ **Data Entry Part 3: Simple Math Video** ◦ **Data Entry Part 4: Using a List Box Video** • **.Net Programming Model (Lecture)** • **Extra features we will show you:** ◦ Default buttons (Accept & Cancel) ◦ Tool tips ◦ Tab Order ◦ Printing out your code ◦ Setting colors/fonts for code & printer in Visual Studio **Quiz 2 will cover lesson 2**
	• **Introduction to variables** Slides

- More VB Training videos for you
- Visual Basic Developer Center
- Beginner Developer Corner videos.
- CLR Programming Model

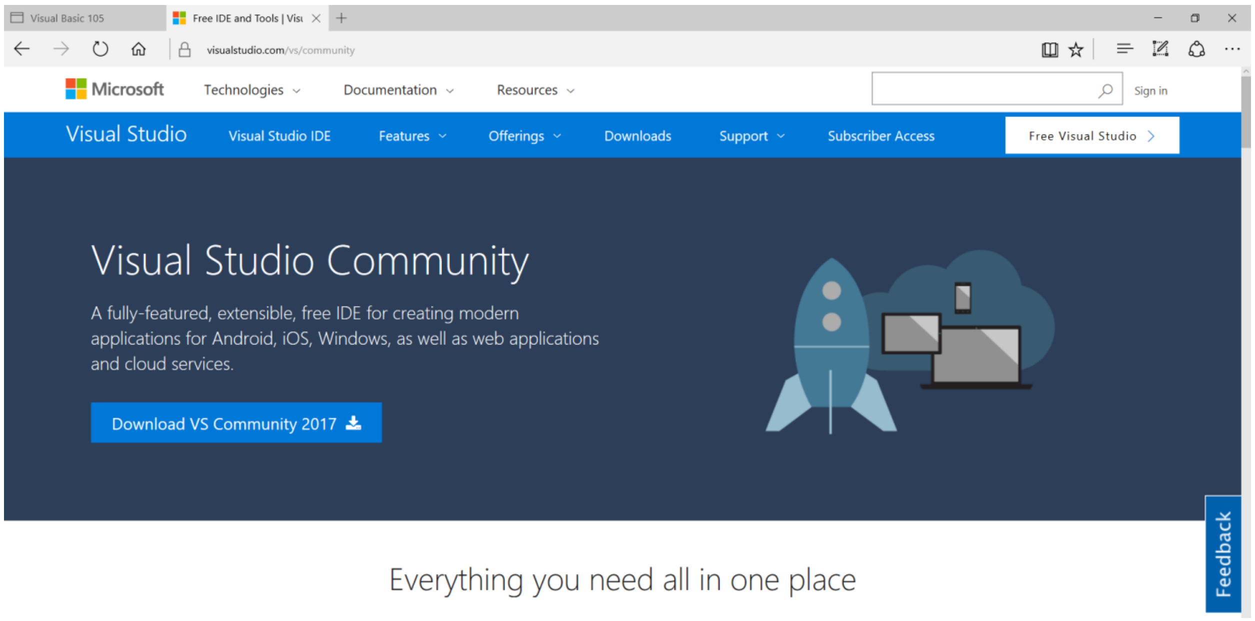
Projects

- **1- Book Store Project** description
  - Form Layout
- **2- In-N-Out Project**

  Sample User Interface
  - **Starter package**
- **3- ATM** for Extra Credit(due the week before the final)

Slides & Classroom Projects

Visual Studio will not run on an Apple® or Linux machine. So, if you have a Mac, you must install Windows first or use the Visual Studio for Mac version. Install Visual Studio from the link on your VB page. You will be directed to a web page that looks like the one shown below.



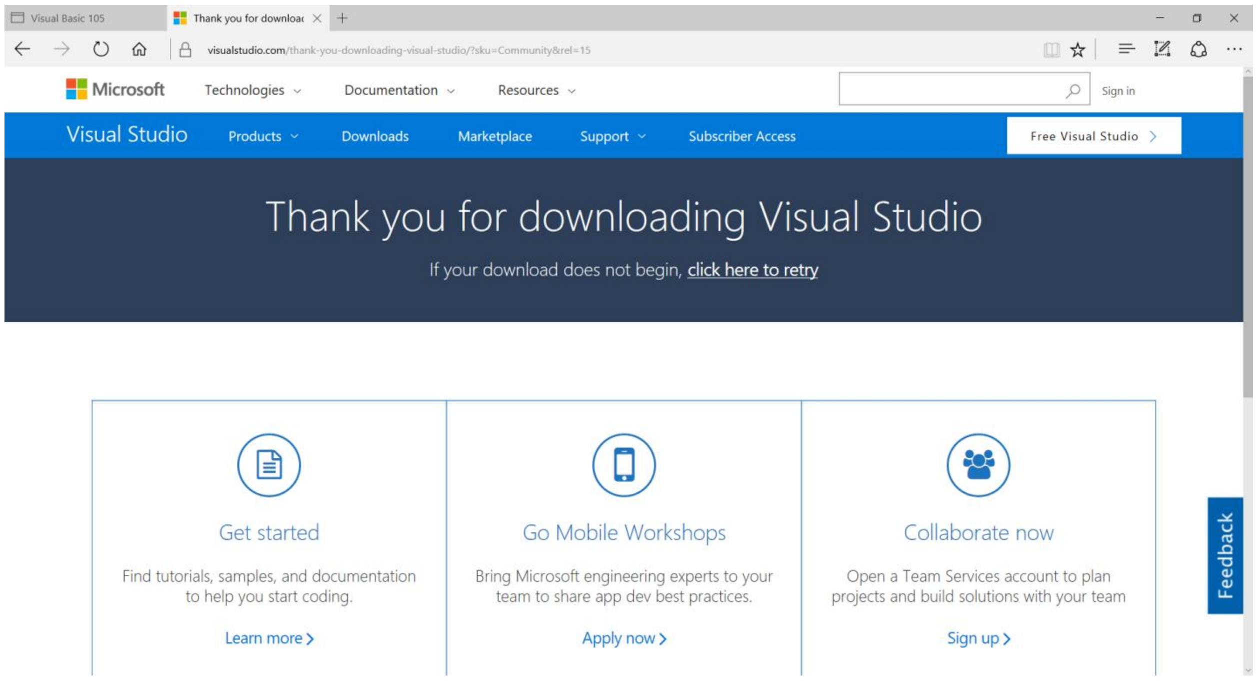
Visual Studio Community

A fully-featured, extensible, free IDE for creating modern applications for Android, iOS, Windows, as well as web applications and cloud services.

Download VS Community 2017

Everything you need all in one place

1. Click Download VS Community.
2. Very quickly, you will see another message saying the installer has downloaded. Choose Run from that menu.
3. Finally, a security warning appears so click Run again to start the installation.



Thank you for downloading Visual Studio

If your download does not begin, click here to retry

Get started

Find tutorials, samples, and documentation to help you start coding.

Learn more >

Go Mobile Workshops

Bring Microsoft engineering experts to your team to share app dev best practices.

Apply now >

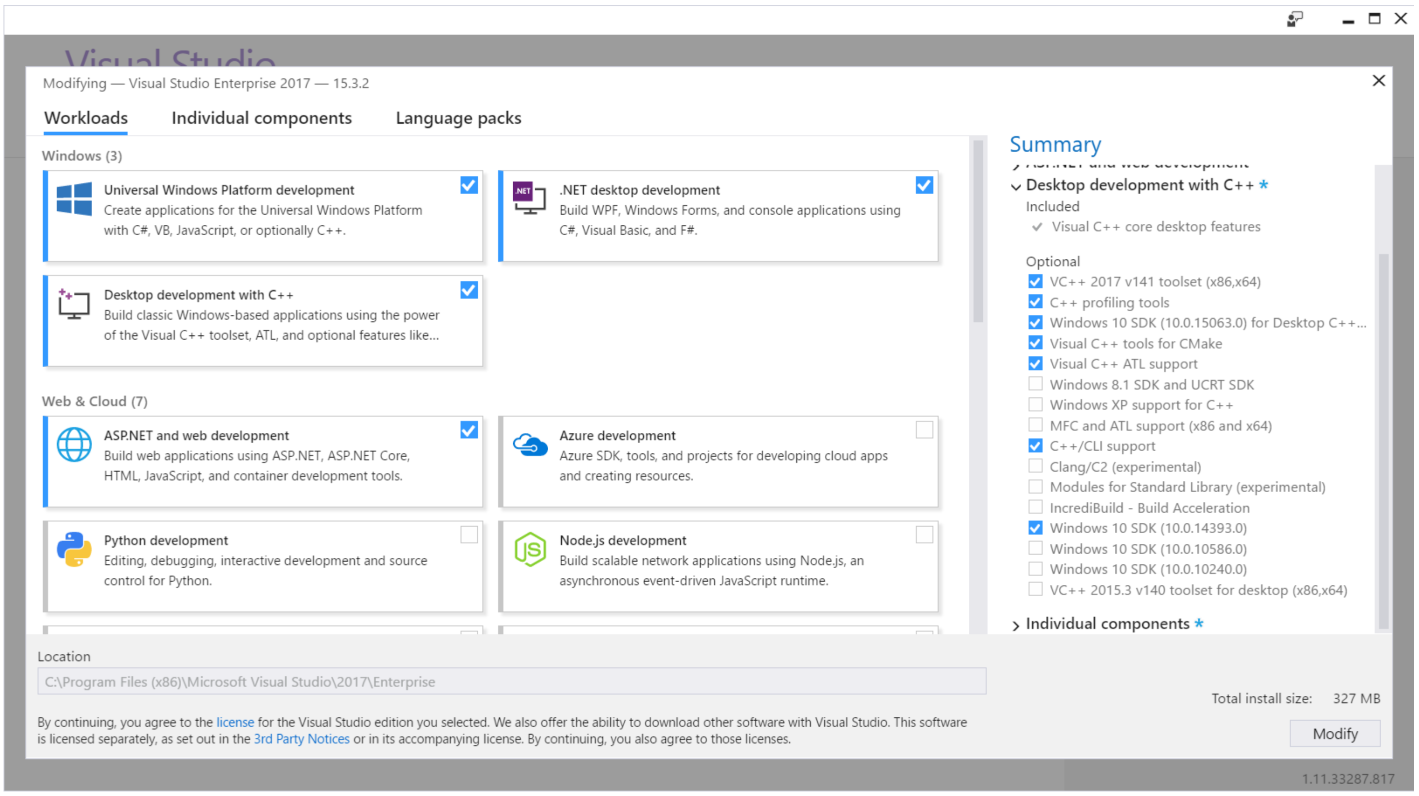
Collaborate now

Open a Team Services account to plan projects and build solutions with your team

Sign up >

- Starting with VS 2017, the setup screen lets you choose which types of applications you want to install. When asked about which features to install, all you need is ".NET desktop development" so check that box.
- If you want to build Web apps then select ASP.NET
- To create Modern Universal Apps, choose Universal Windows Platform
- If you will be taking my C++ classes choose Desktop development with C++ AND C++/CLI support under the optional items on the right side.

Look at my screen shot below to see what I chose. Of course, you can add other stuff or all of the features if you like. You can always add more stuff later by going to Start | Visual Studio 2017 Installer. For now, click Modify and wait until it is finished.



Modifying — Visual Studio Enterprise 2017 — 15.3.2

Windows (3)

- [✓] Universal Windows Platform development — Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.
- [✓] .NET desktop development — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.
- [✓] Desktop development with C++ — Build classic Windows-based applications using the power of the Visual C++ toolset, ATL, and optional features like...

Web & Cloud (7)

- [✓] ASP.NET and web development — Build web applications using ASP.NET, ASP.NET Core, HTML, JavaScript, and container development tools.
- [ ] Azure development — Azure SDK, tools, and projects for developing cloud apps and creating resources.
- [ ] Python development — Editing, debugging, interactive development and source control for Python.
- [ ] Node.js development — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Summary

Desktop development with C++ *

Included
- ✓ Visual C++ core desktop features

Optional
- [✓] VC++ 2017 v141 toolset (x86,x64)
- [✓] C++ profiling tools
- [✓] Windows 10 SDK (10.0.15063.0) for Desktop C++...
- [✓] Visual C++ tools for CMake
- [✓] Visual C++ ATL support
- [ ] Windows 8.1 SDK and UCRT SDK
- [ ] Windows XP support for C++
- [ ] MFC and ATL support (x86 and x64)
- [✓] C++/CLI support
- [ ] Clang/C2 (experimental)
- [ ] Modules for Standard Library (experimental)
- [ ] IncrediBuild - Build Acceleration
- [✓] Windows 10 SDK (10.0.14393.0)
- [ ] Windows 10 SDK (10.0.10586.0)
- [ ] Windows 10 SDK (10.0.10240.0)
- [ ] VC++ 2015.3 v140 toolset for desktop (x86,x64)

Individual components *

Location
C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total install size: 327 MB

1.11.33287.817

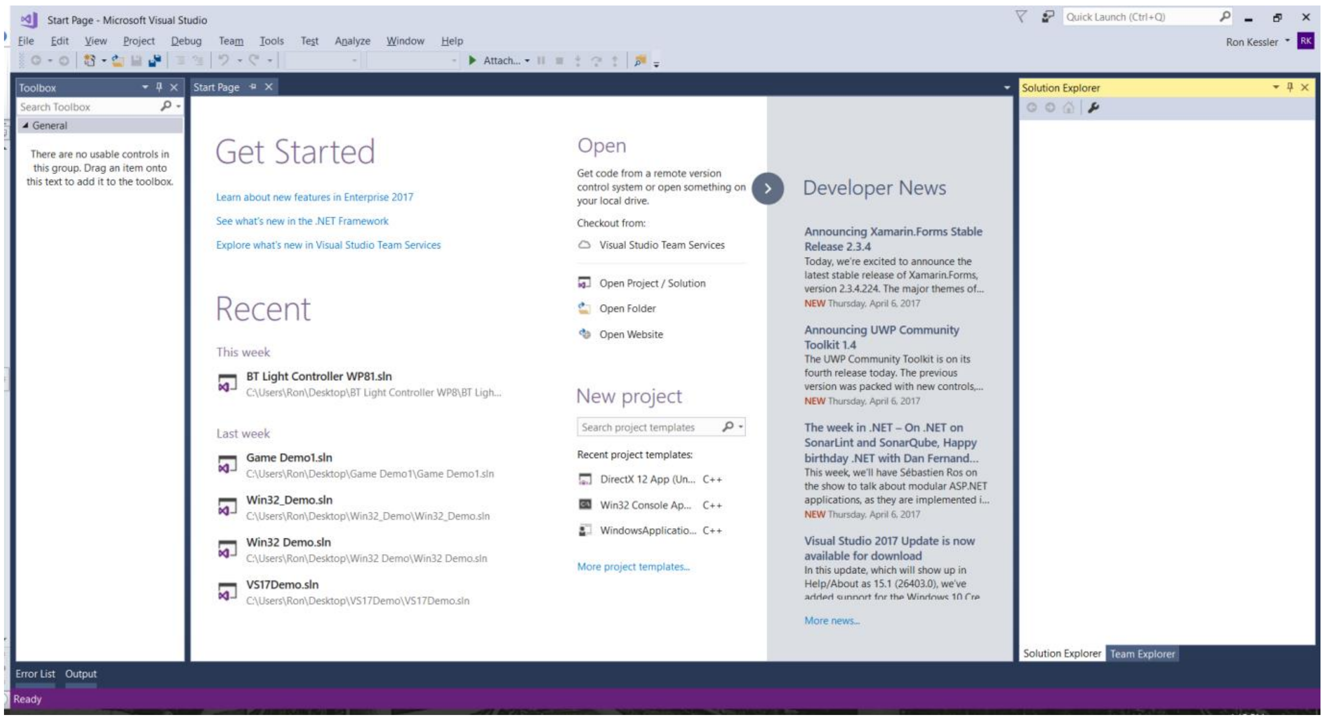
Setup Screen for the Options I chose

Step 2
Visual Studio Updates

As a professional developer, you must learn a great deal of things besides VB. The first is that your development computer must always have the latest operating system (O/S) and Visual Studio updates. You can check my home page for the latest update links for VS. Make sure your computer is configured to receive automatic updates.

From the start menu you should find Visual Studio 2017 so click on it and make sure it runs. Notice that on the start menu you can also run Visual Studio 2017 Installer when you want to modify options.

If all is well you should see a screen like the one below. This is called the start page and is intended to show the latest news about VS and also has a lot of links for online training videos. You can see on the left that you can open an existing project or create a new one. I usually close it by clicking on the little "X" on the Start Page tab. I use the drop down menus at the top to do my work.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Get code from a remote version control system or open something on your local drive.

BT Light Controller WP81.sln
C:\Users\Ron\Desktop\BT Light Controller WP8\BT Ligh...

Game Demo1.sln
C:\Users\Ron\Desktop\Game Demo1\Game Demo1.sln

Win32_Demo.sln
C:\Users\Ron\Desktop\Win32_Demo\Win32_Demo.sln

Win32 Demo.sln
C:\Users\Ron\Desktop\Win32 Demo\Win32 Demo.sln

VS17Demo.sln
C:\Users\Ron\Desktop\VS17Demo\VS17Demo.sln

Announcing Xamarin.Forms Stable Release 2.3.4
Today, we're excited to announce the latest stable release of Xamarin.Forms, version 2.3.4.224. The major themes of...
NEW Thursday, April 6, 2017

Announcing UWP Community Toolkit 1.4
The UWP Community Toolkit is on its fourth release today. The previous version was packed with new controls,...
NEW Thursday, April 6, 2017

The week in .NET – On .NET on SonarLint and SonarQube, Happy birthday .NET with Dan Fernand...
This week, we'll have Sébastien Ros on the show to talk about modular ASP.NET applications, as they are implemented i...
NEW Thursday, April 6, 2017

Visual Studio 2017 Update is now available for download
In this update, which will show up in Help/About as 15.1 (26403.0), we've added support for the Windows 10 Cre

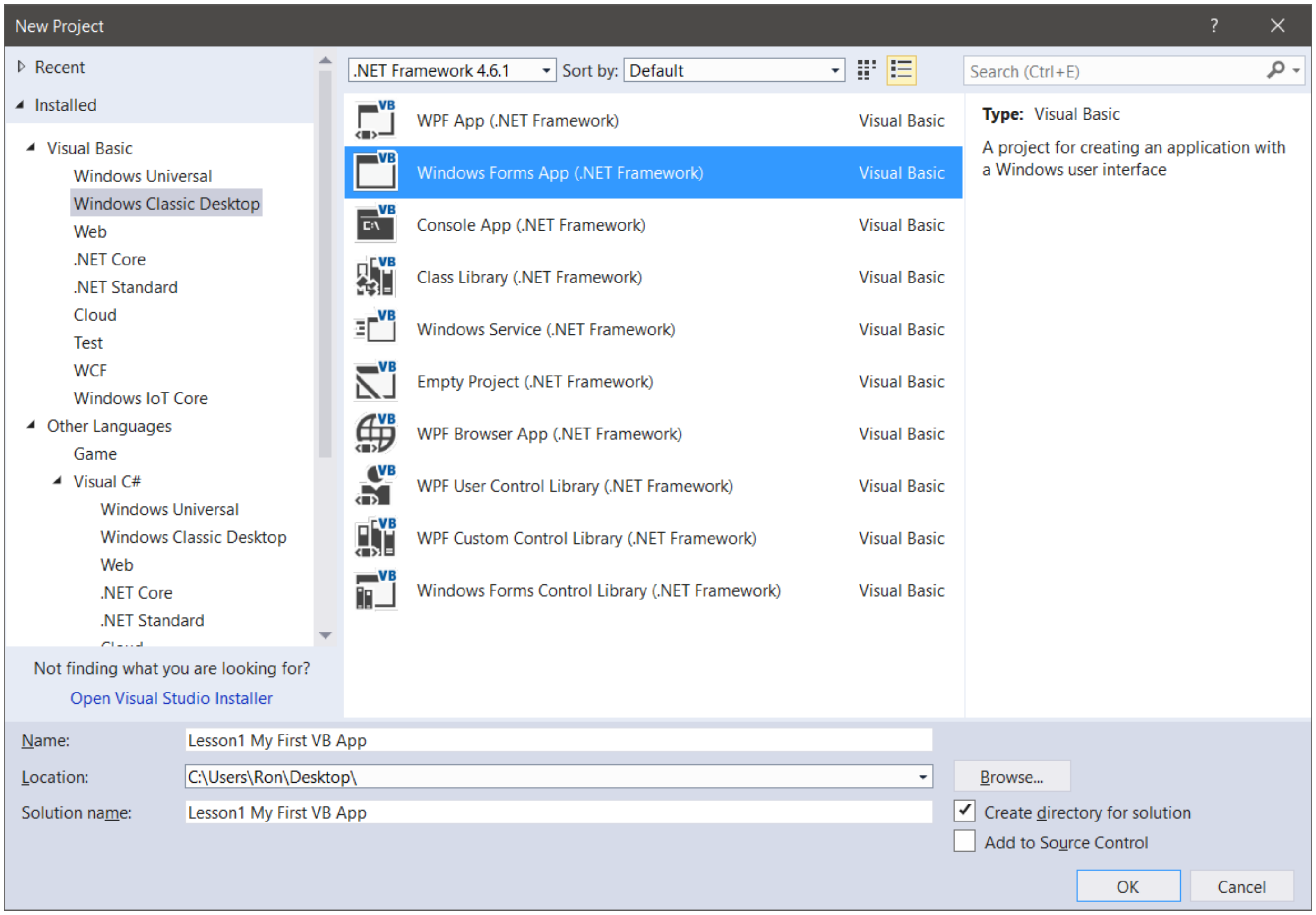
Step 3
Getting to Know Visual Studio

Look at Lesson 1 on your VB page on my website. You will see my "Introduction to Programming Video Series 1".

1. I want you to listen to video #1 on the introduction to Visual Studio now. Be sure to take notes as I am speaking. For now, just watch the video and don't try to build the project yet.
2. When you are finished come back here to move on.

Step 4:
Create your first project

1. Now, open VS and choose New Project from the File menu. Do not choose Web project or anything else.
2. In the New Project dialog box select Visual Basic | Windows Classic Desktop | Windows Form Application (.Net Framework). In the Name box, Type Lesson1 My First VB App and save it to the desktop. Then click OK.


New Project
Recent
Installed
Visual Basic
Windows Universal
Windows Classic Desktop
Web
.NET Core
.NET Standard
Cloud
Test
WCF
Windows IoT Core
Other Languages
Game
Visual C#
Windows Universal
Windows Classic Desktop
Web
.NET Core
.NET Standard
Not finding what you are looking for?
Open Visual Studio Installer
.NET Framework 4.6.1
Sort by: Default
Search (Ctrl+E)
WPF App (.NET Framework) Visual Basic
Windows Forms App (.NET Framework) Visual Basic
Console App (.NET Framework) Visual Basic
Class Library (.NET Framework) Visual Basic
Windows Service (.NET Framework) Visual Basic
Empty Project (.NET Framework) Visual Basic
WPF Browser App (.NET Framework) Visual Basic
WPF User Control Library (.NET Framework) Visual Basic
WPF Custom Control Library (.NET Framework) Visual Basic
Windows Forms Control Library (.NET Framework) Visual Basic
Browse...
Add to Source Control
OK
Cancel

Type: Visual Basic

A project for creating an application with a Windows user interface

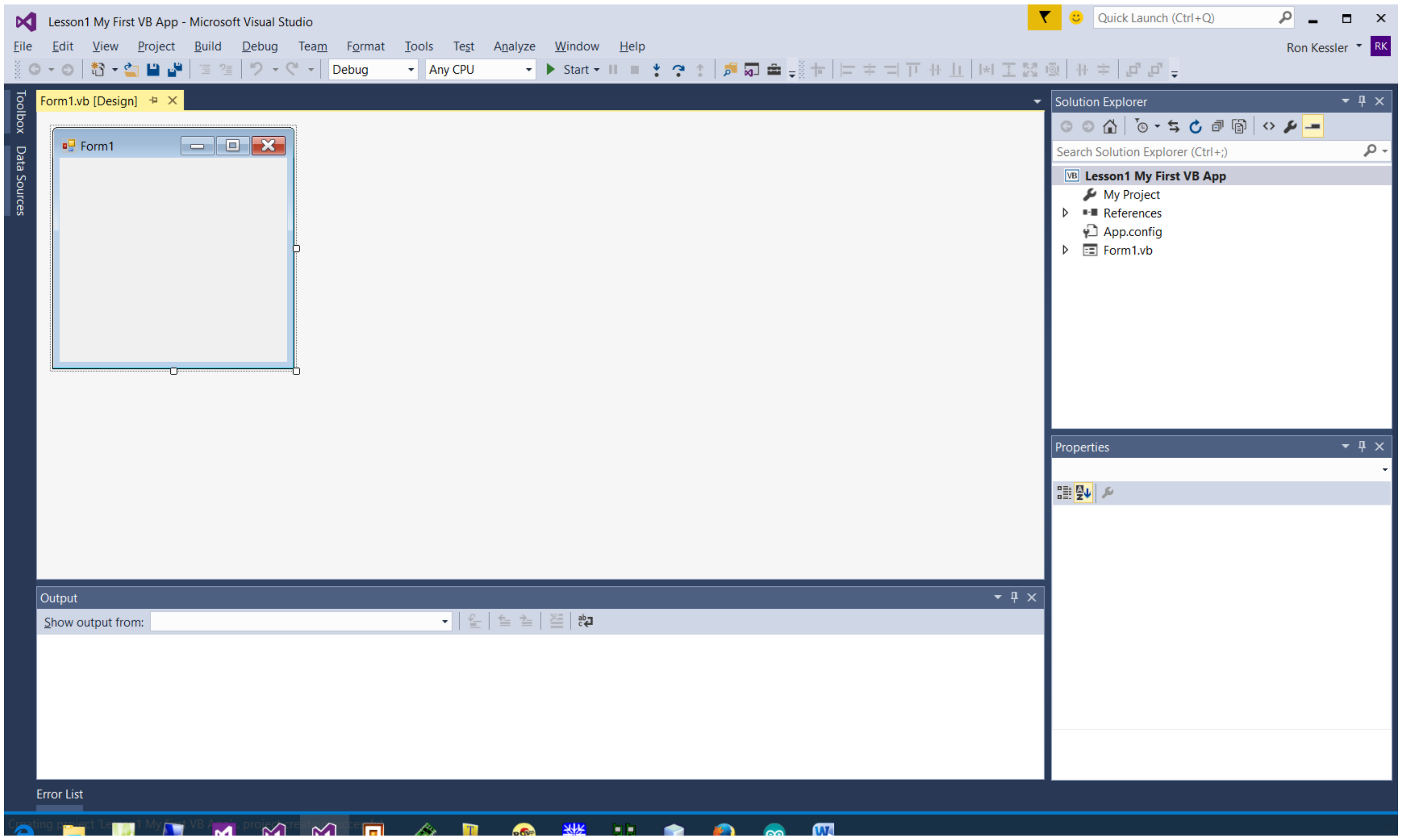
Name: Lesson1 My First VB App

Location: C:\Users\Ron\Desktop\

Solution name: Lesson1 My First VB App

✓ Create directory for solution

Now you will see a screen like mine. There are several sections shown. The top right pane is called Solution Explorer. This is where all the pieces and parts of your project are kept. You can add music, pictures, video, and other things to the project using this pane.



You also notice a blue box that says Form1. This is called a Windows form (Win Form) and this is the actual Window people will see and use when you give them your app.

Visual Studio is called an IDE. That stands for “Integrated Development Environment”. There are many such applications that programmers can use. You may have heard of Eclipse or NetBeans that can be used for Java and other apps, for example. Inside VS you can create apps for the Windows desktop, web applications where people can purchase things, Windows phone apps, and something new called Universal Applications. In VS 2105, Microsoft introduces these Universal apps as being able to run on any Windows 10 device. So, I can create a universal app and it will run on my phone, Surface Tablet, desktop, or my Raspberry Pi microcontroller.

In the latest version of VS, you can create projects for Android and Apple as well. It is now possible to create apps for any device on the planet…all inside one IDE that you are comfortable with!

NOTE: By default, VS2017 changed the location where projects are saved. To make it easy, save your projects inside your Documents Folder under Visual Studio | Projects like I show you below. You can tell VS to save them somewhere else if you want to. Use the little browse button to locate the folder location you want. Many students create a desktop folder for CS105 and keep all homework, notes, and projects in there. Do not change the location for templates!

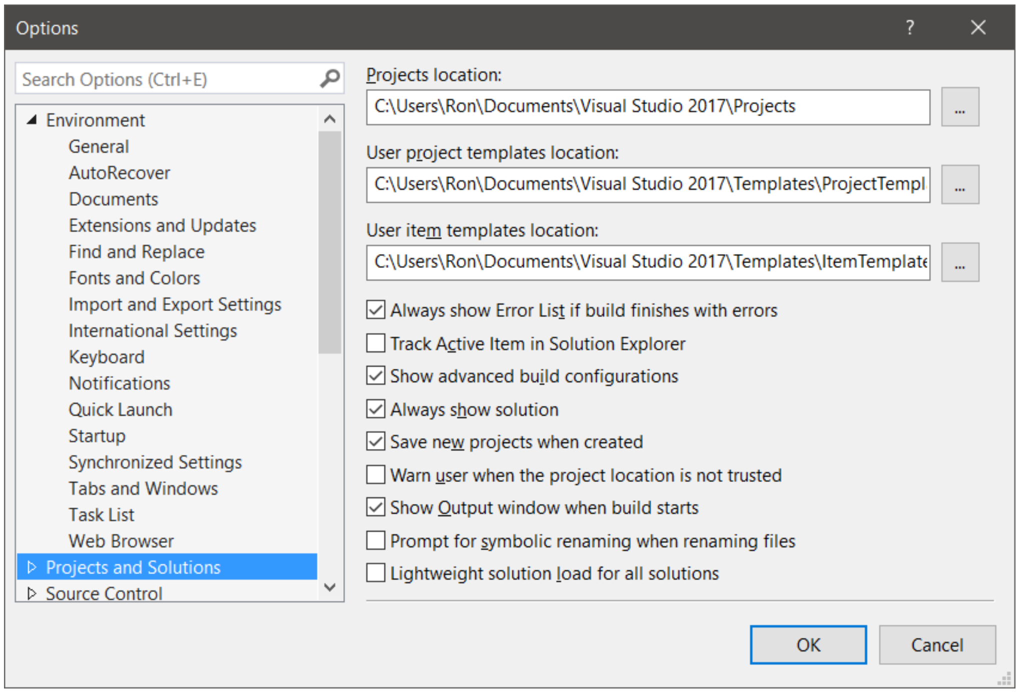
If you use VS 2015, then your project location will be like the one shown which is just fine!

To change common settings in VS menus, choose Tools | Options | Projects & Solutions.

Here, you can change the folder to save projects to.

Choose Fonts to change how your IDE looks when writing code.

Please only select the checked items shown in this figure.



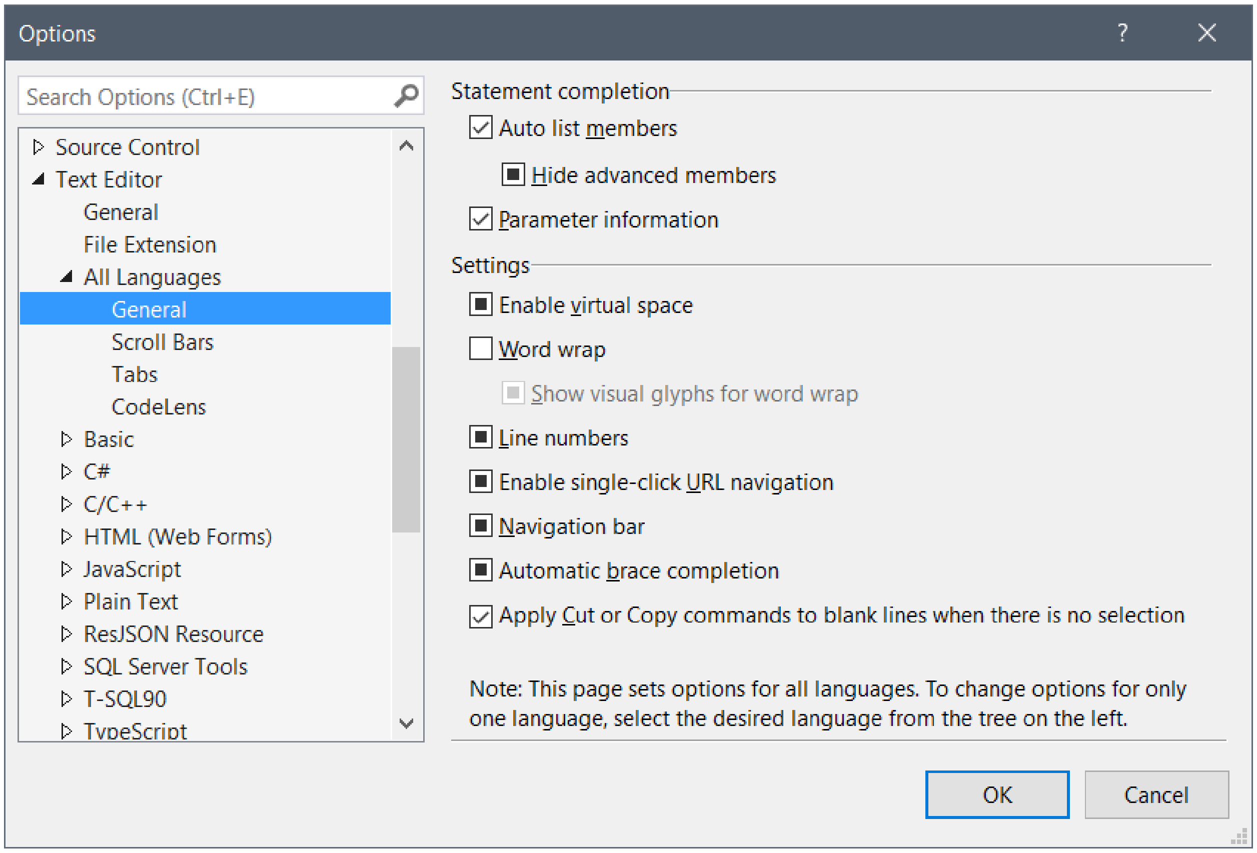
Projects location:
C:\Users\Ron\Documents\Visual Studio 2017\Projects

User project templates location:
C:\Users\Ron\Documents\Visual Studio 2017\Templates\ProjectTempl

User item templates location:
C:\Users\Ron\Documents\Visual Studio 2017\Templates\ItemTemplate

- [x] Always show Error List if build finishes with errors
- [ ] Track Active Item in Solution Explorer
- [x] Show advanced build configurations
- [x] Always show solution
- [x] Save new projects when created
- [ ] Warn user when the project location is not trusted
- [x] Show Output window when build starts
- [ ] Prompt for symbolic renaming when renaming files
- [ ] Lightweight solution load for all solutions

You will notice in the videos that I have line numbers turned on so I can talk about a specific line of code. Many students like the line numbers so to turn them on, choose Text Editor | All Languages | General and click Line numbers on the right pane. There are a ton of settings you can play with here so check them out.


Options

Search Options (Ctrl+E)

- Source Control
- Text Editor
  - General
  - File Extension
  - All Languages
    - General
    - Scroll Bars
    - Tabs
    - CodeLens
  - Basic
  - C#
  - C/C++
  - HTML (Web Forms)
  - JavaScript
  - Plain Text
  - ResJSON Resource
  - SQL Server Tools
  - T-SQL90
  - TypeScript

Statement completion

- ☑ Auto list members
  - ■ Hide advanced members
- ☑ Parameter information

Settings

- ■ Enable virtual space
- ☐ Word wrap
  - ■ Show visual glyphs for word wrap
- ■ Line numbers
- ■ Enable single-click URL navigation
- ■ Navigation bar
- ■ Automatic brace completion
- ☑ Apply Cut or Copy commands to blank lines when there is no selection

Note: This page sets options for all languages. To change options for only one language, select the desired language from the tree on the left.

OK Cancel

Step 5
Add Features to your first project

1. Start video #2 and follow along as you build your first VB app with me. Pause the video to keep pace and be sure to take lots of notes. Remember, this is like my lecture material!
2. Now listen to video 3 & video 4. You will add new features to the project you just created.
3. Now I want you to save your project and create another one on your own. See how much you can build on your own. This will let you know if you understand the concepts.

You will find several other videos in lesson 1 so feel free to examine them. Video 6 shows you how I set up my computer to use VB so you might want to set yours up the same way. You will be assigned some of the other ones later as we go on.

Vocabulary & Concepts you should know at this point

IDE

Control

Property

Name property

Text Property

Source Code

How to rename a control

How to give the form a caption

Design time

Run Time

MSIL (Microsoft Intermediate Language)

Solution Explorer

.Net Framework

Toolbox

Properties Window

Code Editor

How to run a project (F5)

Lesson 2: Getting User Input

- In this lesson you will learn to create applications that get user input from the keyboard.
- You will build projects using textbox controls so the user can enter information.
- You will learn how to customize label controls to display messages on screen.
- And you will use button controls to handle the behavior of your application.

Lesson 2: Getting User Input

Textbook Chapters & Videos for Lesson 2

- Murach Chapters 3 & 4
- Videos on Data Entry 1-4

Data Entry 1

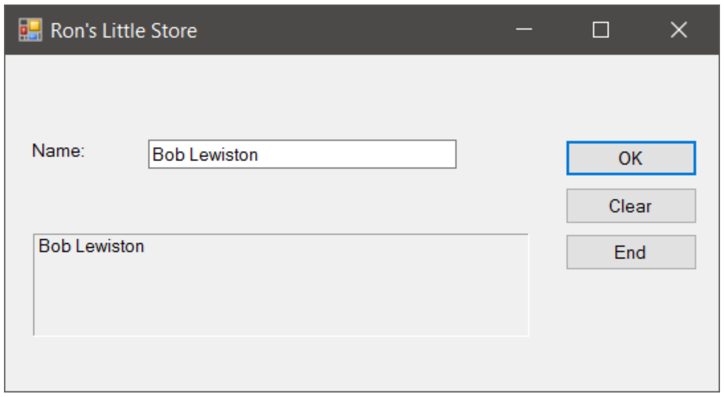
As a programmer, you must remember the three basic processes we use to create applications:

1. INPUT
2. PROCESS
3. OUTPUT

Our phones, tablets, and desktop computers are useless unless data is entered into them via the keyboard, mouse, or even a microphone. After we gather the necessary data, we have to process it. I often refer to this step as "Do the Math". In the projects that follow in the lesson, you will learn to do some simple things like compute sales tax and the balance due when users are placing orders at our fictitious online store.

Finally, we must do something with our computed results and that is where "Output" comes in. For our class, output means displaying the results on the screen or in a message box. We could however, save data to the hard disk or send it via Bluetooth to another device. So when you begin a new project, please keep these steps in mind and that will help you decide where to start coding your app.

Project 1: Data Entry part 1: Using Controls



Name: Bob Lewiston

Bob Lewiston

Now, let's take a closer look at Data Entry Part 1. Start the video for the project from the web page.

Before we build this together, let me show you what it is supposed to do.

Features:

- When the user types in their name and click OK, we take what they type and copy it to a label control. This is the large box with a 3-D border around it. Remember, I use the labels to simulate printing out a receipt to our customer.
- When they click the Clear button, the app will clear the textbox, the label, and place the blinking cursor in the "Name" textbox so they can start over.
- The End button closes our app.

Now, before I give you some tips and explain how to code the project, keep in mind that you, the coder gets to choose how the app behaves when someone clicks on a button or types something into a textbox. Since we are using object-oriented programming, nothing happens to our app until someone clicks on something or types something.

Also, you cannot make changes to your app when it is running. So make sure to click the red stop button on the VS toolbar before editing code.

Now let me walk you through the process of building this app. We are going to approach this in a step-by-step process so you can see how I work. This will save you time and frustration when you are learning. Once you get the hang of it, you will build apps your own way. I am going to detail this process for this project because you are getting started. But after this project, I will expect you to be able to design these user interfaces/forms on your own.

Steps to design Data Entry 1

1. Create a new VB Windows Classic Desktop project in VS. Name it "VB Lesson2 Data Entry 1".
2. When the project opens, double-click Form1.vb in solution explorer to display your main form. Then resize your form a bit so it looks like mine.
3. Now open the toolbox and add the controls you see. If the toolbox covers your form, then click on the little push pin in the top right corner of the toolbox to "pin" it. Now you can drag controls onto your form.
4. In the tool box, expand the "Common Controls" section and drag a label onto the form. This will say "Label1" on screen when you place it. We need it to say "Name" so change its TEXT property to "Name". Press ENTER to lock in the change.

   a. Now add a textbox and change its NAME property to txtName.
   b. Now add your three buttons. The name property of the OK button is btnPrint. Change its TEXT property to "OK".
   c. Rename the Clear button to btnClear and change its TEXT property to "Clear".
   d. Now edit the End button. Its NAME property is "btnQuit" and its TEXT property is "End".
5. Setting up the big label:
   a. Now add another label to represent a receipt. Change its NAME property to lblReceipt.
   b. Delete anything in its TEXT property so it looks empty on the screen.
   c. Change its Autosize property to false so you can change its size to look like mine.
   d. Now change the borderstyle property of the big label to Fixed3D.
6. Now single-click on the blue top border of the form. Change the NAME property of the form to frmMain and change the forms TEXT property to "Ron's Little Store" or something similar.
7. Click on the two diskettes on the VS toolbar to save your changes. **PLEASE GET IN THE HABIT OF SAVING YOUR WORK OFTEN.** I always save everything before testing my changes.

Steps to Code Our App

1. Students always wonder where they should start coding. My suggestion is to make a list of the features you want and start coding each feature. The order we do that in is not really important for this app so I am going to do it this way:
   a. Write code to handle the OK button.
   b. Write code for the Clear button.
   c. Write code for the End button.

Let's do this! Double-click the OK button so you have a place to write your programming instructions. It should look like this.

```
Private Sub btnPrint_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrint.Click

        '---now take what they typed into the textbox and put it in the label

End Sub
```

2. This is called an "Event Handler" or a "Click event Handler". I just call it a click event for short. When they click on this button, the code we write will be executed one line at a time in the order we type it.

So, add a comment like I did so we will know what/why we coded this when we come back to at some later date. Commenting your code is super important so make sure you do it.

3. Now add this line of code. It says take what they typed in the name textbox and copy it into the big label called lblReceipt. If you read it from right-to-left it will help you understand what it is doing. Just like in a math formula, the left side of the = sign is the destination of where the data is supposed to go. The right side is where the data is coming from.

```
Private Sub btnPrint_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrint.Click

        '---now take what they typed into the textbox and put it in the label
        lblReceipt.Text = txtName.Text

End Sub
```

4. Now SAVE your changes and press F5 to run it and see if it works. Whatever you type in the textbox should appear in the big label. I hope it worked. If not, stop your project and fix your errors. Save again and run to see if it worked.
5. Let’s code the Clear button. Make an event handler for it by double-clicking on it.
    a. I want to do three things when they click this button. So let’s clear the name textbox, clear the big label, and place the blinking cursor into the name textbox so they can start over.
    b. Here are the codes to do that:

```
Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear.Click
       '---now clear the textbox and label
       txtName.Clear()
       lblReceipt.Text = ""          'labels do not have a clear method


       '---show cursor back in name textbox so they can start over
       txtName.Focus()

   End Sub
```

So, in this event handler, we used the Clear() method to clear the textboxes. But labels do not have a Clear() method so we do it the old-fashioned way: we delete any text in the TEXT property of the label. We used to do that for textboxes also (and you still can) but now I use the Clear() method all the time. Remember a method is like an ACTION that you, the coder types in. Methods have names like verbs such and Clear, Focus, Close, Open, Trim, etc.

6. Now save all and run again to see if your new features work. Remember to stop your app before making edits.
7. The last step is to allow the user to close the app. Double-click the “End” button so you can make another event handler. Type in Me.Close() in that handler and save all and run again. All of your features should work and we are done! A complete code listing is below. Notice my comments about how to center your form on the screen.

```
'The form starts up in the center of the screen. Set the StartPosition property for the form to CenterScreen

Public Class frmMain

    Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnQuit.Click
        '---now close the form. "ME" refers to the form itself. Close is a method (action) that tells VB what to do. In this
        '   case, close the form and remove it from memory.
        Me.Close()


    End Sub

    Private Sub btnPrint_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrint.Click

        '---now take what they typed into the textbox and put it in the label

        lblReceipt.Text = txtName.Text

    End Sub

    Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear.Click
        '---now clear the textbox and label
        txtName.Clear()
        lblReceipt.Text = ""          'labels do not have a clear method


        '---show cursor back in name textbox so they can start over
        txtName.Focus()


    End Sub
End Class
```

Notice how the event handlers are in the order we created them. I do this because I want the “quit” code to be at the bottom and the data input to be at the top. It makes more sense to me and I can go back and find what I’m looking for easier. On a big project, this will really help you keep organized.

Terms and Concepts To Know From This Project

Property

Method

Event

How to clear a textbox vs a label

How to rename controls and change what they display on screen (Instead of Button1 it says Clear, for example).

Learn the basic commands such as: Focus (), Clear (), .Text, Me.Close ()

Form properties:

- How to change the title or caption
- How to center the form on the screen

How to save your changes

Input, Process, Output as it relates to this app.

Lesson 2: Data Entry Part 2

In this lesson you will learn to manipulate text that users input into your apps.

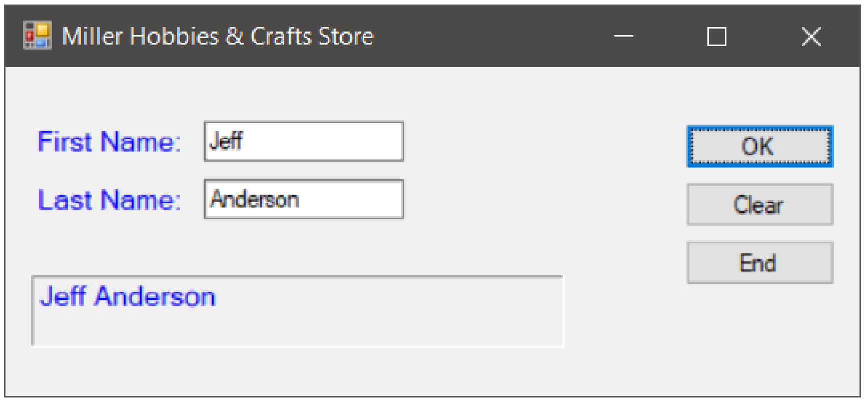
Data Entry 2

As a programmer, you must learn how to work with text. Programmers refer to text data as *String* data. In this project you will learn how to *concatenate* the first name and last name that they type in so you can display their full name. Please always remember that when you work with data, you must tell the computer what type of data you are sending it. If you enter two numbers so the app can add those numbers, you have to tell VB what type of numbers you are inputting. In other words, we have to tell it we are inputting integers, decimals, or something else.

In this project, we are going to learn how to tell VB we will be inputting string data.

Project 2:

Data Entry part 2: Working with strings



Here is what the finished app looks like.

Notice that it is like project 1 that we just completed. As you can see, when someone enters their name and click OK, I code VB to take whatever is in the First Name textbox, add a space, and then append the last name to make one long string that will hold the full name. Finally, that completed string is copied to the large label that we used before. When we take string data (text) and manipulate it like this, we say we concatenate the string. It just means we are adding/deleting some characters to make another string.

The Clear and End buttons work as you would expect except the clear button has to be coded to clear both textboxes and the label this time.

Steps to design Data Entry 2

1. Create a new VB Windows Classic Desktop project in VS. Name it "VB Lesson2 Data Entry 2".

2. Open and follow along with the video for this project here. This video shows several tips so please watch it. You can fast forward through the stuff you already know.
3. When the project opens, double-click Form1.vb in solution explorer to display your main form. Then resize your form a bit so it looks like mine.
4. Now open the toolbox and add the controls you see. Remember, if the toolbox covers your form, then click on the little push pin in the top right corner of the toolbox to "pin" it. Now you can drag controls onto your form.
5. In the tool box, expand the "Common Controls" section and drag two labels and two text boxes to your form and align them like my example. In this example I changed the ForeColor property to make them blue and changed the Font Size property to 10 so they look bigger. Be sure to do this so you get used to working with a variety of properties.
6. Change the text properties of each label to match mine.
    a. Now change the name of the first textbox to txtFirstName and the second textbox to txtLastName.
    b. Now add your three buttons. The name property of the OK button is btnPrint. Change its TEXT property to "OK".
    c. Rename the Clear button to btnClear and change its TEXT property to "Clear".
    d. Now do the End button. Its NAME property is "btnEnd" and its TEXT property is "End".
    e. The large label needs to be set up like you did with Data Entry 1. Be sure to set its autosize property to false and make it wider like I did. Change its name to lblReceipt and its borderstyle to Fixed 3D.
7. Now set up your form properties. Make it startup in the center of the screen (StartPosition Property) and change the Caption to say something other than Form1 by changing its TEXT property.
8. Now SAVEALL then run the app to make sure it displays properly on screen. If everything is working, stop it so we can code it.

Steps to Code Our App

1. Now, make an event handler for each button starting with the OK button. Remember the easiest way is to double-click on each button. This will bring up your code behind window where you can program the controls.

Your btnOK event handler should look like mine:

```
Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnOK.Click

    '---now show first and last names with a space in between
    '   When you connect strings together like this, use the '&' sign not a '+' sign.
    '   This process is called concatenation!!!!

    lblReceipt.Text = txtFirstName.Text & " " & txtLastName.Text

End Sub
```

2. Notice how I concatenate the first and last name. I tell VB that the result is going into the large label. Try to remember to add the .Text bit or it won't work. I am setting the text property in the lblReceipt control to display the first name, then a space, then the last name. This formula you see is not a math formula. It is called an assignment statement. In VB, the "=" sign can do math OR assign a value to a control like we do here.
3. So, the first thing to do type lblReceipt = txtFirstName.Text. The next part adds a space after their first name. In VB, I type in the ampersand (&) and then two double quotes with a space between them " " like this.

But wait, I also want it to append or add the last name to this string so I type another & and then the name of the control where the data is coming from which is the txtLastName textbox. And yes, add the .Text here also.

If you read this statement backwards it will make more sense. It says, grab the data in the first name textbox, add a blank space, and add the data from the last name textbox and put the whole thing into the lblReceipt label so we can see it on screen.

Many students get confused when writing assignment statements. The destination of your data comes first before the "=" sign and the actual data itself comes after the equal sign.

4. Make an event handler for the Clear button.
    a. Write code to clear the first name textbox.
    b. Write code to clear the last name textbox.
    c. Now clear the lblReceipt control (remember no Clear method for labels.)
    d. Now set the cursor to the first name textbox
5. Make an event handler for the End button and write code to close your app.
6. SAVE ALL then press F5 to run it and test it.
7. If you have errors, stop the project and fix them then save again and run again until you are happy with the results.

Summary

In this simple project you got more practice with building a Windows desktop app in VB. You learned how to provide a way to let someone enter in two pieces of data and then you learned how to concatenate those pieces of data to make a new string and display it on screen.

Every time you build a new VB app you will get faster at it as you learn your way around VS and also VB.

One last thing, make sure you comment your code and put a title and date and your name at the top of your code like I do in the videos. If you send me home work without comments and good documentation, I will ding you some points!

Terms and Concepts To Know From Data Entry 2

BorderSize Property

ForeColor Property

Font Property

Concatenation

What the & character does

How to add a blank space to a string

What is the difference between and assignment statement and a math statement?

Lesson 2: Data Entry Part 3

In this lesson you will learn to use simple math coding to compute totals, sales tax and work with pricing of items.

Data Entry 3

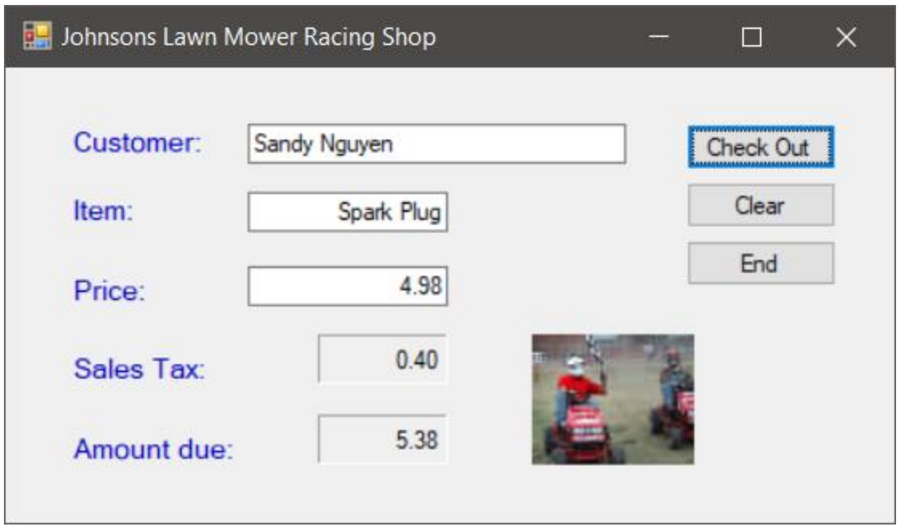
As a programmer, you must learn how to work with all types of data. In this project we will look at how we deal with money. The app represents a simple online store app you are used to seeing on the web. Here the user enters in their name, the item they want, and the price. When they click the checkout button, the sales tax and the total due is computed for them. The little image you see is inside a picture box and I just added it for fun. You won't add this to our project. I just wanted you to see that you can!

As you can see, the major things to learn are going to be in the checkout button's event handler. I will show you how to make sure VB knows we need to do math this time. I also show you how to format your results with two decimal places so it looks nice.

Project 3: Data Entry part 3: Working with simple math

Here is what the finished app looks like.

Notice that it is more complex than the first two projects because now you will learn how to process data.


Johnsons Lawn Mower Racing Shop

Customer: Sandy Nguyen
Item: Spark Plug
Price: 4.98
Sales Tax: 0.40
Amount due: 5.38

The Clear and End buttons work as you would expect except the clear button has to be coded to clear more controls than before.

Steps to design Data Entry 2

1. Create a new VB Windows Classic Desktop project in VS. Name it "VB Lesson2 Data Entry 3".
2. Start the video for this project now and follow along. There are several new concepts in this project so make sure you take good notes.

3. Add the controls as I do. Be sure to use the TextAlign property on the textboxes and the price, sales tax, and amount due boxes so the money will line up right and look professional.
4. Notice the sales tax and amount due boxes are labels with the borderstyle set to fixed/3D like we have been doing with our receipt labels. Why do I use a label here and not a textbox? I don't want them to edit those values. You will quickly learn that people type all kinds of junk into your apps and it is our responsibility to make sure the data is correct before processing it.
5. Use controls that minimize the risk of people putting data in the wrong place. Also, when you run this, if someone types in letters instead of a real price, the app will crash. That just means VB won't understand how to do math on letters so it will throw an error. Errors in VS are called exceptions.
6. One thing you will learn in this video is how to do casting. Casting means you are telling VB to change the data from one type to another type. In our case, everything types into your app from the keyboard is always treated as String data. So, when someone enters in 5.95 VB treats it as string… not currency. I show you how to convert the price they enter into a decimal number so we can do math.

I mention these issues because you have not learned how to validate data yet. But you will. Now, build your app, save all, run it and test it for bugs (errors). Make sure you understand how I compute sales tax and how you do it in VB.

Terms and Concepts To Know From Data Entry 3

Casting

FormatNumber vs FormatCurrency

CDec function

How to compute sales tax and totals

TextAlign property on textboxes and labels

Data entry from the keyboard is always treated as string by VS.

Here is the complete code for this project. Make sure you understand everything I do here because our projects will become more involved very soon.

```
Private Sub btnCheckout_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCheckout.Click
        '---here is where we are going to calculate the tax and amt. due
        '   We will also format the money correctly so it look like a cash register.

        'FormatNumber is a built-in VB Function
        lblTax.Text = FormatNumber(txtPrice.Text * 0.08)        'this makes it use 2 decimal places

        lblBalDue.Text = CDec(txtPrice.Text) + CDec(lblTax.Text)  'CDec makes VB treat text as
                                                                  currency. CDec=convert to decimal

    End Sub


Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
        '---clear everything and set cursor to name textbox
        txtCustomer.Clear()
        txtItem.Clear()
        txtPrice.Clear()
        lblTax.Text = ""
        lblBalDue.Text = ""
        txtCustomer.Focus()
End Sub


Private Sub btnEnd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEnd.Click
        'end this beauty!
        Me.Close()

End Sub
```

Lesson 2: Data Entry Part 4

In this lesson you will learn to use List Boxes. You will fill the list box with items to sell and when the user picks one it is “added to their cart”. Again, we are not validating data so if you don’t enter a price, it will crash!

Data Entry 4

In this project we will look at how we deal with letting a user select an item from our store using a list box control. You use them all the time but now you will learn how to build you own. There are several ways to add items to a list box but here we will learn how to add items using a little text editor inside VS. You can use the Sorted property of the listbox to alphabetize the items.

This app lets the user select a product and enter a price. When they click the OK button, the sales tax and the total due is computed for them as we have already learned how to do. The image you see is inside a picture box and I just added it to make it look less boring!

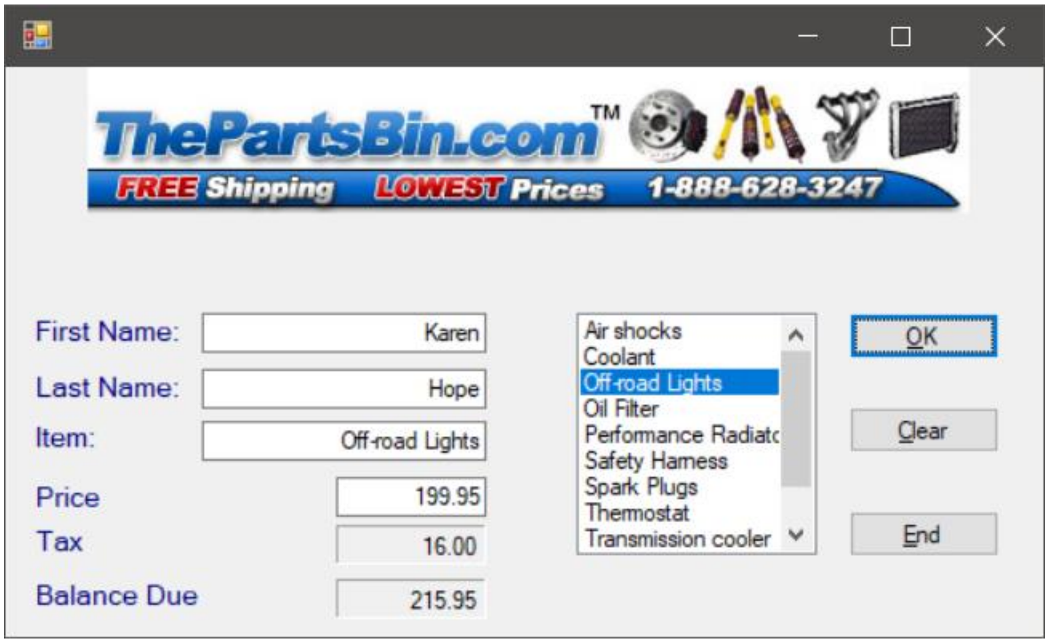
As you can see, the new things to learn are going to be in the event handler for the listbox. The code inside the checkout button's event handler should look very familiar to you by now.

Project 4:
Data Entry part 4: Working with List Boxes

Here is what the finished app looks like.

Notice that it is more complex than the first three projects because now you will learn how to process data and use a listbox control to let them select a product.

.



First Name: Karen
Last Name: Hope
Item: Off-road Lights
Price 199.95
Tax 16.00
Balance Due 215.95

Air shocks
Coolant
Off-road Lights
Oil Filter
Performance Radiatc
Safety Harness
Spark Plugs
Thermostat
Transmission cooler

Steps to design Data Entry 4

1. Create a new VB Windows Classic Desktop project in VS. Name it "VB Lesson2 Data Entry 4".
2. Start the video for this project now and follow along. The main things you will learn include:
    a. How to add items to a listbox at design time.
    b. How to create the event handler for Mr. Listbox.
    c. How to grab the item selected from the Listbox and copy it to a textbox.
    d. How to compute tax and totals as before and cast to the correct data type (good practice)
    e. How to set the tab order of the controls.
    f. How to create short-cut keys for your buttons.
3. Add the controls as I do. Be sure to name them as I do in the video!
4. When you add the listbox, it will be setup to only allow the user to select one item. Later in the course you will learn how to let them select multiple items.
5. You can set the Sorted property of the listbox to true to sort the items.
6. Items in a listbox are organized using index numbers. These numbers start from zero! So, the first item in the box is item 0 and the next is item 1, and so forth. VB doesn't care what your items are called. When someone clicks on an item, it uses the index number to know which one was clicked.
7. The event handler that is created when you double-click the listbox is called **SelectedIndexChanged**. Any time someone clicks on an item this event handler is triggered so be aware of that.
8. Be sure to use the TextAlign property on the textboxes and the price, sales tax, and amount due boxes so the money will line up right and look professional.
9. Notice the sales tax and amount due boxes are labels with the borderstyle set to fixed/3D like we have been doing with our receipt labels.

Also remember we are not validating data so what will happen if you put in the wrong characters for price?

Here is the complete code for Data Entry 4:

```
Public Class frmMain


   Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnOK.Click
        '---here is where we are going to calculate the tax and amt. due
        '   We will also format the money correctly so it look like a cash register.


        lblTax.Text = FormatNumber(txtPrice.Text * 0.08)        'this makes it use 2 decimal
places

        lblBalDue.Text = CDec(txtPrice.Text) + CDec(lblTax.Text)  'CDec=convert to decimal

   End Sub


   Private Sub lstProducts_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles lstProducts.SelectedIndexChanged
        '---when they click on an item, this code runs(executes) and takes the item chosen
        '   and puts it into the text property of the txtItem textbox.
        txtItem.Text = lstProducts.SelectedItem
   End Sub



   Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
        '---clear everything and set cursor to name textbox
        txtFirstName.Clear()
        txtLastName.Clear()
        txtItem.Clear()
        txtPrice.Clear()
        lblTax.Text = ""
        lblBalDue.Text = ""
        txtFirstName.Focus()
   End Sub



Private Sub btnEnd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnEnd.Click
        Me.Close()
   End Sub

End Class
```

Terms and Concepts To Know From Data Entry 4

Know how to add items to a listbox at design time.

Know how to set tab order

Know how to create short-cut keys

What is the default event handler for a listbox called?

How are items in a listbox managed by VB?

How do you sort listbox items?

Be sure you understand why I had to cast the data for price.

Lesson 3: Working with Variables and Dynamic Properties

In this lesson you will learn to use variables to manage your data. There is more on using List Boxes, and you will learn how to keep a running total and running count as your customers buy stuff at your store.

Lesson 3: Introduction to Variables, Counts & Totals and Other Controls

Textbook Chapters & Videos for Lesson 3

- Murach Chapter 4
- Videos 7-10 on your VB page under Lesson 1 (Video Series Part 2)

Learning About Variables

What to do now:

- I want you to download, unzip, and open this Introduction To Variables project and look at it while viewing the next video. There are a lot of comments in the project so read them.

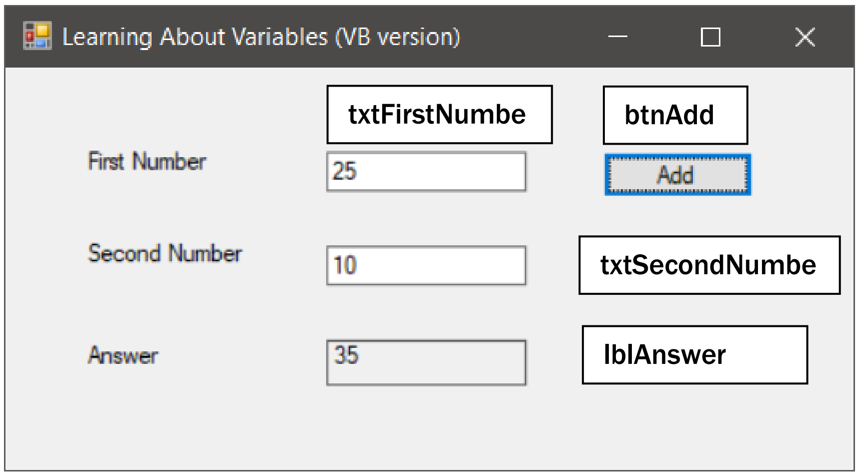
- This video is from my C# class but everything I explain applies to VB also. Take Good notes!

- Please look at these Slides about **VB Data types** while listening to the part near the end of the video where I discuss data types.

Now watch this video on Learning How to Use Variable. **Pay attention to what I am teaching you. Don't get distracted by the C# code (although it will be obvious to you how it works at this point).** I talk about how important it is to tell VB what data you are using. I talk about the different types of errors we get when coding in any language. Take notes!!!

You should create this project AFTER watching the video so you can get some good practice. Use the codes below. I show you what I named the controls in this screen shot (txtFirstNumber, btnAdd, etc.).

Add your controls and make the event handler for the button.



In VB we use different codes to create our variables than C# does. Look at the code below as you watch the video. You need to learn the VB code to work with variables.

Notice in the video I use four steps to accomplish this in the btnAdd handler.

```
Private Sub btnAdd_Click(sender As Object, e As EventArgs) Handles btnAdd.Click
        '---WRONG Answer   Concatenating instead of doing math
        'lblAnswer.Text = txtFirstNumber.Text + txtSecondNumber.Text

        '---with variables
        ' Step 1: Declare local variables And initialize them()
        Dim firstNumber As Integer = 0  'this holds num 1 in RAM
        Dim secondNumber As Integer = 0  'hold 2nd number in RAM
        Dim result As Integer = 0              'hold result

        'Step 2: Assign values To variables

        firstNumber = Convert.ToInt32(txtFirstNumber.Text)
        secondNumber = Convert.ToInt32(txtSecondNumber.Text)

        'Step 3 Do the Math

        result = firstNumber + secondNumber

        'Step 4: Display results
        lblAnswer.Text = result.ToString()


    End Sub
```

How this code works

In step 1 we always create our variables before using them…ALWAYS! I also initialize them myself even though VB is supposed to do it for us. You do it like I am showing you. NEVER put spaces in your variable names.

Notice in VB we use the keyword Dim. Dim stands for dimension which reserves enough space in RAM for the type of data we are using.

We created our variables in the event handler and that means these are LOCAL variables. That means we can only access them inside this event handler!

In Step 2 we assign values to our variables. We take values from a textbox and store them in our newly created variables. And notice, I cast those values from the textboxes because otherwise they will be treated as string data. Integers require 32 bits of memory (4 bytes). In the video I show you how to use the intellisense to cast to the correct size of integer. Write this down.

Recall that casting is what we call converting one data type to another. If we do not tell the CPU exactly what to do it takes its best guess. And that is usually wrong. I showed you what happens when you want the computer to add two numbers without explicitly telling it the numbers are integers. I always tell students that when you type in data from the keyboard, the computer treats the data as string (text) and therefore won't do math correctly. MAKE SURE YOU UNDERSTAND WHY LINE 30 IN THE VIDEO FAILS!

In step 3 we do the math. Notice how clean this is when you use variables and not have to deal with the textboxes any more.

In Step 4 we display the results. But wait! Now I have to cast/convert my result (integer in this case) back to a string because that is what textboxes and labels love. To do it, use the new .ToString() method.

Code Listing for Introduction to Variables

```
'____________________________________________

'L E A R N I N G   A B O U T   V A R I A B L E S
'                  VB Spring 2016
'                    Ron Kessler
'           Visual Studio 2015 Version
'____________________________________________
'
'This project shows how to use variables to do simple math.
'I want you to watch the video I made on this. It is in C# which
'is very similar to VB. Comments are different and in C# we put
'a ; at the end of the lines. However, everything I say in that video
'applies to VB. At the end of the video when I show the C# data types,
'open up the VB data types using the link next to this project and view that. You
'will see they are very similar. VB just uses different words for the same thing.

Public Class Form1
   Private Sub btnAdd_Click(sender As Object, e As EventArgs) Handles
btnAdd.Click
       '---WRONG Answer   Concatenating instead of doing math
       'lblAnswer.Text = txtFirstNumber.Text + txtSecondNumber.Text

       '---with variables
       ' Step 1: Declare local variables And initialize them()
       Dim firstNumber As Int32 = 0      'you can use Integer instead of Int32 if you
want
       Dim secondNumber As Int32 = 0
       Dim result As Integer = 0             'see, I can use Integer in VB

       'Step 2: Assign values To variables

       firstNumber = Convert.ToInt32(txtFirstNumber.Text)
       secondNumber = Convert.ToInt32(txtSecondNumber.Text)

       'Step 3 Do the Math

       result = firstNumber + secondNumber

       'Step 4: Display results
       lblAnswer.Text = result.ToString()

   End Sub

End Class
```

Terms and Concepts to Know from This project

Know how to create variables.

Know how to initialize variables.

How many bits does an integer use?

What is a bit vs a byte?

How many bits in a byte?

Know what Int32 means.

Know how to cast a string to an integer.

Know how to cast integer back to a string.

Make sure you understand concatenation.

Understand why this code fails:
lblAnswer.Text = txtFirstNumber.Text + txtSecondNumber.Text

Local scoped variables.

Design time, run time, & logic errors.

Lesson 3: List Boxes & Dynamic Properties

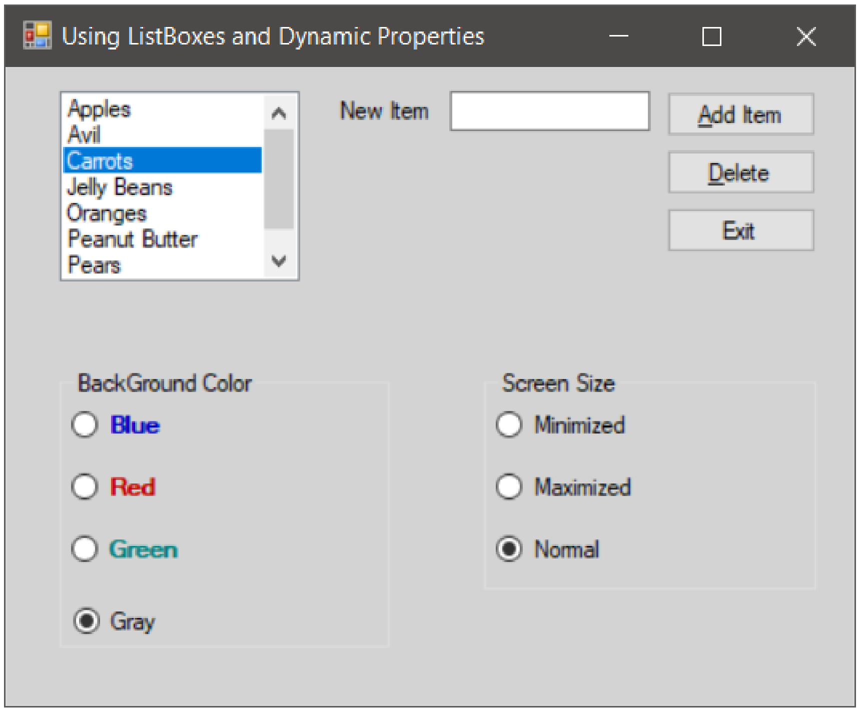
- By now, you have had a chance to make several VB projects and should be feeling more comfortable with VB and VS. In this lesson, I am going to show you many new things to make your projects have more capabilities. We will cover quite a bit of new material so go slowly and make sure you understand each part before jumping to the next section
- The first thing you will learn in this lesson is how to use more features of the List Box. For example, you will learn how to add and delete items in the list box while the program is running.
- You will also learn how to change properties at run time. This means your user can change background colors and form properties while they are using the program.
- You will also learn how to work with Radio Buttons and Check Box controls. I will show you the event handlers for these controls and how to use them.
- You will also learn about .Net namespaces.
- Finally, you will see how I track down coding errors when the program will not compile.

List Boxes & Dynamic Properties

What to do now:

- I want you to download, unzip, and open the project for this section.
- Then listen to the video as you look at the code in the project. The project in the video looks like the one below:

You can see the two groups of radio buttons on this form. Radio Buttons are designed to work in a group so that only one of them can be selected at a time. In the video I show you how to have two or more groups of them on a form by placing them inside Group Boxes.



Group Boxes

The group box control is on the VS toolbox under the "Containers" section and you just drag it to the form. Then you put your buttons inside it.

The text property of group boxes allows you to identify them. You can see I named mine "Back Ground Color" and "Screen Size".

Working with Colors & Namespaces

In order to change the background color of a form, you need to tell VS to include some features for you. This is where I show you how to add features/capabilities from the .Net framework. The .Net framework is what Microsoft calls the massive number of files and folders that were added to your hard disk when you install Windows and Visual Studio.

Everything that we build in VB, C#, and C++ are located on your computer. But we only add the features we need to our projects. Otherwise, our projects would be too huge! The features are contained in something called dynamic link libraries (DLL's) and are arranged in files and folders they call namespaces. To add a namespace to a VB project we use an "Imports" statement. I show you all this in the video.

It is very important that you listen to the video carefully and take plenty of notes. This project looks rather simple. However, I am using it to show you more complicated features and capabilities that you will need to know as a professional coder.

Terms and Concepts to Know from This project

Know how to work with VB constants.

Know how to add Namespaces to your project.

What are the default event handlers for Radio Buttons & Check Boxes?

Make sure you know how to add/delete items in a list box.

How do you sort a list box?

Know how to fill a list box at design time and how to add an item from a textbox.

What does the VB keyword "Me" refer to?

What is a "collection" in .Net?

How do you move event handlers around for better organization to make your code easier to read/follow?

Lesson 3: Totals & Counts

Every programmer needs to know how to handle data. In this section I show you how to keep a RUNNING COUNT of the items someone buys from our store. I also show you how to keep a running TOTAL so we know how much they owe us at checkout.

By learning these basic programming concepts, you will be able to add much more functionality to your apps than we have done so far.

You will learn how to create variables in a new place. So far we have used local variables inside event handlers like the button_click() event. Now you will discover that you can define variables in a different section of your code so you can do more powerful and useful things with your data.

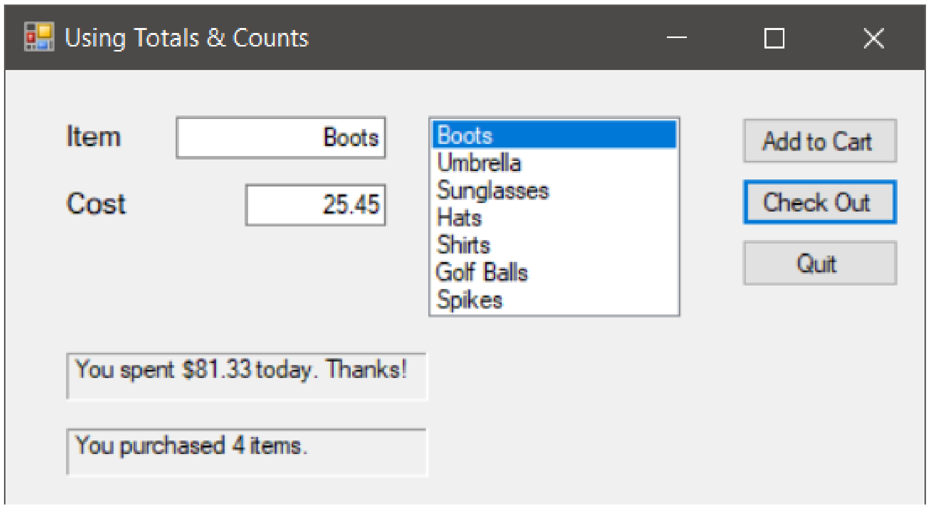
To make this demo work more naturally, I put the prices for each item in the text box so when they choose an item, the item is copied to the Item textbox and the price is copied to the Cost textbox. This is a chance to show you how to work with and manipulate string data. You will learn how to “split” a string so they app will put the data where I want it.

During this lesson, you will learn how to debug your project when you get logic errors. This is a super valuable skill you must learn.

Working with Totals & Counts, String Manipulation, & Debugging

What to do now:

- I want you to download, unzip, and open the Totals & Counts demo project which is on the website in lesson 3.
- Then listen to Part 1 video for this project. The project is similar to the one shown below.
- When the first video ends abruptly, close it and listen to Part 2. I finish the stuff I want to teach you in part 2.



You spent $81.33 today. Thanks!

You purchased 4 items.

- Please just watch the video and pause it when you need to write down some notes.
- I show you a ton of new features:
  - How to add items & pricing to a listbox at design time.
  - How to use the Decimal data type.
  - How to use the Short integer data type.
  - How to use shorthand operators like +=, -=, etc.
  - How to use the VB conversion functions like CDec, CInt, etc.
  - How to split item and price from a string and clear spaces.
  - How to use the Mid function.
  - How to use the .ToString or the CStr methods.
  - How to keep a running total in memory.
  - **How to deal with logic errors!**
  - How to set break points.

Code Listing for Counts & Totals Demo

Make sure you understand the difference between local and form-level variables as you examine this code.

```
'-----------------------------------------------------------------------------------------

'                         U S I N G   T O T A L S   &   C O U N T S

'                                      Ron Kessler
'
'-----------------------------------------------------------------------------------------

'Notes:

'   This program introduces you to VB counts & Totals. The program keeps a
running total of the ‵amount  they spend and the total number of items they
bought.

'   The program also shows you how to use module-level variables

'Updated 4/11/2016
Option Explicit On

Public Class frmCounts
    'to store money, use Decimal data types
    'to store whole numbers, use Short Integer (Short), Integer, Long Integer(Long)

    '---module level variables
    Dim grandTotal As Decimal = 0     'hold the running total
    Dim numItems As Short = 0       'hold the number of items bought
```

Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAdd.Click

```
        '---take item and cost & put them in textbox
        txtItem.Text = Trim(Mid(lstItems.SelectedItem, 1, 15))
        txtCost.Text = FormatNumber(Mid(lstItems.SelectedItem, 16).Trim)
        '   txtCost.Text = FormatNumber(txtCost.Text)

        '---add this cost to the running total
        'grandTotal = grandTotal + CDec(txtCost.Text)
        grandTotal += CDec(txtCost.Text)
        numItems += 1     'same as numItems = numItems + 1
    End Sub
```

```
 Private Sub btnCheckOut_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCheckOut.Click

        lblTotalSpent.Text = "You spent " & FormatCurrency(grandTotal) & " today.”

        lblNumItems.Text = "You purchased " & numItems & " items."

    End Sub

    Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnQuit.Click

        Me.Close()

    End Sub
End Class
```

How the Code Works- A Closer Look

As I explain in the videos, it is important for you to understand that WHERE you declare variables has a huge effect on how you can use them. We need two variables that will not get reset because we need to add values to them.

To create a variable that DOES NOT get reset/re-initialized, define it outside of any event handler. Notice below the grandTotal and numItems variables are inside the Class definition where the Form code begins but they are OUTSIDE of any Click event. This allows us to change their values and access them from **anywhere** inside this form. We can access them from any event handler in this form. That is why they are called Form-Level, Class-Level, or Module-Level variables. They have Form-Level scope.

How to Create Form-Level Variables

Look at this code snippet from the project.

```
Public Class frmCounts                      ‘The start of the Form

    '---module level variables
    Dim grandTotal As Decimal = 0     'hold the running total
    Dim numItems As Short = 0       'hold the number of items bought
```

Here, I declare variables right under the section where the Form is defined (Visual Studio put this *Public Class frmCounts* part in for us). If I were to create the variables inside the btnAdd_Click event handler, they would be re-defined each time the button is clicked. The code below says:

1. Create room in RAM to store a Decimal number.
2. Name it grandTotal for me.
3. Set its initial value to 0.

Dim grandTotal As Decimal = 0

If this code is placed inside the btnAdd_Click event handler, the computer will create it all over again each time they press the button. So, that is why there is no way it can keep a running total or count. I gets set to zero each time we try to add something to it. Remember that this is a LOGIC error. The program runs but you get the wrong answer. These are the hardest errors to find by the way.

Using Short Cut Operators

Now, let's review the ways we can manipulate data in a variable. We can use short-cut operators to make it easier to add, subtract, multiply, & divide values in an existing variable.

To add values to a runningTotal I take the new string value from txtCost.Text, cast it to a Decimal type (so we can do math), and add it to the existing value in my variable. **Remember, Totals go up or down by a variable amount because the prices are all different!**

Look at this code from the project. I show two ways of doing the same thing:

'---add this cost to the running total using one of these techniques.
The new grand total = old grand total + the new price

```
grandTotal = grandTotal + CDec(txtCost.Text)        Old way

grandTotal += CDec(txtCost.Text)                    New way
```

Now recall how we handle a running COUNT.

Please remember that Counts increase or decrease by a fixed amount. You choose what amount you want…count by 2’s, 10’s, whatever. In this case we increment by 1 because we are keeping track of the number of items purchased. There are two ways to do this as you can see. Use the one that makes the most sense to you.

```
numItems += 1     'same as numItems = numItems + 1
```

Terms and Concepts to Know from This Section

- How to add items & pricing to a listbox at design time.
- How to use the Decimal data type.
- How to use the Short integer data type.
- How to use shorthand operators like +=, -=, etc.
- How to use the VB conversion functions like CDec, CInt, etc.
- How to split item and price from a string and clear spaces.
- How to use the Mid function.
- How to use the .ToString or the CStr methods.
- How to keep a running total in memory.
- **How to deal with logic errors!**
- How to set break points.
- Understand Format vs FormatCurrency, vs FormatNumber.
- How/where to define Class-Level variables.
- What is the main difference between a Count and Total?

Lesson 3: Ron's Pet Store

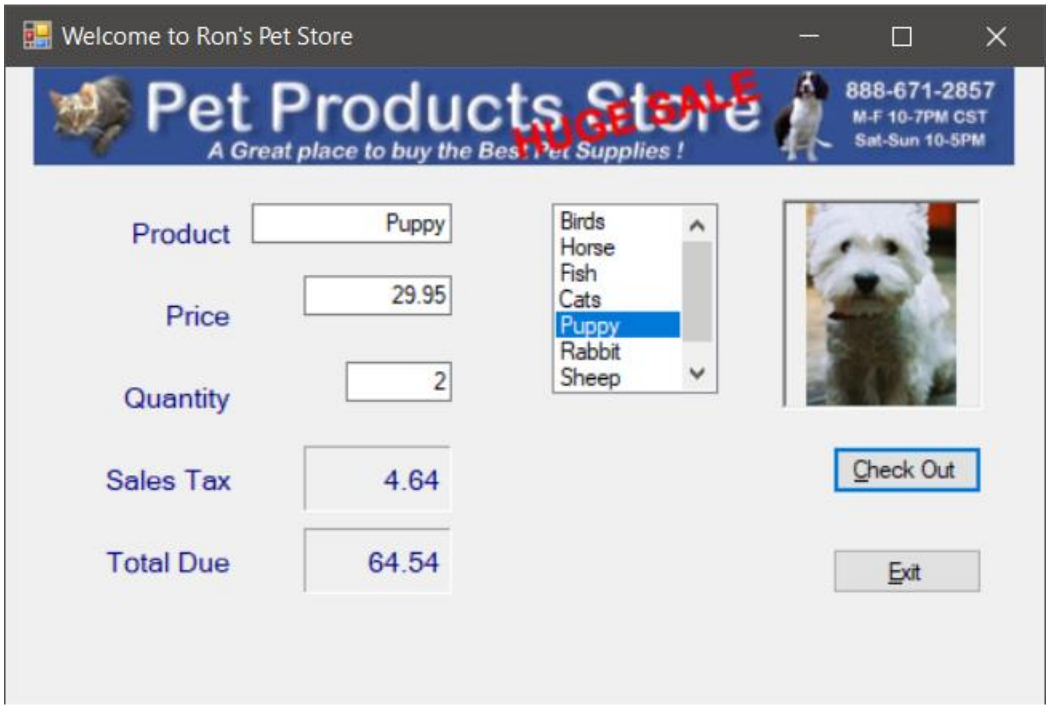
This part of Lesson 3 shows how to do a few more ways to create your shopping cart programs.

- I show how to allow the quantity to be entered by the user.
- I show how to name your Forms.
- How to use a Picture Box to display a banner on screen and to show thumbnails of the items they pick in the list box.
- **I show how to add items and pricing to a listbox in CODE.**
- I show how to access files in different folders inside your solution. This is a really important skill!

Working with Ron's Pet Store Demo

What to do now:

- I want you to download, unzip, and open this Pet Store Demo so you can follow along with me. If you want the images of the animals I use get them from here.
- Now listen to the video for this project. The project is shown below.



How it Works

- When they select a listbox item, we parse the item description, price, and image file from the data in the listbox. Notice also that now we can change the quantity in case they want more than one puppy.
- The code for this part is in the SelectedIndexChanged() event. When they click **Check Out** we do the math.
- Notice also I created keyboard short-cut keys for the buttons. When they press ALT-C it triggers the click event for the button as if I clicked it with the mouse. To do this, go to the text property of a button. Mine is Check Out. To make a letter the short-cut put a "&' in front of the

letter like this : &Check Out. I know it looks funny but it works. Notice the '&' does not show up on the screen. But let's say you wanted an ampersand to show up on a button. Well, all you do is make two of them like "Down && Out". Now one will show up in the text of the button.

A Closer Look At the Code

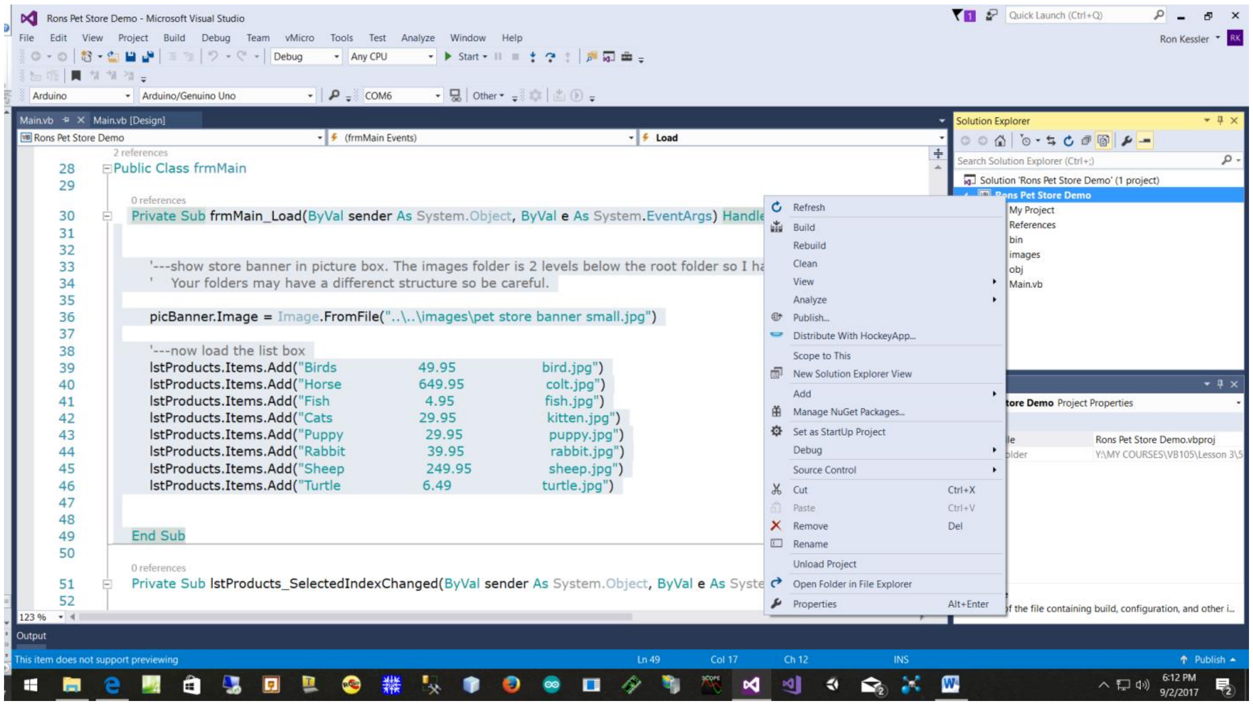
First, I want to add items to my Listbox as soon as the program is visible on screen. To do that, I double-click on the form to create what is called the Form Load event handler (I learned it as the Form Load event so either is fine for me). Please understand that ANY code you place in this section will execute JUST BEFORE THE FORM IS VISIBLE.

Let's check it out…

```
Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'---show store banner in picture box. The images folder is 2 levels below the root folder so I
have to use "..\" twice!
'   Your folders may have a differenct structure so be careful.

      picBanner.Image = Image.FromFile("..\..\images\pet store banner small.jpg")

      '---now load the list box
      lstProducts.Items.Add("Birds          49.95          bird.jpg")
      lstProducts.Items.Add("Horse          649.95          colt.jpg")
      lstProducts.Items.Add("Fish            4.95          fish.jpg")
      lstProducts.Items.Add("Cats           29.95           kitten.jpg")
      lstProducts.Items.Add("Puppy           29.95           puppy.jpg")
      lstProducts.Items.Add("Rabbit          39.95           rabbit.jpg")
      lstProducts.Items.Add("Sheep           249.95          sheep.jpg")
      lstProducts.Items.Add("Turtle          6.49           turtle.jpg")

   End Sub
```

First, notice I tell VS where to find the folder where my images are. Here is what I do when I add a music/photo/images folder to my project.

1. In solution explorer, right-click on the name of your PROJECT. (Rons Pet Store Demo). By the way DO NOT USE APOSTROPHE (') OR OTHER CHARACTERS IN THE NAME OF YOUR PROJECT.
2. Choose New Folder.
3. Name it and click OK. It should show up in solution explorer which means it is part of your project when you zip it up and give it to someone.
4. To add items, I open the project in File Explorer. So right-click on your project again (in solution explorer) and choose "Open folder in File Explorer". Now you can edit it in Notepad and add the data you want. In my case, I typed in the names of the image files I processed in Photoshop.



```
2 references
Public Class frmMain

    0 references
    Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handle


        '---show store banner in picture box. The images folder is 2 levels below the root folder so I ha
        '   Your folders may have a differenct structure so be careful.

        picBanner.Image = Image.FromFile("..\..\images\pet store banner small.jpg")

        '---now load the list box
        lstProducts.Items.Add("Birds            49.95           bird.jpg")
        lstProducts.Items.Add("Horse            649.95           colt.jpg")
        lstProducts.Items.Add("Fish              4.95            fish.jpg")
        lstProducts.Items.Add("Cats             29.95            kitten.jpg")
        lstProducts.Items.Add("Puppy             29.95            puppy.jpg")
        lstProducts.Items.Add("Rabbit            39.95            rabbit.jpg")
        lstProducts.Items.Add("Sheep            249.95            sheep.jpg")
        lstProducts.Items.Add("Turtle            6.49            turtle.jpg")


    End Sub

    0 references
    Private Sub lstProducts_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As Syste
```

How to Change Folder Levels in Code

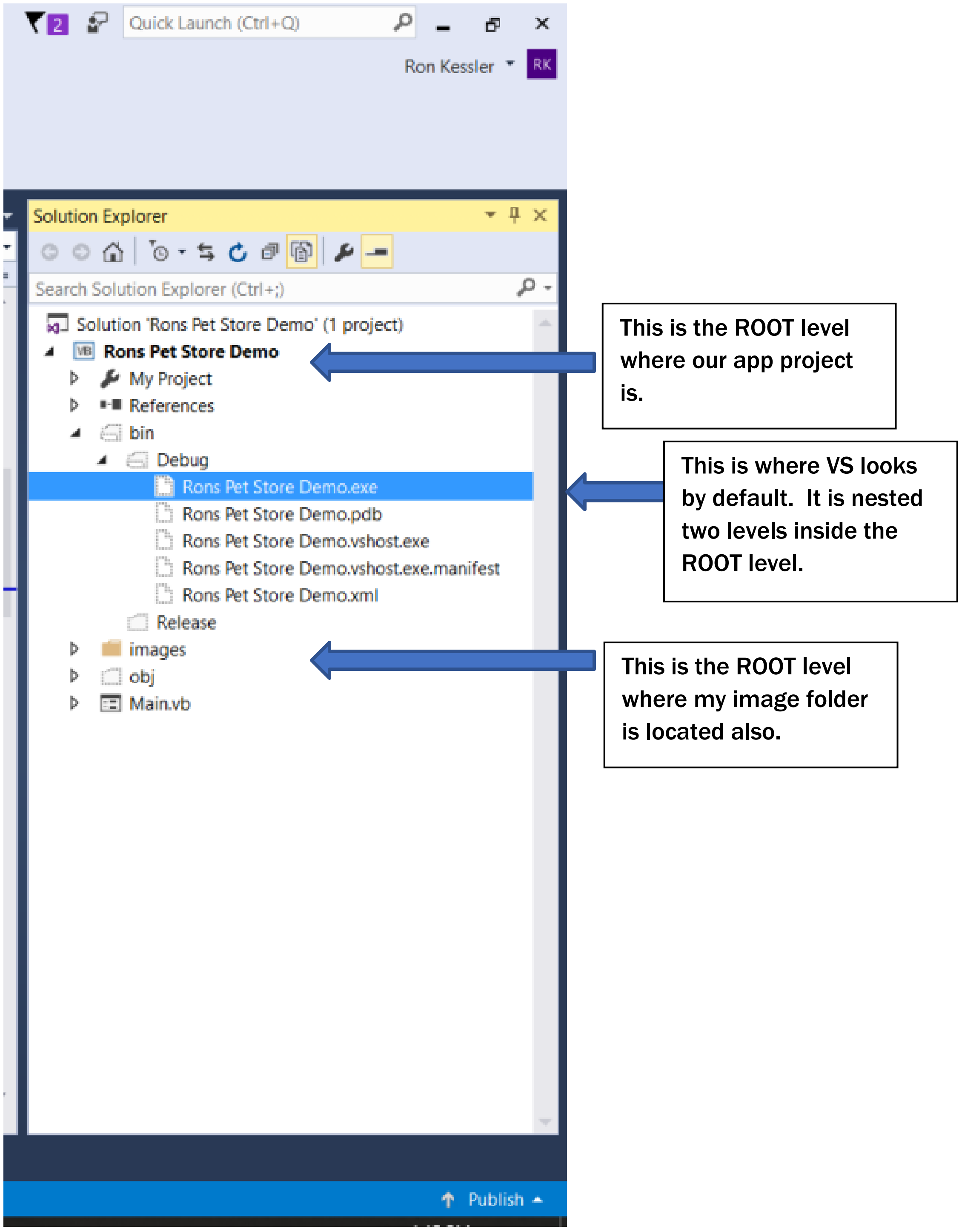
5. Now I must tell VS the path where the image file resides so I can display the banner image at the top of the form. I added a picture box to the top of the form to display this banner. So I will handle that first. I placed it inside the Form_Load event so it will show up when the app is visible.

6. Notice I typed the name of the picture box and use the image property to define the path as you can see here.

   picBanner.Image = Image.FromFile("..\..\images\pet store banner small.jpg")

7. Look at Solution Explorer and click on the small icon to show all files. You will now see a **debug folder inside the bin** folder (bin means binary). This is where your finished app is created. By default, VS looks inside this folder to find music, photo, or other files you add to the project. Since I did not save them in that location (long story), I need to tell the computer where they are.
8. To tell the computer where your image folder is, we use "..\..\" before the folder name. This looks weird I know. This string tells the computer to look two levels up from this bin\debug location.
   My images folder is in the root level just like my Form is. The computer must move up to the bin folder (up one level) then move up again to the images level (up another level). This is why I had to use two of these "..\" hoochies!
9. Look at the next screen shot that shows the project in VS. You can see what I am talking about in the image.

Ron's Pet Store Project in Solution Explorer showing the folder levels.



Solution 'Rons Pet Store Demo' (1 project)
- Rons Pet Store Demo
  - My Project
  - References
  - bin
    - Debug
      - Rons Pet Store Demo.exe
      - Rons Pet Store Demo.pdb
      - Rons Pet Store Demo.vshost.exe
      - Rons Pet Store Demo.vshost.exe.manifest
      - Rons Pet Store Demo.xml
    - Release
  - images
  - obj
  - Main.vb

This is the ROOT level where our app project is.

This is where VS looks by default. It is nested two levels inside the ROOT level.

This is the ROOT level where my image folder is located also.

Parsing the Item Selected in the Listbox

In order to split the string selected in the list box, I added this code to the **lstProducts_SelectedIndexChanged** event.

```
Dim itemChosen As String = ""

Dim itemPrice As String = ""
Dim itemPhoto As String = ""

'---make sure you understand what I am doing here in this section
itemChosen = Trim(Mid(lstProducts.SelectedItem, 1, 10))     'first 10 characters
                                                            hold the item name
itemPrice = Trim(Mid(lstProducts.SelectedItem, 20, 15))        'characters 20-35
                                                            hold the price
itemPhoto = Trim(Mid(lstProducts.SelectedItem, 40))      `characters 40 and on
                                           hold the name of the photo for each item

txtItem.Text = itemChosen
txtPrice.Text = itemPrice
picThumbs.Image = Image.FromFile("..\..\images\" & itemPhoto)
```

1. **First, I created three local variables to hold the item name, the price, and the name of the image file to display (horse, dog, etc.).**
2. **Then I use the VB Mid function to split or parse the selected string. I first tell VB to grab the first 10 characters from the SelectedItem and Trim off blank spaces at the beginning and end of this string. I then tell it to save that data in the itemChosen variable. Remember, as I describe in the video, if you read this line from right to left will make more sense when you are learning.**
3. **The next line takes the characters STARTING at position 25 and grabbing 15 more characters to get the price. BE SURE YOU UNDERSTAND WHAT THIS DOES. IT STARTS AT CHARACTER 20 AND TAKES 15 FROM THERE. SO IT GRABS CHARACTERS 20-35, removes the blanks, and assigns the characters to the itemPrice variable.**

4. Next, I need to extract the file name of the image I want to display when they click on an item. To do that I tell VB to move to character 40 and READ TO THE END OF THE LINE. Notice that when you use a starting number with Mid but no ending number, it just grabs everything from character 40 (in this case) to the end of the selected string. I then trim off the blanks and assign it to the itemPhoto variable.
5. Finally, we can display our results on screen. I now assign the values in my variables to the two textboxes and the picThumbs picture box using the code you see above.
6. If all goes well, when they click an item, that item description and the price will show up instantly in the right textbox. The little picture box will display the image of the product also.
7. Please realize that when you add items to a listbox in this manner, you need to count the location of your items because they will be different than mine. Also, if you want to store 10 pieces of data this way you can. You just have to grab each little substring out of the selected string. When VB stores the selected string, it stores the entire line in memory. We just grab a certain number of characters to use. Therefore, it only reads the selected item one time.
8. Check out the Substring() VB function in your book or online. You may find this easier to work with.

I want you to know that storing information in a listbox this way is terribly inefficient. As you learn more about data structures in more advanced courses, you will see how data can be organized and processed in very cool ways.

Code Listing for Ron's Pet Store

```
Public Class frmMain

    Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load


        '---show store banner in picture box. The images folder is 2 levels below the root folder so I
            have to use "..\" twice!
        '   Your folders may have a differenct structure so be careful.

        picBanner.Image = Image.FromFile("..\..\images\pet store banner small.jpg")

        '---now load the list box
        lstProducts.Items.Add("Birds          49.95          bird.jpg")
        lstProducts.Items.Add("Horse          649.95          colt.jpg")
        lstProducts.Items.Add("Fish            4.95          fish.jpg")
        lstProducts.Items.Add("Cats           29.95           kitten.jpg")
        lstProducts.Items.Add("Puppy           29.95           puppy.jpg")
        lstProducts.Items.Add("Rabbit          39.95           rabbit.jpg")
        lstProducts.Items.Add("Sheep           249.95          sheep.jpg")
        lstProducts.Items.Add("Turtle          6.49          turtle.jpg")


    End Sub

    Private Sub lstProducts_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles lstProducts.SelectedIndexChanged

        '---start with 1 item and clear the previous sale
        txtQuantity.Text = 1
        lblTax.Text = ""
        lblTotal.Text = ""

        Dim itemChosen As String = ""
        Dim itemPrice As String = ""
        Dim itemPhoto As String = ""

        '---make sure you understand what I am doing here in this section
        itemChosen = Trim(Mid(lstProducts.SelectedItem, 1, 10))        'the first 10 characters hold
the item name
        itemPrice = Trim(Mid(lstProducts.SelectedItem, 20, 15))        'characters 20-35 hold the
price
        itemPhoto = Trim(Mid(lstProducts.SelectedItem, 40))            'characters 40 and on hold
the name of the photo for each item

        txtItem.Text = itemChosen
        txtPrice.Text = itemPrice
        picThumbs.Image = Image.FromFile("..\..\images\" & itemPhoto)

    End Sub
    Private Sub btnCheckout_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCheckout.Click

        Dim numItems As Integer = 0
        Dim salesTaxRate As Decimal = 0
        Dim totalPrice As Decimal = 0
        Dim taxDue As Decimal = 0
        Dim itemPrice As Decimal = 0

        '---compute item price and assign it
        itemPrice = CDec(txtPrice.Text)

        '---get number of items
        numItems = CInt(txtQuantity.Text)

        '---assign tax rate
        salesTaxRate = 0.0775


        '---compute tax
        taxDue = salesTaxRate * numItems * itemPrice

        '---format tax into label with 2 decimal places
        lblTax.Text = FormatNumber(taxDue, 2)

        '---compute total due & put it into label formatted as currency
        totalPrice = (itemPrice * numItems) + taxDue
        lblTotal.Text = FormatNumber(totalPrice, 2)

    End Sub

    Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnQuit.Click

        Me.Close()

    End Sub
End Class
```

Terms and Concepts To Know From Ron's Pet Store

- How to work with picture boxes.
- Add data to a listbox at run time in code
- How to use the Form_Load event handler.
- How to store pictures in a folder inside your solution.
- How to access files from another folder in code.
- How to parse three pieces of data from a string.
- How to show thumbnail images in a Picture Box.
- How to use the smart tag of a picture box.
- Make sure you can compute sub-totals, sales tax, and total due when using a quantity variable.
- How to format output to 2 decimal places WITH/WITHOUT a '$' sign.
- How to tell the computer to find files/folders inside the project folders structure.
- Be sure you understand the **MID** function. I also suggest you learn the Substring() method since we use that in C# and C++.

Lesson 4: Decision Structures

This part of Lesson 4 shows how to make your code handle decisions. In VB the two I teach you are **IF statements, and Select Case**.

Once you learn to master these, you will be ready to create serious projects.

Lesson 4: Decision Structures

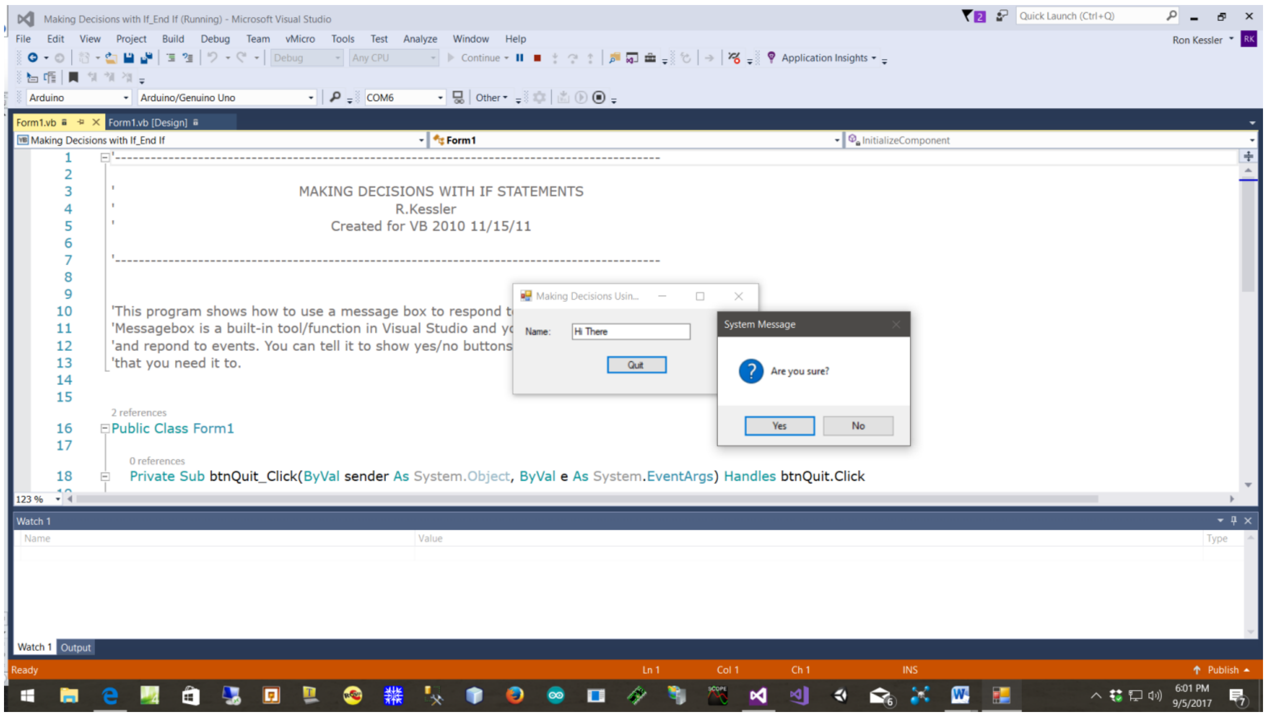
How to Use If Statements

Textbook Chapters & Videos for Lesson 4

- Murach Chapter 5
- Videos under lesson 4 on If Statements and Select Case. Be sure to look at the slides n that section as well!

What to do now:

- I want you to download, unzip, and open this demo project so you can follow along with me.
- Now listen to the video for this project. The project is shown below.
- Now open this slide deck and look at the first part on 'If' statements so you can see what options you have. Stop at the 'Select Case' part. That comes next.
- When you run this project and click the Quit button, the program prompts you with "Are you sure?". If you click 'Yes', the app ends. If not, I show another annoying message. This is a very short project but it shows you how to create a way for users to decide things. It also shows how to get a response back from Mr. MessageBox and that is an important skill.



```
'-------------------------------------------------------------------------------------

'                    MAKING DECISIONS WITH IF STATEMENTS
'                                R.Kessler
'                         Created for VB 2010 11/15/11

'-------------------------------------------------------------------------------------


'This program shows how to use a message box to respond t
'Messagebox is a built-in tool/function in Visual Studio and y
'and repond to events. You can tell it to show yes/no buttons
'that you need it to.


2 references
Public Class Form1

    0 references
    Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnQuit.Click
```

A Closer Look At the Code

As you can see from the code below, when they click btnQuit, I display a message box with a message and yes & no buttons. Then, based on which button they click, the app responds.

The first If statement displays the message box but you can see at the beginning and end of that line there are new terms.

Let's walk through it. The first part starts with the If keyword. This lets VB know we are expecting some type of response or decision to be made. The Messagebox.Show displays a regular box like you already learned to do. However, notice I told it to add 'YesNo' buttons… not just an 'OK' button. I also use a question mark icon to make it look professional.

```
If MessageBox.Show("Are you sure?", "System Message", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question)
```

The last part of this statement checks to see if they clicked on the "Yes" button. You code that by placing these instructions after the "=" sign:

```
= Windows.Forms.DialogResult.Yes Then
         Me.Close()
```

In the latest versions of VB, the message box is now considered a Dialog Box. So, in order to see if they clicked Yes, the code above is required. If they in fact did click 'Yes', we type in a **Then** keyword and on the next line, we tell VB what we want it to do. In this case close the app using Me.Close().

But wait! What if they clicked 'No'? Well notice I coded an 'Else' block where I show another message box just to bug them!

When you use the If logic like this, you are using Block Mode. You are writing code to test for a specific condition. When the user responds, you get to decide what the app does.

Here is the complete Code Listing for you.

```
Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnQuit.Click

        If MessageBox.Show("Are you sure?", "System Message", MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question) = Windows.Forms.DialogResult.Yes Then
                    Me.Close()
        Else
           MessageBox.Show("Then make up your mind", "Last Chance", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation)
        End If

    End Sub
```

- Here are the rules. **Every 'If' must have an 'End If' clause.**
- If VB discovers they clicked the 'Yes' button, it never bothers checking the Else block. The choice is clear...choose one or the other but not both.
- Notice I used an Else clause to handle the case when they click 'No'. I did not create another If statement that explicitly tests the 'No' button. If they did not click 'Yes' then they had to click 'No'.
- You only use one Else clause.
- If I wanted to test multiple conditions, I would use 'Else If' statements. You can have any number of them if you need to test several conditions,
- I suggest you type all this in lower case and let VB fix it for you to save time!

Terms and Concepts To Know For If Statements

- How to create a one-life If statement
- How to use block if statements
- What is Block Scope?
- Nested If statements
- How to get a response from Mr. MessageBox.
- How to do multiple If tests (If, Else If).
- Boolean expressions
- Relational operators
- Logical operators
- Short-circuit operators

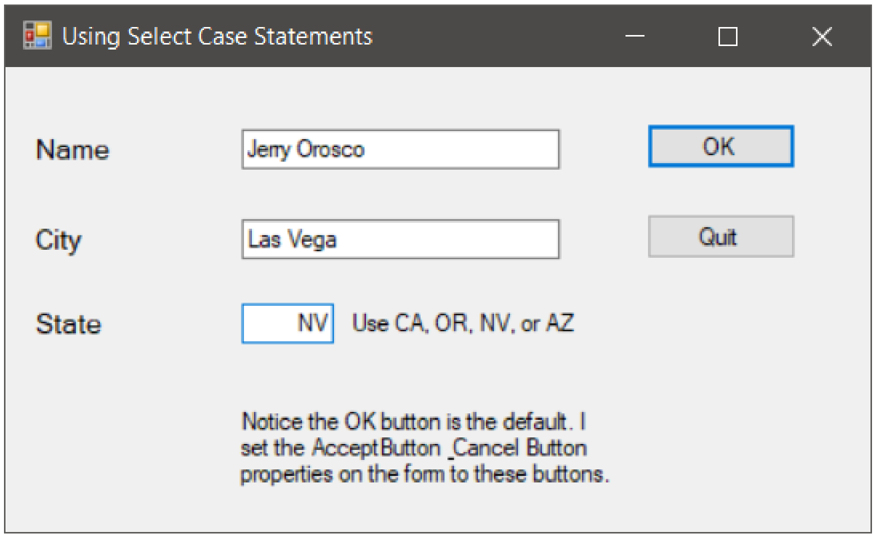
How to Use Select Case

What to do now:

- First of all, let me tell you that in VB, the Select Case structure is very similar to the "switch" structure in C#. However, as you work with it, you will find it much more useful than switch. VB has many more useful options so I think you will like it.
- Why do I need to use Select Case in the first place? Well, in large programs that have to make numerous decisions, this structure is way cleaner and easier to edit than a bunch of If statements. I use this whenever I can. Once you see an example I think you will understand why I like this so much.
- So download this Select Case Demo I made for you here or from lesson 4 online and open it up.
- Next, listen to the video for this project. The project is shown below.
- Now re-open the slide deck on If & Select Case you looked at before and look at the part on 'Select Case' statements so you can see what options you have. Stop at the Loops part.

How it Works

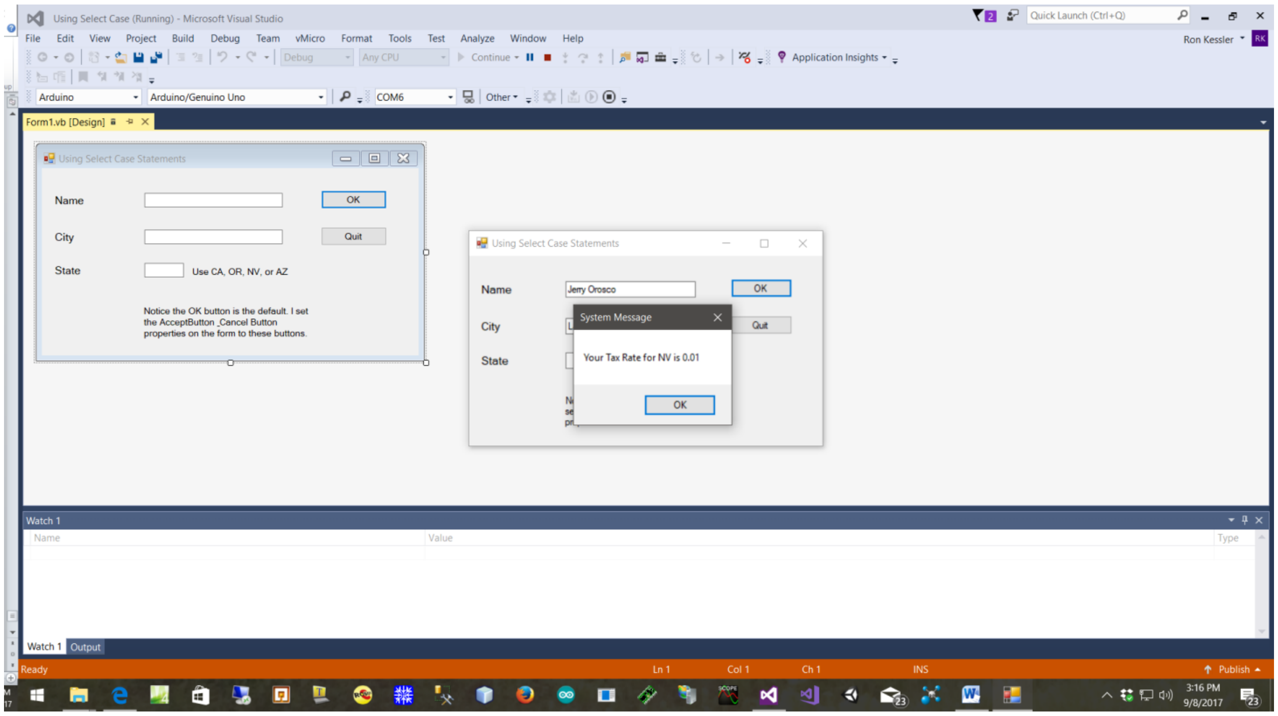
When you run this project and fill in the information and click OK, the program computes the sales tax for the state you entered. Notice I limit your choices. Also, because of a little trick I will show you, you can type in the state codes in upper or lower case.



Name: Jerry Orosco
City: Las Vega
State: NV Use CA, OR, NV, or AZ

Notice the OK button is the default. I set the AcceptButton ,Cancel Button properties on the form to these buttons.

As you can see in the next screen shot, I use a Message Box to display the results.



Notice the OK button is the default. I set the AcceptButton ,Cancel Button properties on the form to these buttons.

Your Tax Rate for NV is 0.01

The Quit button behaves like the previous demo on If statements. This is, it prompts you with "Are you sure?". If you click 'Yes', the app ends. If not, I show another annoying message.

I also use the CancelButton property of the form to let the user hit the ESC key to exit. To do that, go to the properties of the FORM… not the button. Select the CancelButton property and use the drop down box to select the btnQuit button.

A Closer Look At the Code

First of all, I need some variables to hold my data. So in the btnOK_Click handler I created two local variables named salesTaxRate and stateChosen.

```
'---now let's decide what the sales tax rate is based on the state they chose

        Dim salesTaxRate As Single
        Dim stateChosen As String
```

I made the first variable a Single precision number. This is analogous to a float in C# and C++. I did it to get you used to working with different data types. salesTaxRate will hold the tax rate for the chosen state.

My other variable is used to hold the state code they typed in. I get their input from the textbox like you have done many times. But notice what I did. I took what they typed and told VB to capitalize the letters by using the command UCase (upper case). And yes, VB has a LCase (lower case) conversion function also. This eliminates testing the input from the user. No matter how they type in "CA" for example, my code works for "ca", "Ca", or "cA".

```
'---capitalize what they typed using the UCase built-in VB function
        stateChosen = UCase(txtState.Text)
```

Now let's look at how the program decides what the tax is going to be. Instead of using a bunch of If statements, I use Select Case.

Here are the rules.

- Each Select Case structure must have an End Select as you see here.
- The value you want to evaluate (stateChosen) on the same line as the Select Case keyword.
- Your choices are created in Case statements. You can have as many as you want.
- Unlike switch in C# and C++, we can "switch" on strings! This is a huge advantage.
- As you will see in my slides, you can use a range of values to Select on and this makes the structure very flexible.

```
stateChosen = UCase(txtState.Text)

        Select Case stateChosen

            Case "CA"
                salesTaxRate = 0.08
            Case "AZ"
                salesTaxRate = 0.05
            Case "NV"
                salesTaxRate = 0.01
            Case "OR"
                salesTaxRate = 0.06

            Case Else
                MsgBox("Please enter CA, AZ, OR, NV")
                txtState.Text = ""
                txtState.Focus()
                Exit Sub
        End Select
```

How it Works

When the user types in a state code like "AZ" and clicks OK, my code grabs the input from the textbox, upper cases it, and passes it to the Select Case section. Now the computer attempts to find an exact match between what was typed in and one of the case statements. WHEN UISNG STRINGS LIKE THIS, YOU MUST TYPE THE STATE CODES IN UPPERCASE so VB can find an exact match. Strings that are inside quotes “” are called string literals. That tells the computer to treat the characters exactly as they are typed and don’t accept anything else.

So let’s say they typed in “AZ”. VB checks the first case (“CA”) and there is no match. It works its way done the cases until it finds a match. When it finds “AZ” it executes the code inside that case block. You can have several lines of code in there if you like. Here it assigns a value to my variable of 5% (.05).

Then, and here is the part to remember, IT QUITS TESTING. Once a case is found, the code jumps down to the End Select line and keeps going if there is more to do. This behaves just like a block If statement does.

But you recall that we used an “Else” block with those If statements to catch any values that didn’t match our tests. Well we do the same thing here. It is called **Case Else and you should always use it!**. Notice I show a Message Box to give them a chance to try again because they didn’t follow directions or made a typo. Oh yeah, I used MsgBox to show a simple box. This is the original message box function when I learned VB years ago and I forgot to change it. But you might find it useful when you need a simple box like I did here.

To me, this code is easier for me to figure out and easier to edit and add new cases than a huge If block of ElseIf and all that. Anyway, now you have two decision structures you can use. In the project, I show how you could do the same thing with If statements like this code below.

```
If stateChosen = "CA" Then
   salesTaxRate = 0.08
ElseIf stateChosen = "AZ" Then
   salesTaxRate = 0.05
ElseIf stateChosen = "NV" Then
   salesTaxRate = 0.01
ElseIf stateChosen = "OR" Then
   salesTaxRate = 0.06
Else
   MsgBox("Please enter CA, AZ, OR, NV")
   txtState.Text = ""
   txtState.Focus()
   Exit Sub
End If
```

Finally, I output the results in a little message box right after the Select Case section. This code should be very familiar to you so I don't need to discuss it here.

```
'---now display a message box and the tax rate
MessageBox.Show("Your Tax Rate for " & stateChosen & " is " & salesTaxRate, "System
  Message")

txtState.Clear()     'start over
txtState.Focus()
```

Here is the complete Code Listing for you.

```
'=============================================================

'       U S I N G   S E L E C T   C A S E   S T A T E M E N T S

'                              Created for VB 2010
'                                   Ron Kessler
'=============================================================

'Updated 11/15/2011


Option Explicit On


Public Class frmMain

   Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles btnOK.Click

      '---now let's decide what the sales tax rate is based on the state they chose

      Dim salesTaxRate As Single
      Dim stateChosen As String

      '---capitalize what they typed using the UCase built-in VB function
      stateChosen = UCase(txtState.Text)

      Select Case stateChosen

         Case "CA"
            salesTaxRate = 0.08
         Case "AZ"
            salesTaxRate = 0.05
         Case "NV"
            salesTaxRate = 0.01
        Case "OR"
            salesTaxRate = 0.06

        Case Else
            MsgBox("Please enter CA, AZ, OR, NV")
            txtState.Text = ""
            txtState.Focus()
            Exit Sub
    End Select

    '==================USING IF-THEN STYLE===================

    'If stateChosen = "CA" Then
    '    salesTaxRate = 0.08
    'ElseIf stateChosen = "AZ" Then
    '    salesTaxRate = 0.05
    'ElseIf stateChosen = "NV" Then
    '    salesTaxRate = 0.01
    'ElseIf stateChosen = "OR" Then
    '    salesTaxRate = 0.06
    'Else
    '    MsgBox("Please enter CA, AZ, OR, NV")
    '    txtState.Text = ""
    '    txtState.Focus()
    '    Exit Sub
    'End If
    '===============END OF USING IF-THEN STYLE================

    '---now display a message box and the tax rate
    MessageBox.Show("Your Tax Rate for " & stateChosen & " is " & salesTaxRate, "System
      Message")

    txtState.Clear()      'start over
    txtState.Focus()

  End Sub
```

Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnQuit.Click

```
    If MessageBox.Show("Are you sure?", "System Message", MessageBoxButtons.YesNo,
      MessageBoxIcon.Question) = Windows.Forms.DialogResult.Yes Then
        Me.Close()
    Else
        MessageBox.Show("Then make up your mind!", "Confused User",
      MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

    End If
  End Sub
```

End Class

Terms and Concepts To Know For Select Case Statements

- How to set up Select Case structure
- How to tell Select Case what it is selecting on
- Understand how it finds a match and how it behaves.
- What is the Case Else used for?
- What kinds of data types can we evaluate with Select Case?
- Be sure to look in your Murach book to see the options for this structure

That completes this course! I hope you learned a lot and found the book and videos helpful. Everything you learned here applies to my class on C#.