Visual Basic 101

Course Handbook Supplement

By Richard Rost

Published By
Amicron Computing
PO Box 1308, Amherst NY 14226 USA
www.599cd.com

First Printing **1/25/2006**

Welcome

Welcome to the 599CD **Visual Basic 101** Handbook.

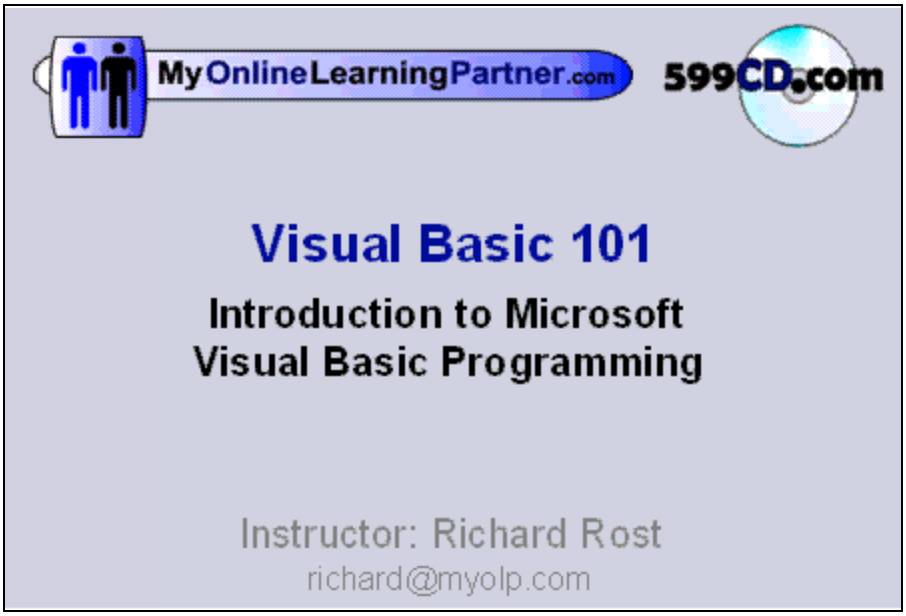
This handbook is designed to be a **supplement** to the full 599CD video course for **Visual Basic 101**. We recommend you use this handbook to follow along with the class videos. This handbook is not meant as a stand-alone study guide.

We do recommend that you watch the course videos one time through, paying attention to the lessons covered. Follow along with the course videos using this guide. Take notes on the pages where needed. Then, watch the videos a second time , practicing the examples yourself on your computer.

Table of Contents

Introduction

Welcome to **Visual Basic 101**, brought to you by **599CD.com**. I am your instructor, Richard Rost.



Objectives for today's class:

- Learn about Visual Basic
- Create a Program
- Learn Basic Commands like MsgBox
- Learn about Controls, such as buttons and text boxes
- Decision Making (if / then)
- Compiling your Program

Our goal for today is to get you up and writing programs in about an hour.

Pre-Requisites: Basic knowledge of Windows

In this class we will be using **Microsoft Visual Basic 6.0 and Windows XP** for our live-action videos. If you are using an earlier version of VB, you should not have a problem following these videos. VB.NET users will find significant differences in the physical appearance of the VB environment, however the programming topics we will discuss are relatively the same. The version of Windows you have does not matter. Most, if not all of these examples should be applicable regardless of what version of VB you're using.

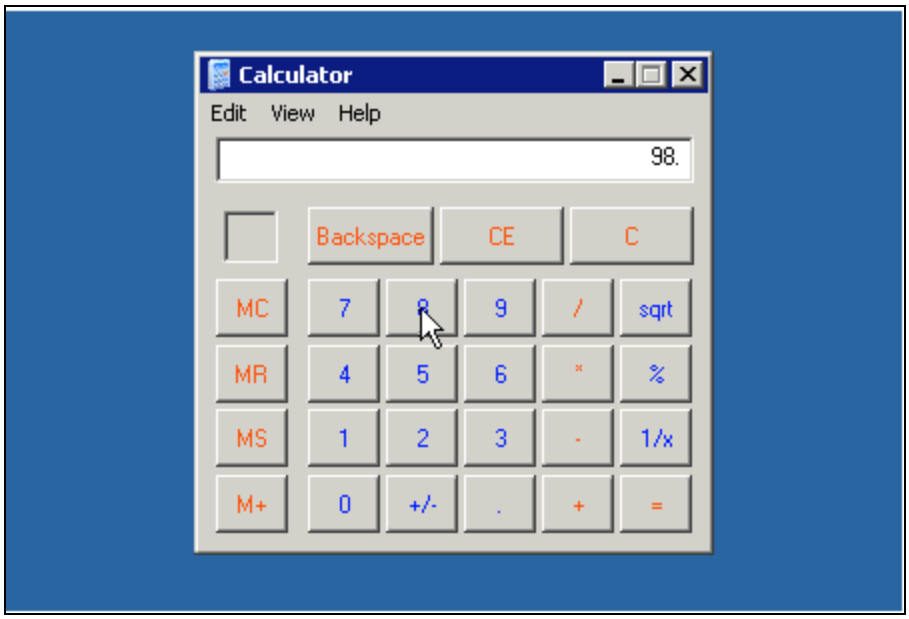
Lesson 2. What is a program?

We have to start out with some basic definitions for what we're doing today. First, what exactly is a **program**? The dictionary defines a **program** as a set of coded instructions that enables a computer to perform a desired sequence of operations.

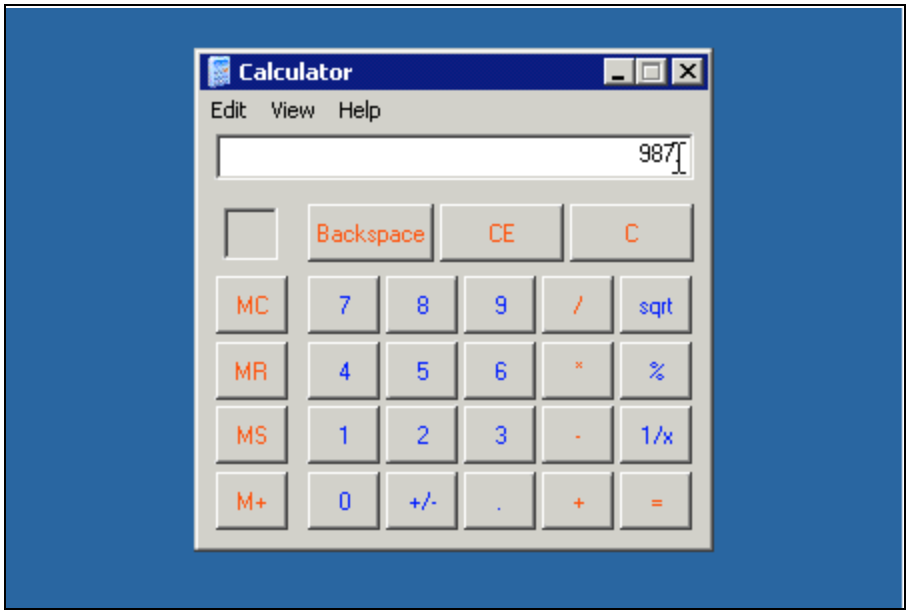
That's essentially all a program does. It performs a series of operations based on the instructions given to it by the computer programmer. That's you. Now in the old days, programmers actually had to program their stuff on "punch cards" which I'm sure was no fun. Then we had to type everything in on the keyboard line by line. That was no fun either. Then finally Windows came along, and made programming much, much easier.

Instead of typing in lines of code, Windows programs are made up of **objects**, like command buttons that you push, text boxes that you type information into, and menu bars that you can select functions from. Windows programs are not even easier to write, but they're easier to use.

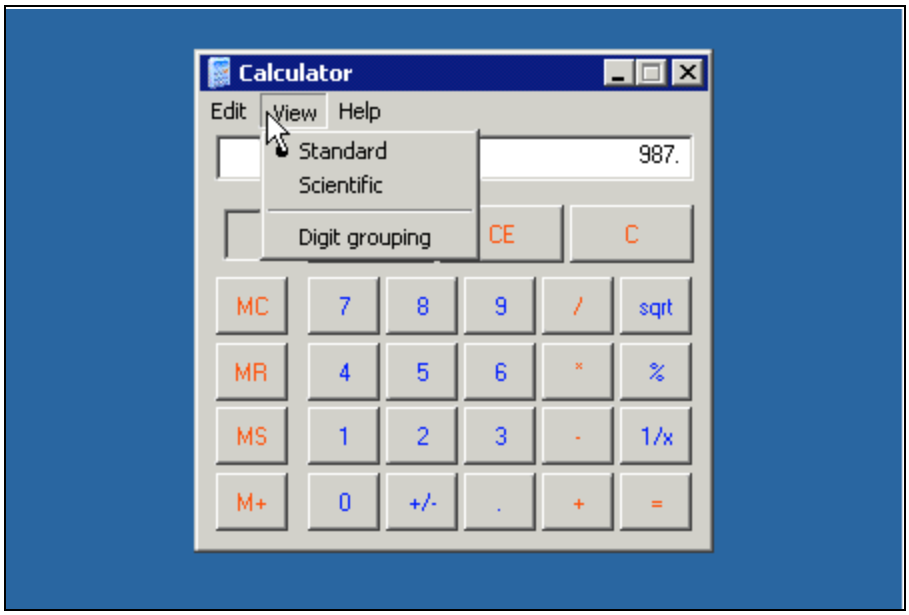
Here's one of my favorite age-old windows programs: the Microsoft Calculator. It comes with Windows and it illustrates many of the common components that you'll find in Windows programs. Calculator has **command buttons** (those little buttons that you can push).



And we can put numbers in its **text box**, or we can type numbers right into the text box.



There's a **menu bar** that allows us to pull down menus.



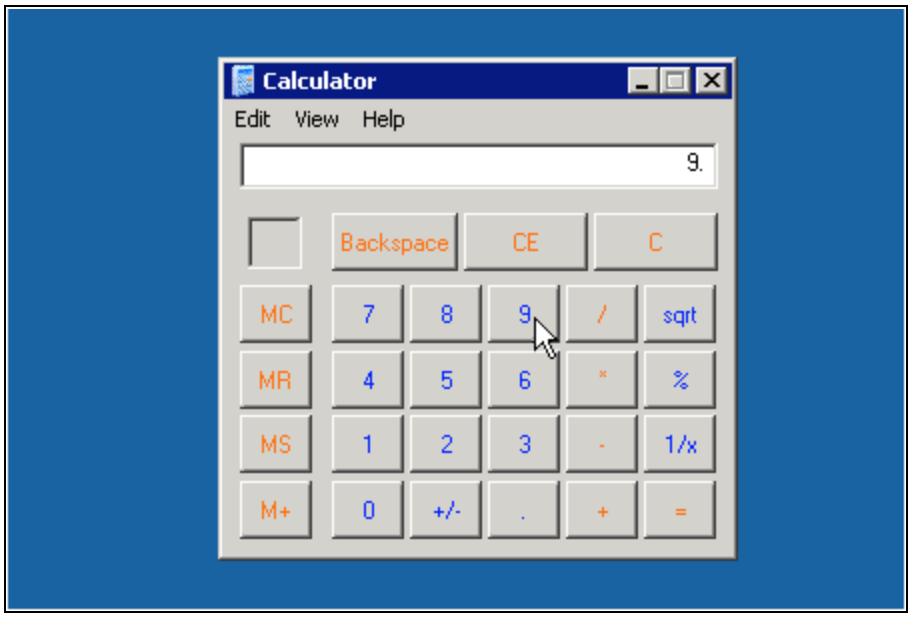
And of course we have our standard minimize, maximize, and close buttons.

So the objects in Windows programs: the command buttons, the text boxes make programming and using a program much, much easier.

Windows programs and all the objects in them generally have **properties, methods, and events**. **Properties** describe a characteristic of a program or an object. For example, a button might have a height, a width, and a color. Those all represent properties.

Windows programs also have **methods**. Methods are built-in functions that a program performs. For example, when we start calculator, it knows to pop a little zero in the window and wait for user input. That's a built in method.

And most Windows programs, if not all, have **events**. Windows programs are event-driven. That means they sit around waiting for the user to do something, generating an event. If we click on the number 9 button, that generates an event inside the program.



There's a little programming code underneath this button (that we'll see how to do today) that then puts a nine in the text box. That's called an event.

Properties, methods and events make up the core of Visual Basic programming.

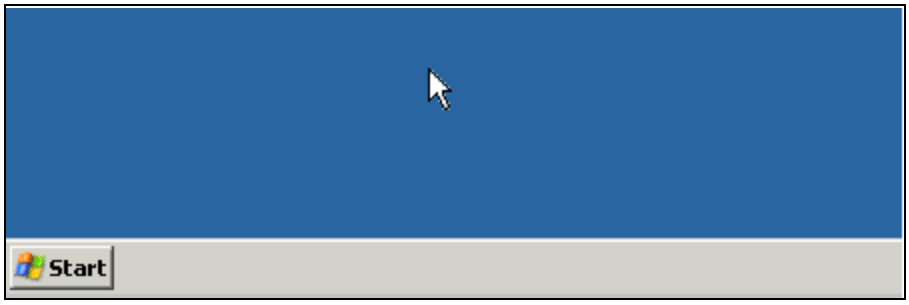
So what is Visual Basic?

Visual Basic is a **programming environment**. It includes a **programming language**, and a **graphical user interface**. The programming language allows us to enter in commands that the computer will execute. The graphical user interface allows us to create forms and buttons and text boxes and then other things visually by clicking them and dragging them with a mouse. The combination of the two provides an extremely powerful programming environment.

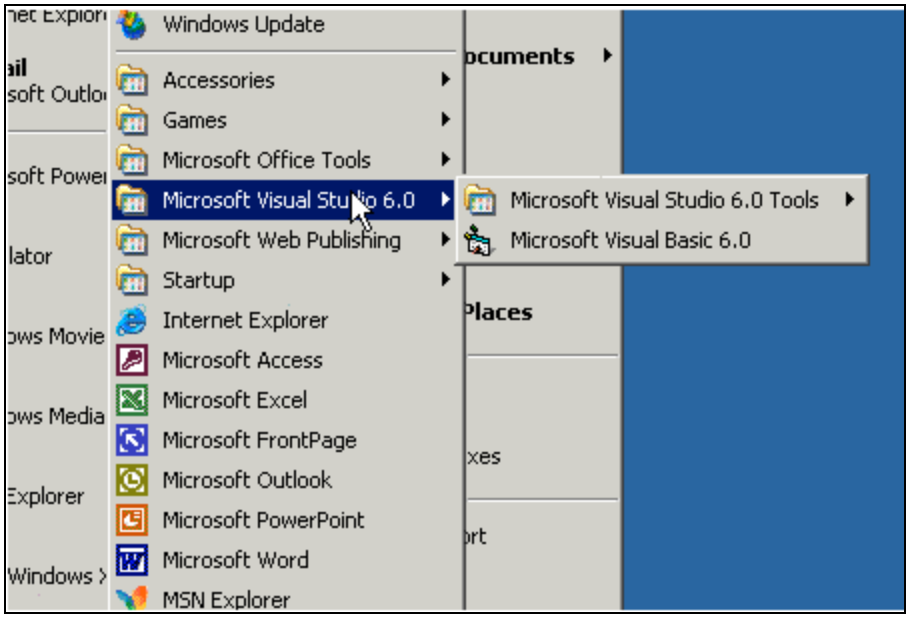
Lesson 3. Our First Program

Now, lets start Visual Basic, and build our first program.

Here we are at the Windows desktop.

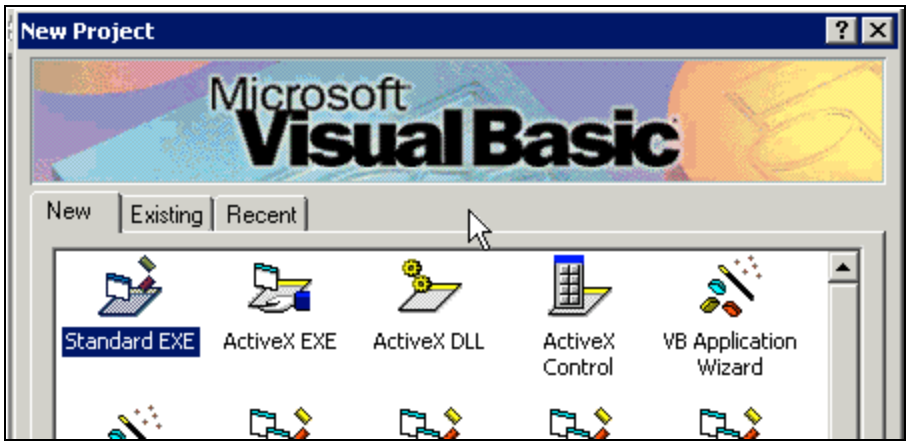


We're going to click on "Start", then "All Programs," and you should **see Microsoft Visual Studio 6.0 or Microsoft Visual Basic 6.0** (depending on what you installed)

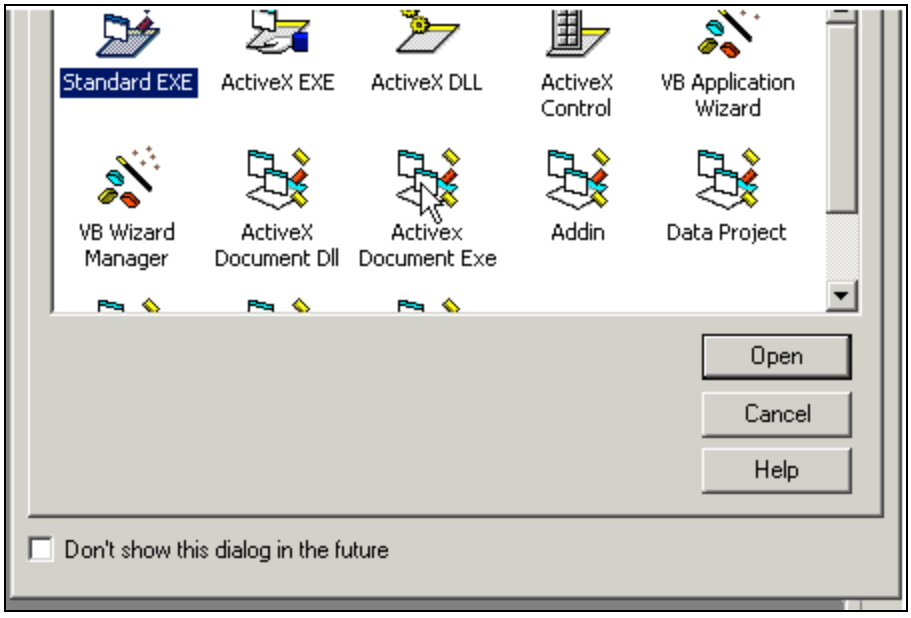


And then you'll see **Microsoft Studio 6.0 Tools and Microsoft Visual Basic 6.0**. This is what we're going to use today. Go ahead and click on Microsoft Visual Basic 6.0.

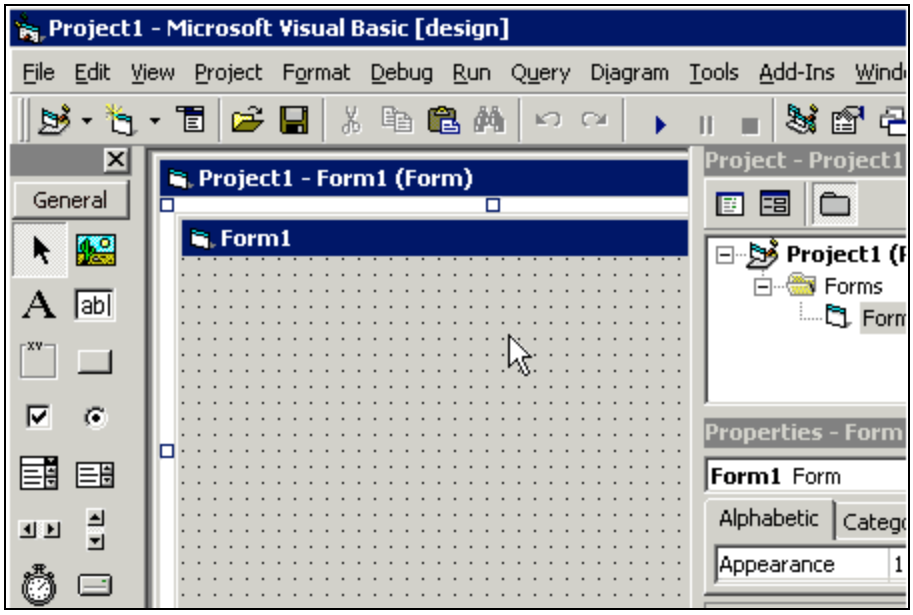
The first window we get up is the new project window.



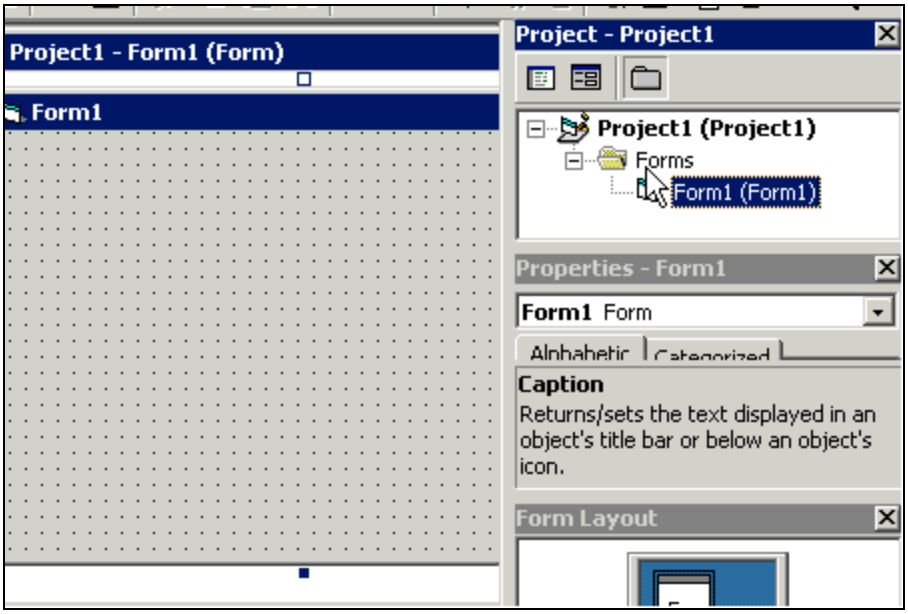
Now there are many ways to start new projects. A **project** in Visual Basic is essentially your program. For today, make sure **standard.exe** is selected and then click on open.



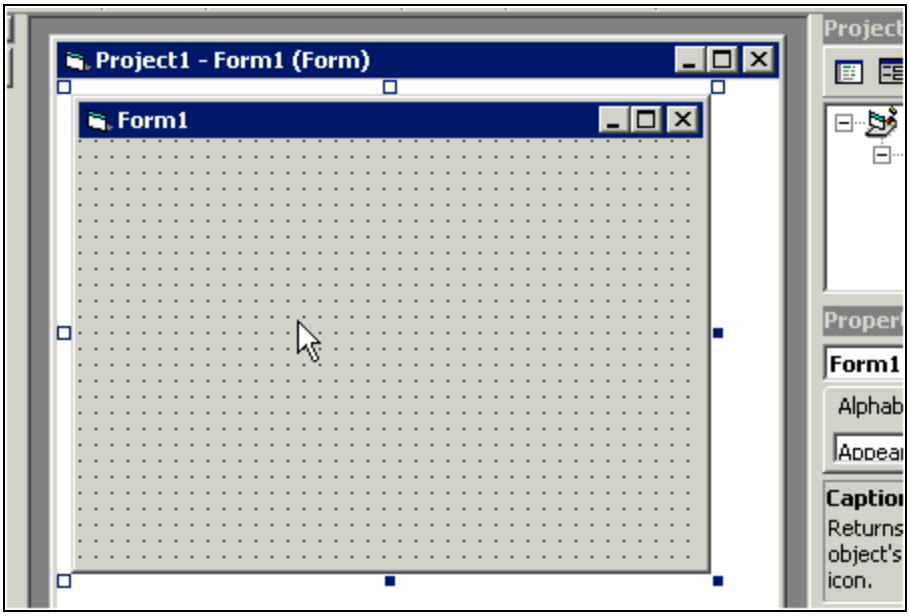
Now we're sitting at **Project1** (a blank project) in Microsoft Visual Basic. You'll notice the Visual Basic interface is very similar to other Windows programs. We have our title bar across the top, our menu bar, and a tool bar. There's also a tool box going down the left hand side.



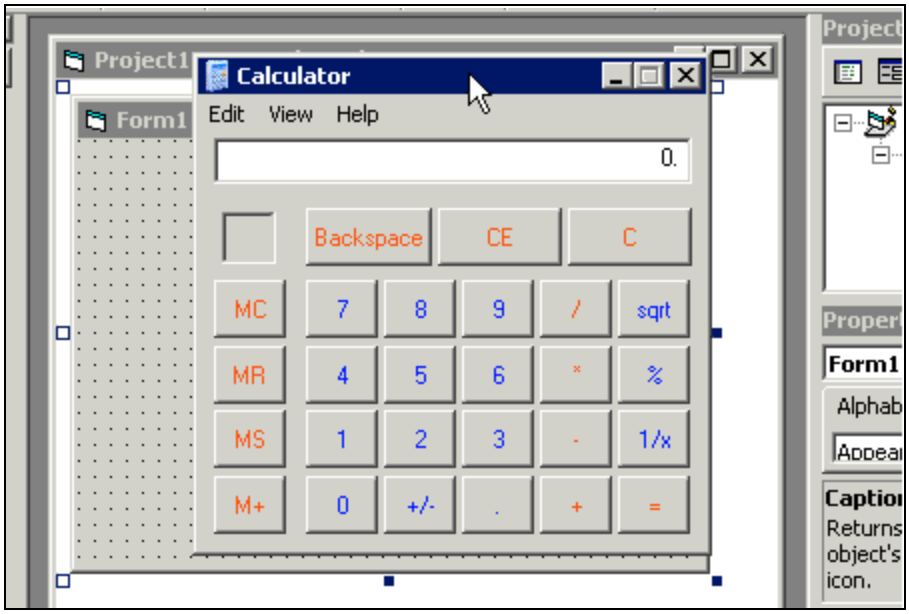
In the middle we have a big blank form – this is where our program will go. On the far right we have our project window, a properties window, and a form layout window.



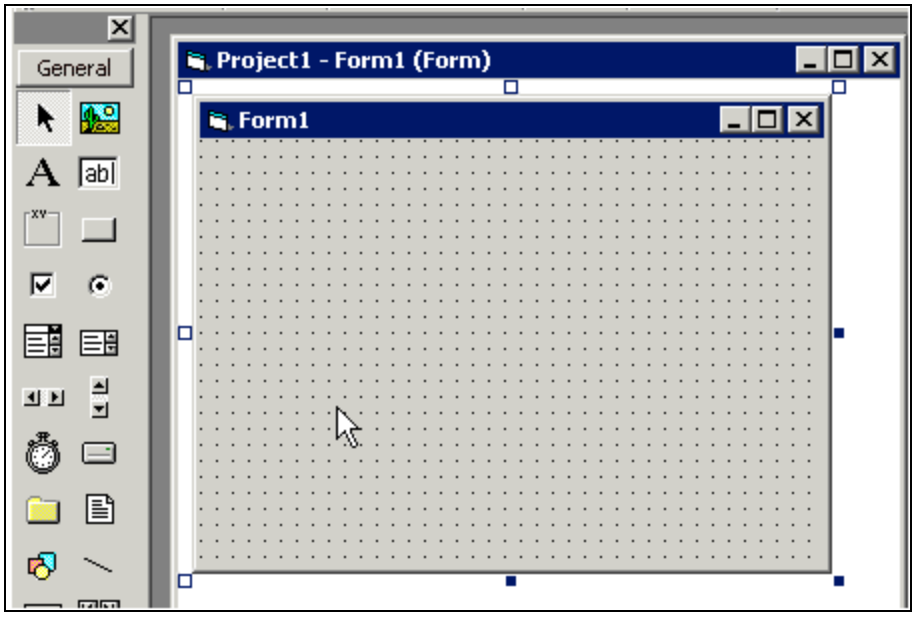
Turn your attention for moment if you will, to this **Form1** window in the center.



Just like the calculator program that we looked at a minute ago, most Visual Basic applications will run on a form. In fact, most Visual Basic applications will have one or more forms. Here's calculator again.

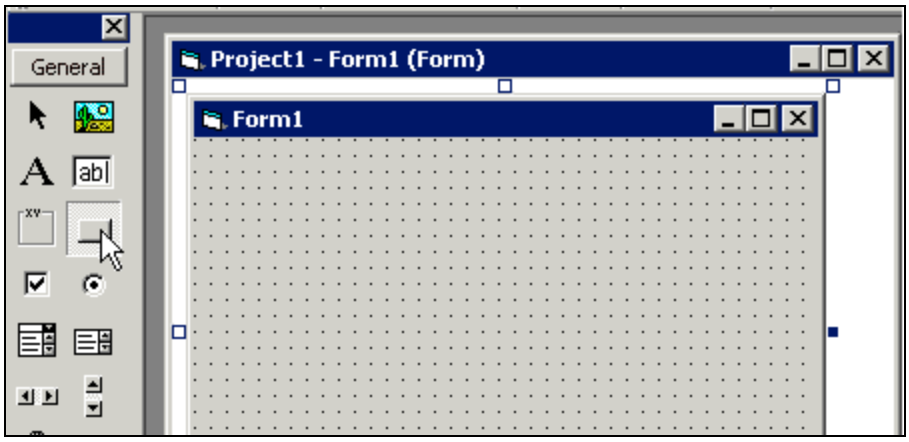


Just picture in the background, behind all these buttons, the big blank empty form. This form is essentially our canvas and on it, we're going to paint different controls.

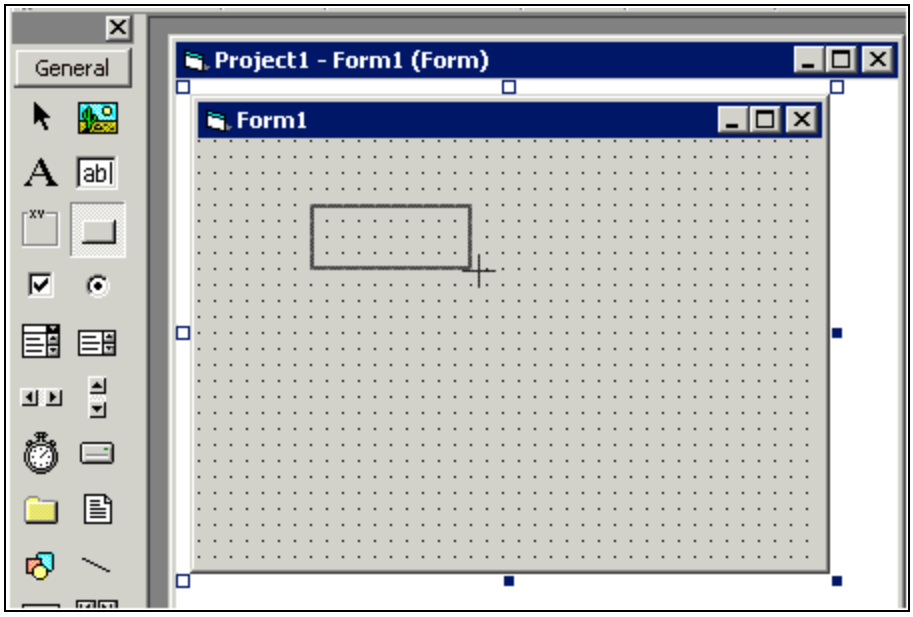


In controls we have buttons and labels and text boxes and pictures and so on. Using these different controls we can create our own custom programs just like calculator.

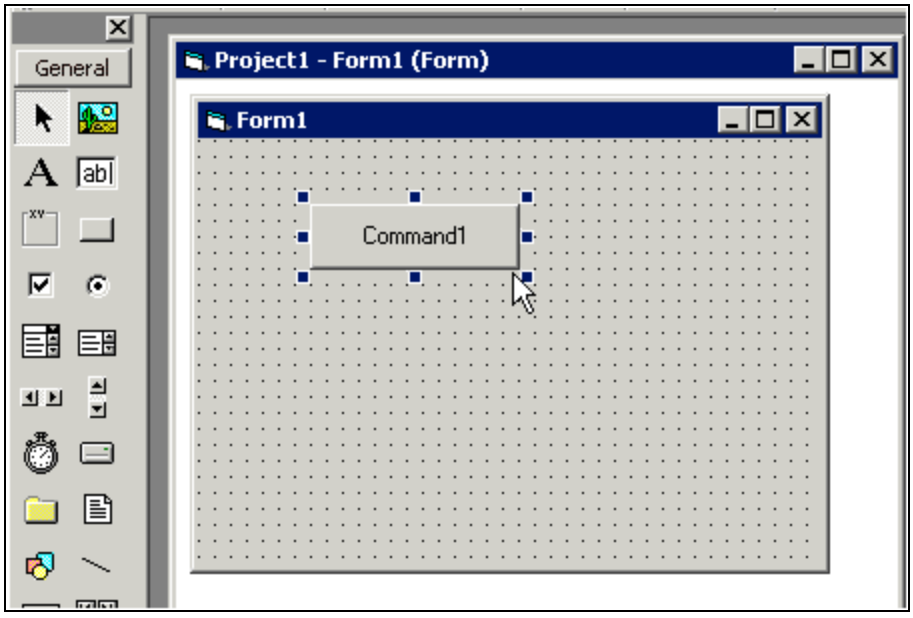
Lets start by building our very first program by putting a button on this form and making this button give us a little message when the user clicks on it. Sounds pretty simple right? Let's take our mouse and find the **command button**.



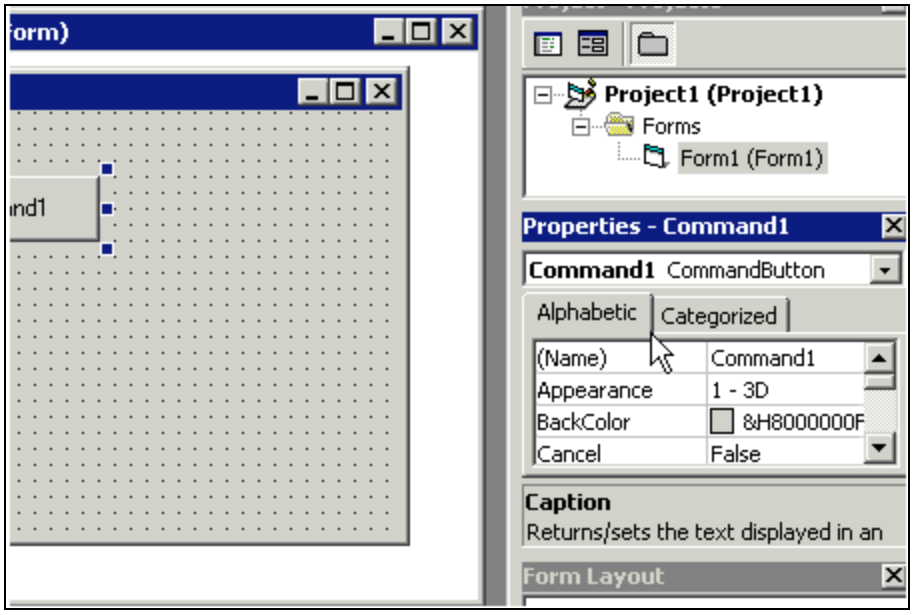
Go ahead and clic k on it and take your mouse over on the form. Notice that the mouse pointer has changed into a plus instead of a pointer icon. Go ahead now and click and drag to draw a little box.



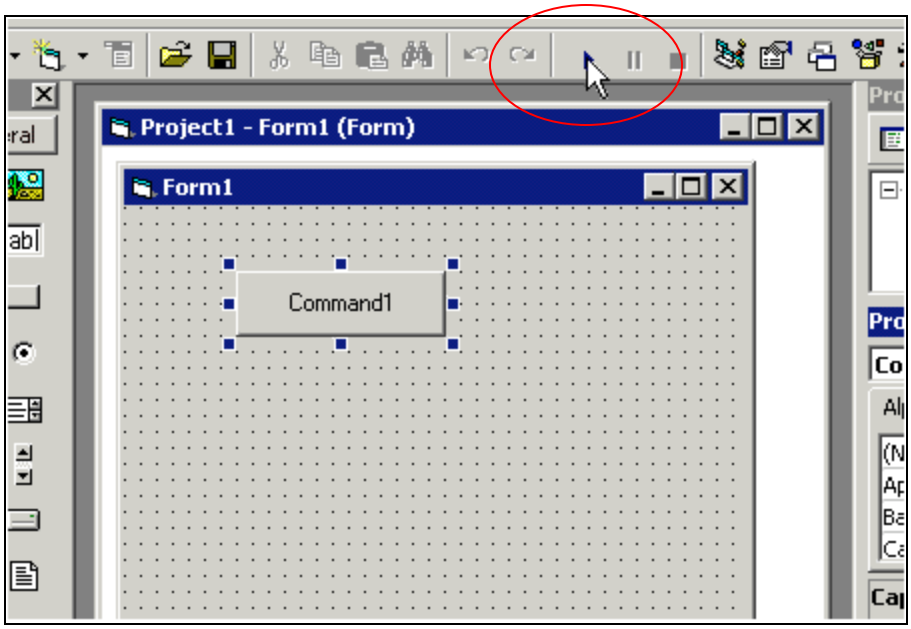
Let it go, and that will create a command button.



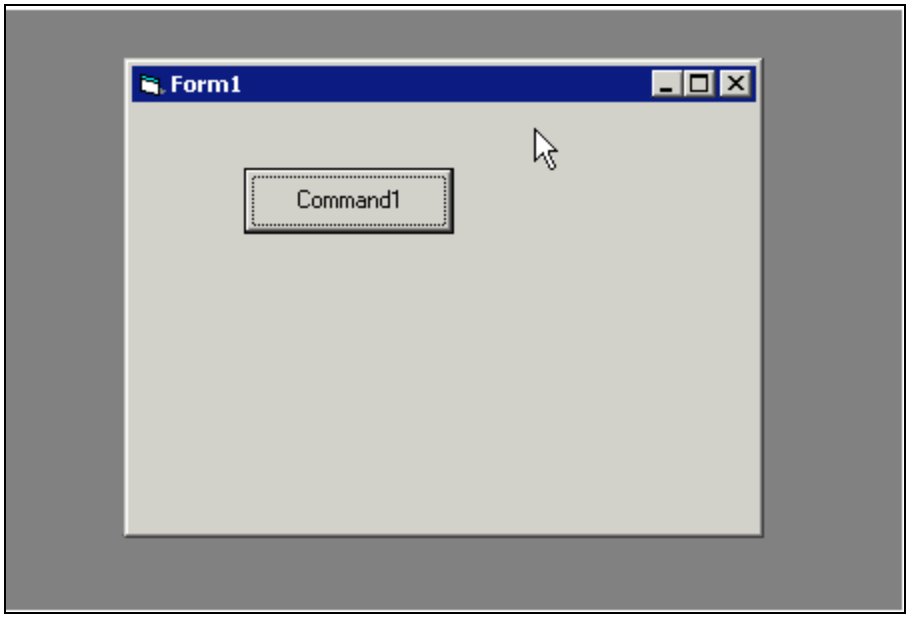
The name of this button is **Command1** . In Visual Basic, all the different objects that you will create, whether they're buttons, or pictures or text boxes will all have a name. That name is one of their **properties**. In fact, you can see the name in the Properties window that says, "Properties for Command1".



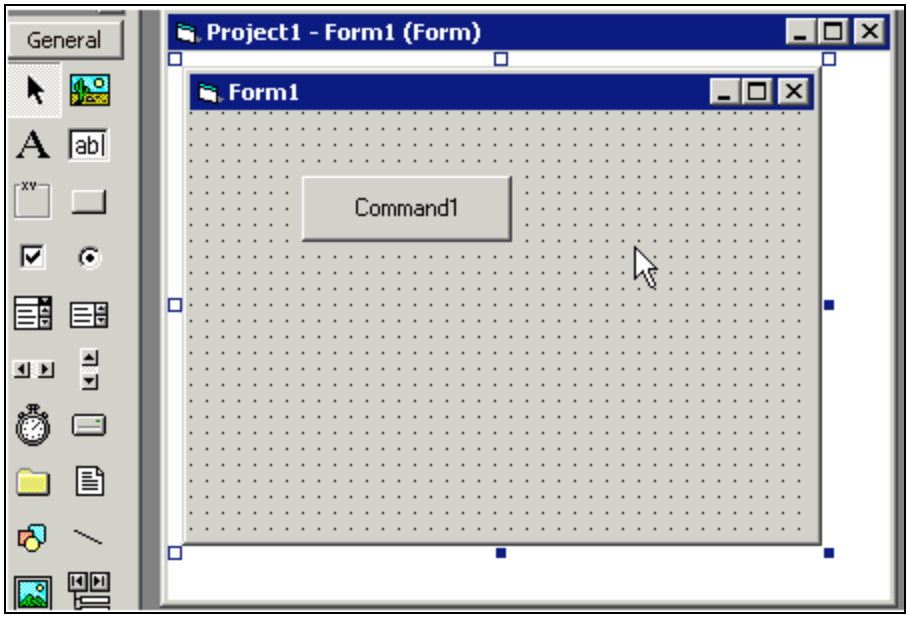
Now that we have a button on our form, let's go ahead and run our program. In other words, we're going to start up our program and see what happens when we click on that button. To run your program, find the button that looks like a little triangle that points to the right.



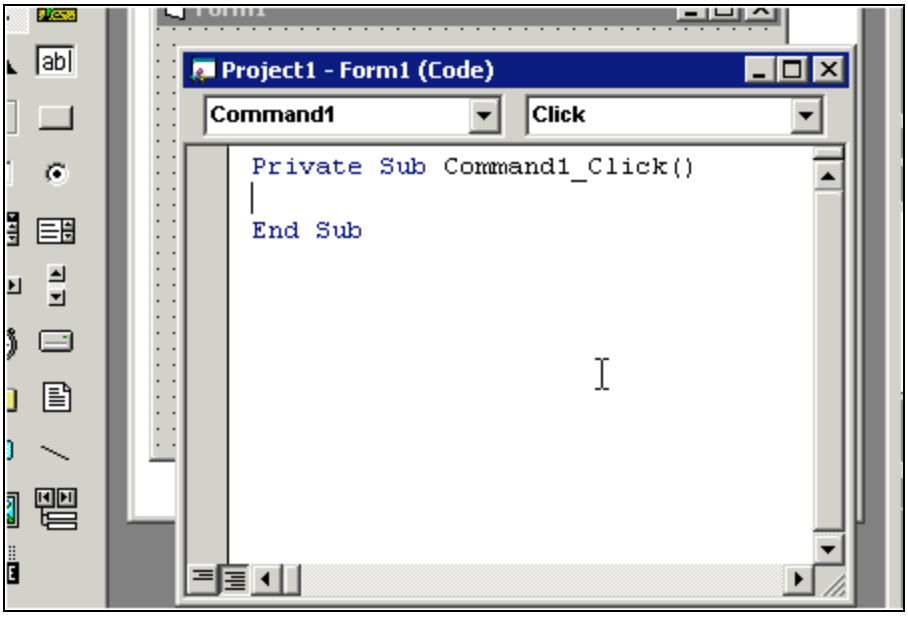
Go ahead and click on that button now.



You can see our program is now running. Here it says **Form1** and **Command1** . Let's click on the button. You'll see that nothing's happening. Why is nothing happening? Well, we haven't told our program to do anything yet when the user clicks on that button. We haven't given this button an **event** yet. So let's see how we do that. Let's go ahead and close this program just like we would close any other Windows program. Notice that we're back in the Visual Basic Design Editor.



In order to tell this button to do something when we click on it, we have to give it an event. We have to do some programming. So let's double click on this button. Take your mouse and double click on it. Notice a code window opens up.

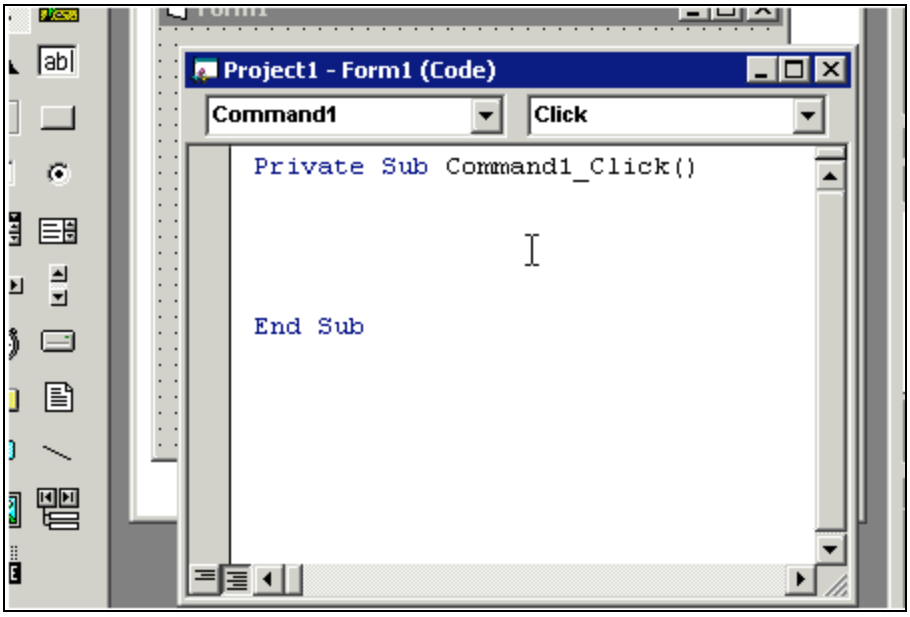


This is called a **code window**. Here it says Private Sub Command1_Click(). What does all this mean? Well for today's class, you don't have to know what all this stuff means. All you really have to know is that everything that happens between the first line (*Private Sub Command1_Click()*) and the second line (*End Sub*) will run whenever you click on that button.

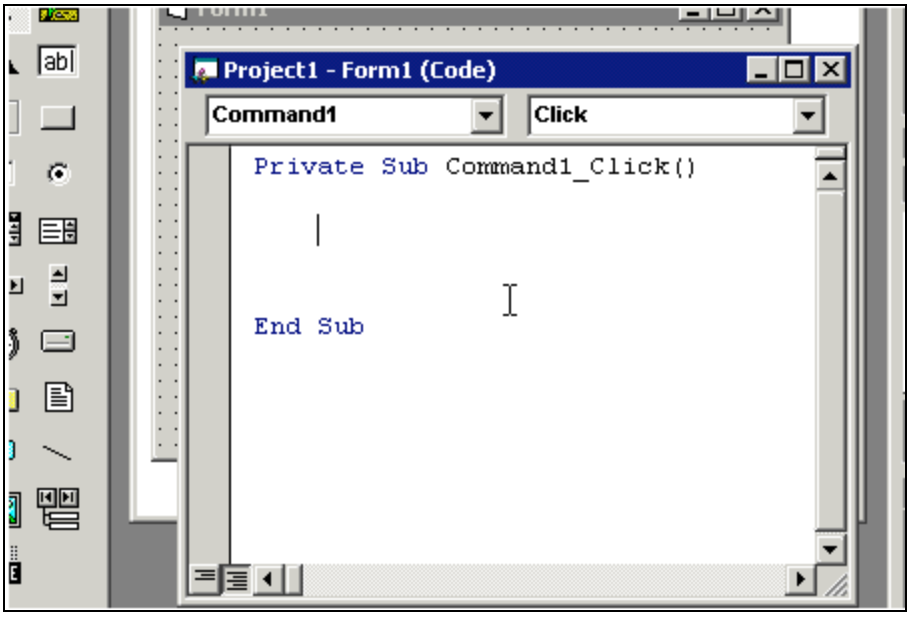
This is a **sub** or **subroutine** that will run when a user clicks on the Command1 Button. Everything that we type in the middle is going to go off when the user clicks on the Command1 Button.

The word **private** just means that the subroutine is private to this specific form.

Let's put an actual command in here and see the program do something. To give yourself some extra room, hit enter a couple of times on your keyboard. Now you have plenty of blank space. Click back up somewhere in the middle



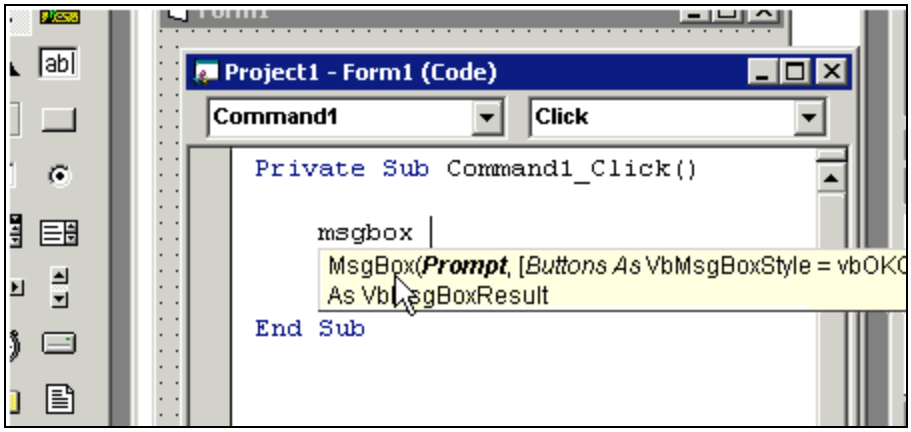
Tab in with the tab key (press the tab key). That's going to move your little blinking cursor over to the right a little bit. It's just considered good programming form to indent your commands inside the private sub.



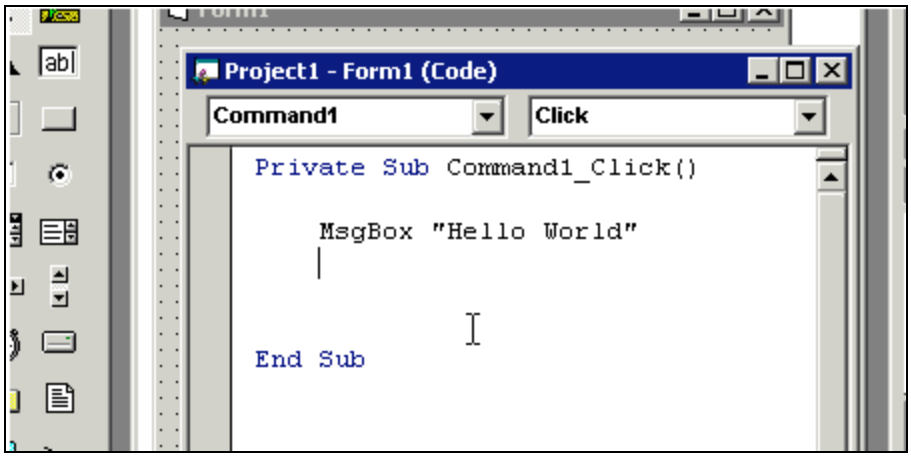
Do you have to do this? Do you have to indent your code inside the subroutine? No. You don't have to. It's just when you build more complex and more complicated programs – the code can sometimes get difficult to read, so it makes it easier to read if it's indented. But you don't have to do that if you don't want to. Now you're going to get your first command. Type this in:

msgbox

and then a space. Notice that a little yellow tool tip pops up.

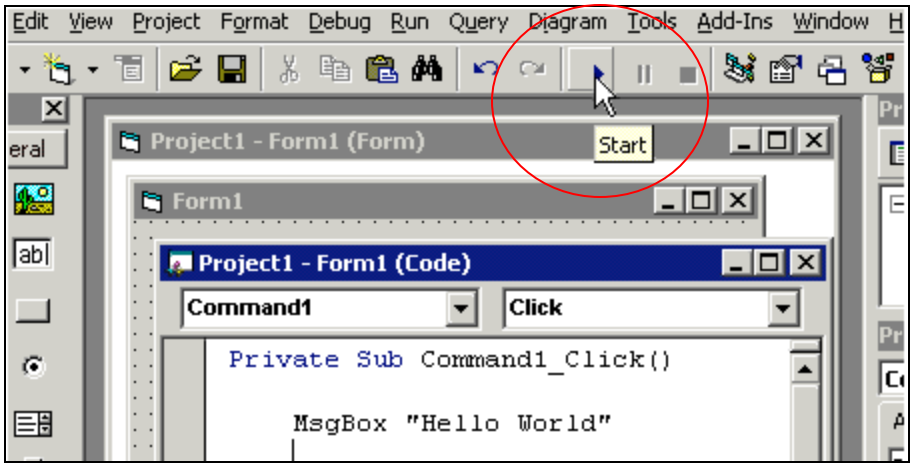


Don't worry about that. It's just giving you some suggestions to tell you what goes next. Next, press open quotes, and then **Hello World** and then close quotes, and press enter.

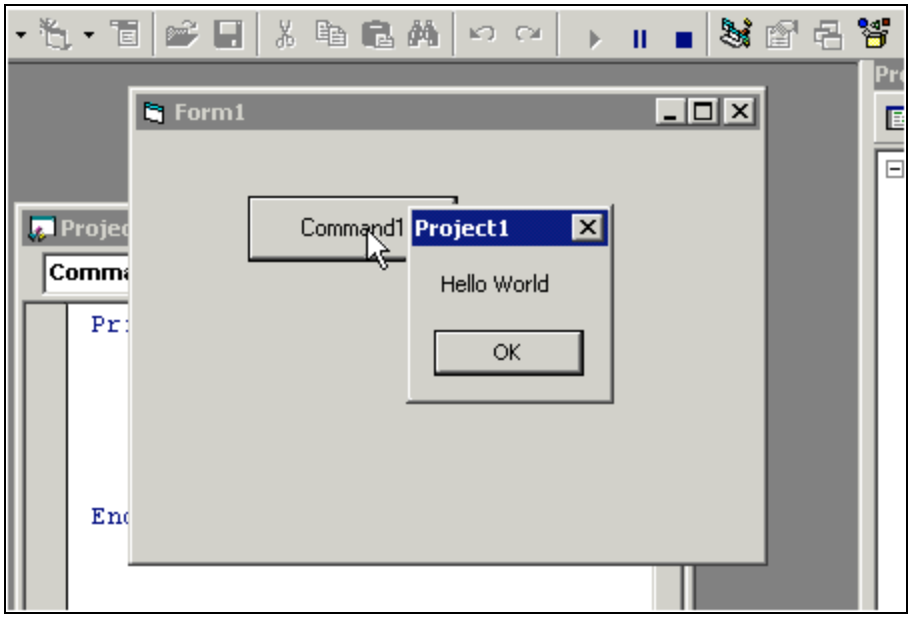


We've now entered in our first command. Essentially, **MsgBox** just pops up a message on the screen. And this specific message box is going to say "Hello World" whenever the user clicks on this button.

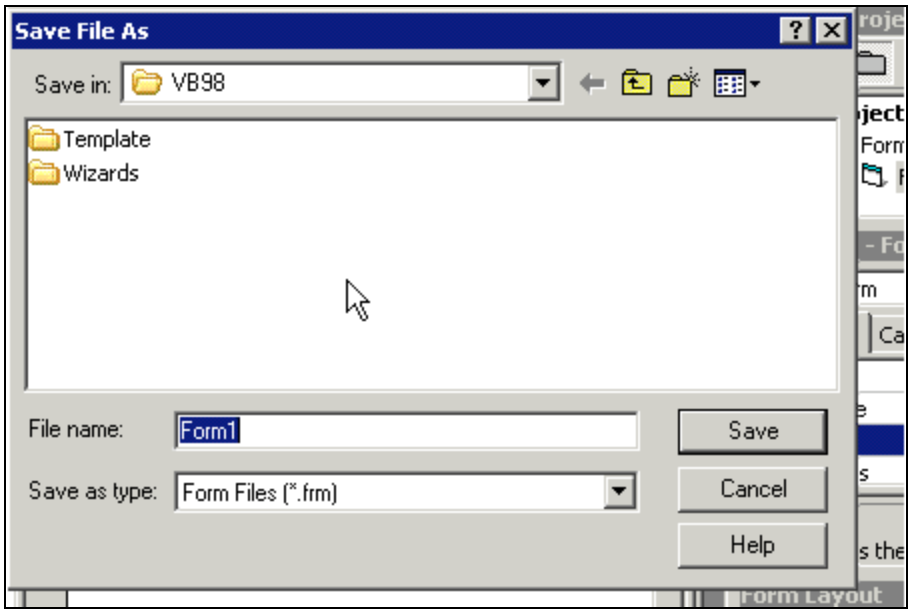
Alright. Let's give it a try. Click on the start button and see what happens.



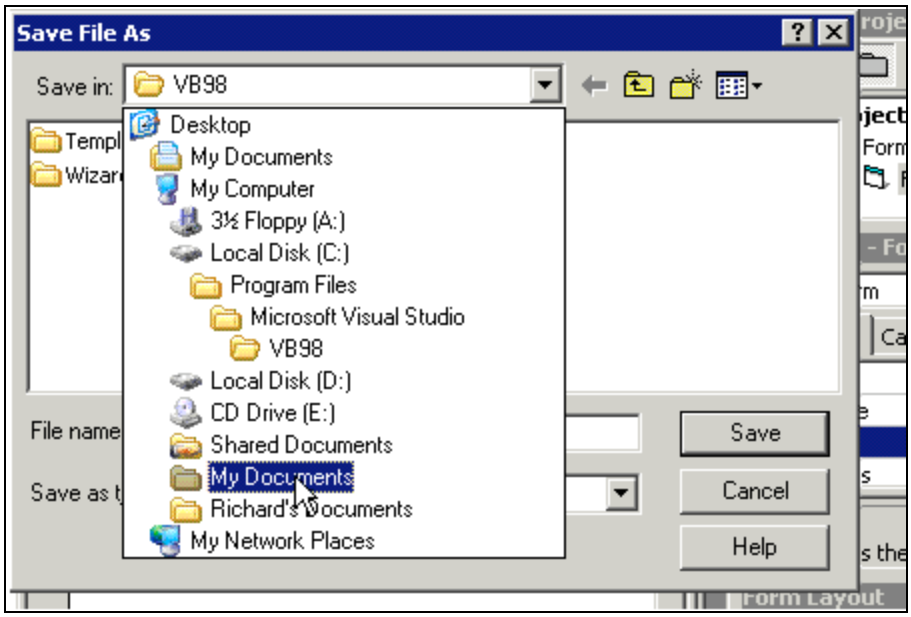
Here's our program. Our program is running now. Let's click on the button and see what we get.



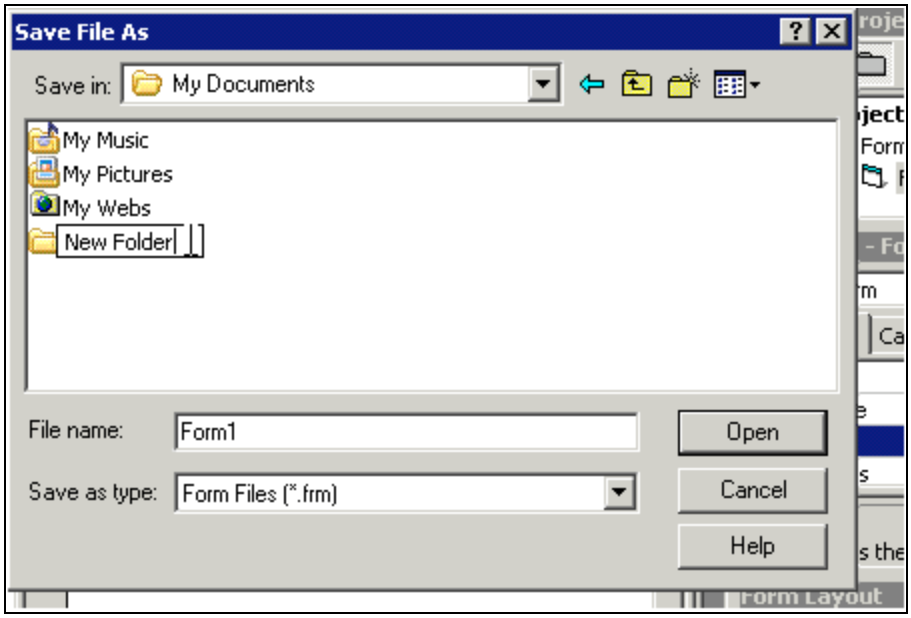
And there it is! There's our message box that say's "Hello World." Click on the OK button. That will close the message box and put you back into your program. Congratulations! You've just written your first Visual Basic program. Give yourself a round of applause (just not too loud cause then the people around you will think there's something wrong with you.). Let's go ahead and close this. You'll return to the Visual Basic Editor. Let's go ahead and save our work. Let's click on the floppy disk button to save our program.



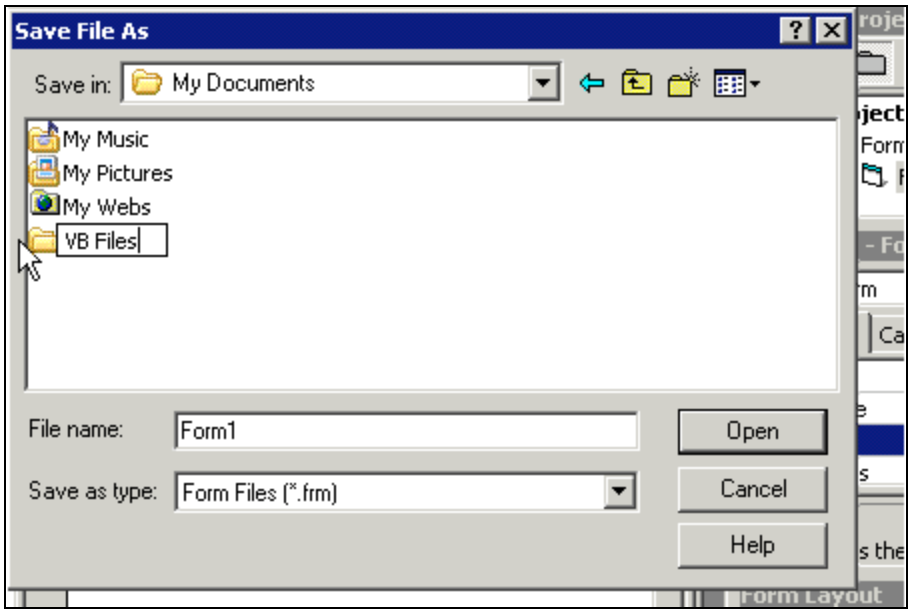
Now for some strange reason, the folks at Microsoft decided to save Visual Basic data files in the **VB98** folder. Instead, save your files in your "My Documents" folder.



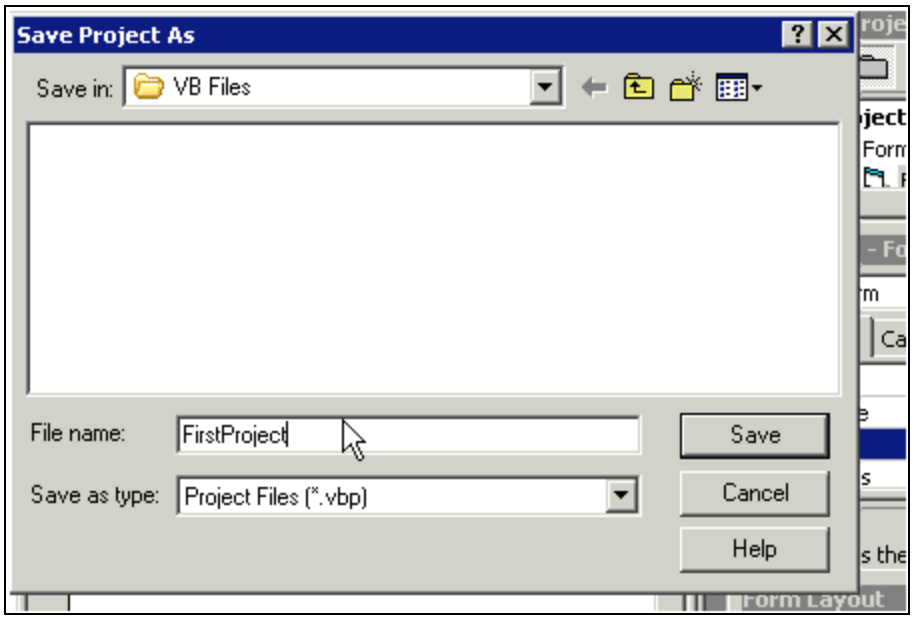
Let's create a folder in here to store our VB stuff. Click on the "Create Folder" button.



And type in "VB Files" and then we can double click in our folder to go into our VB files folder



And now let's save the form. We'll give it a filename called, "**FirstProgram**." This will save the form as an "frm" file. Hit Save. And now we have to save the project file. The project file is like an overall master file that keeps all the information about this particular program in one place. Let's call this one our "**FirstProject**."

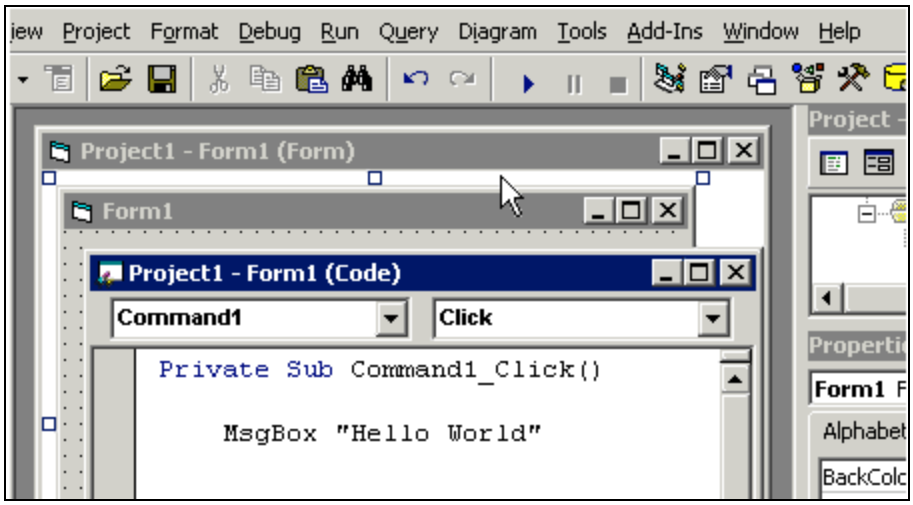


Then hit save. And there we go!

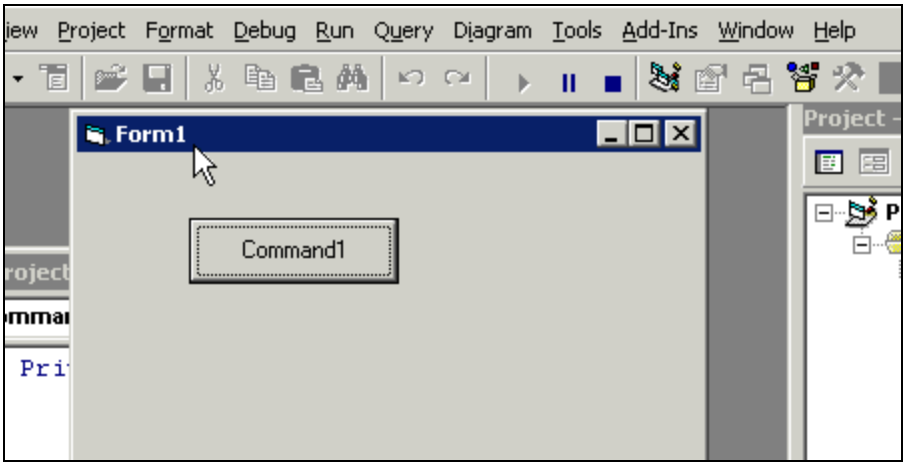
Now you see in just a few minutes, you were able to create your own program, run it, make it do some stuff, and then save that program. You are now a programmer!

Lesson 4. Properties

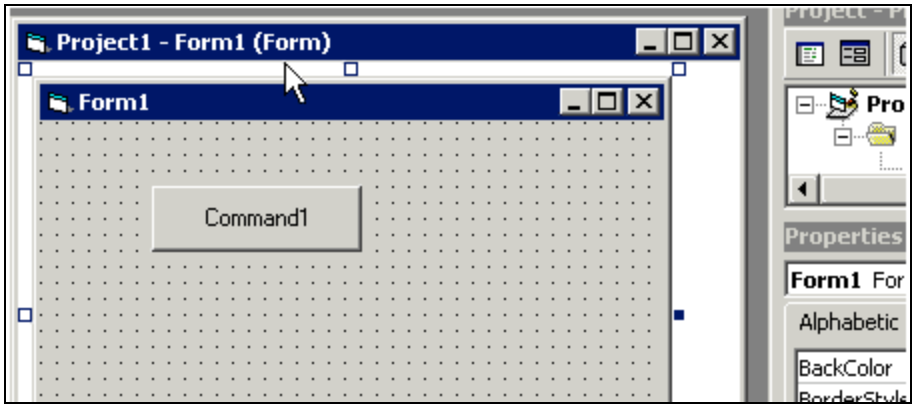
In this lesson, we're going to take a look at some of the properties of the objects we created in the last lesson. Here we are back at our Project1.



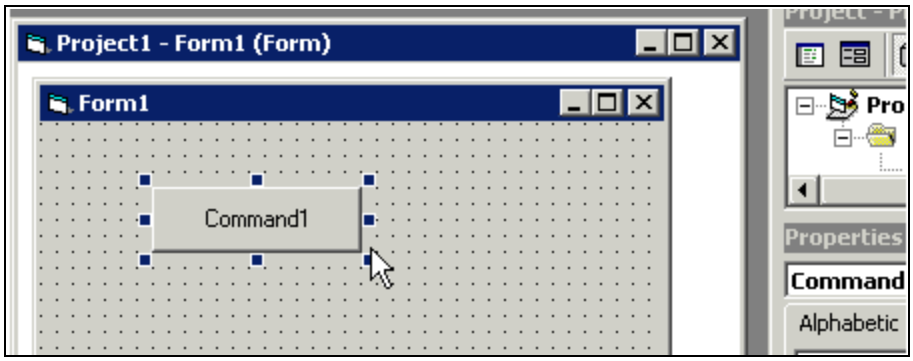
Now when we run this project, we can see we've got Form1 and Command1. Not very informative, is it? It would be nice if we could actually put some stuff on here. Like have Form1 say maybe, "First Project" or "First Program," and have our button say something like "Click Me." Command1 isn't very user friendly, is it? Let's close this.



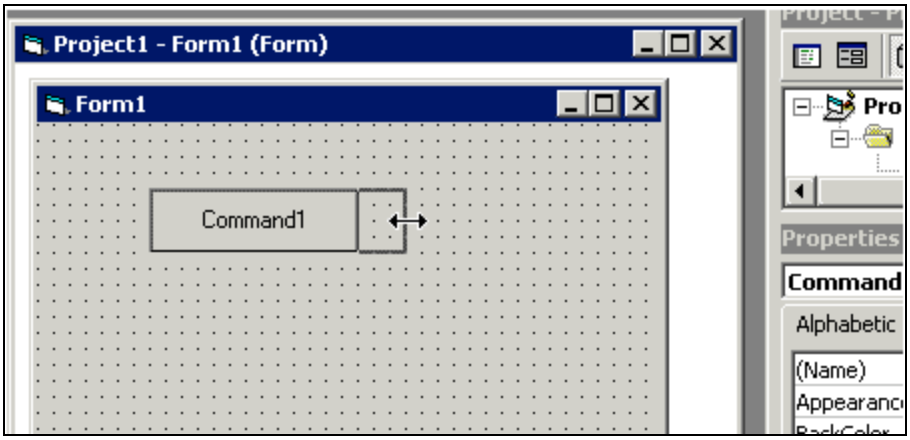
Now to adjust these properties, let's go ahead and close the code window. Now we're back to our form window.



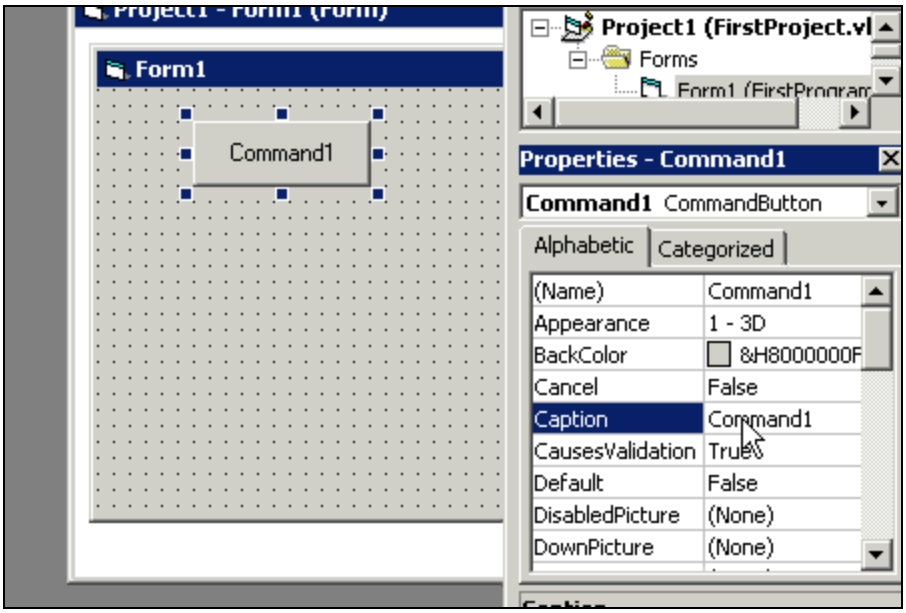
Now here's the form, and here's the button. Let's start with the button. Let's click once on the button. Notice how we get these little dashes and dots around it.



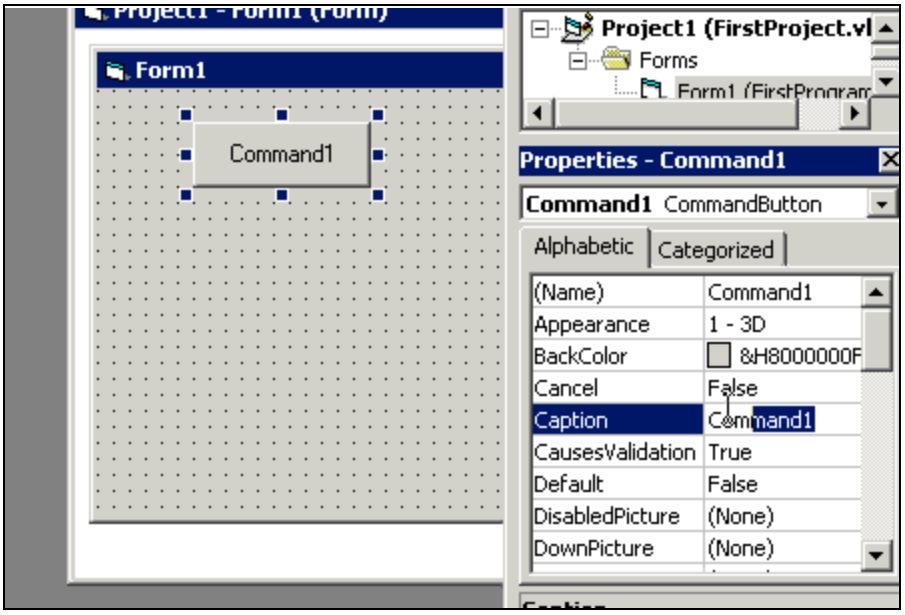
These dots are used to resize the button.



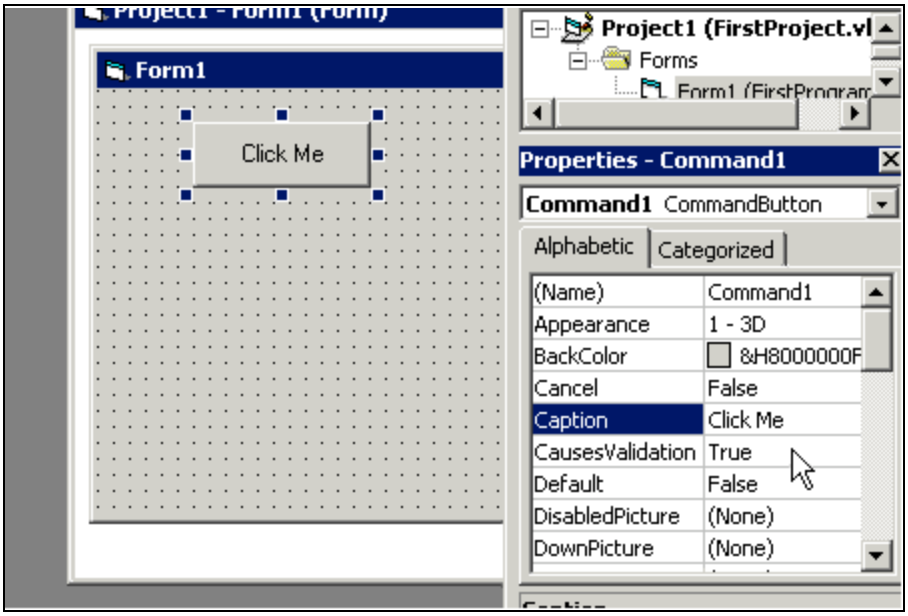
You can also move this button, by the way, by clicking on it and dragging it. But if you look over on the right, you'll see the properties window. And right now since we have Command1 selected or highlighted, it says the properties for Command1. The name of this button is Command1. Look for the **caption property**.



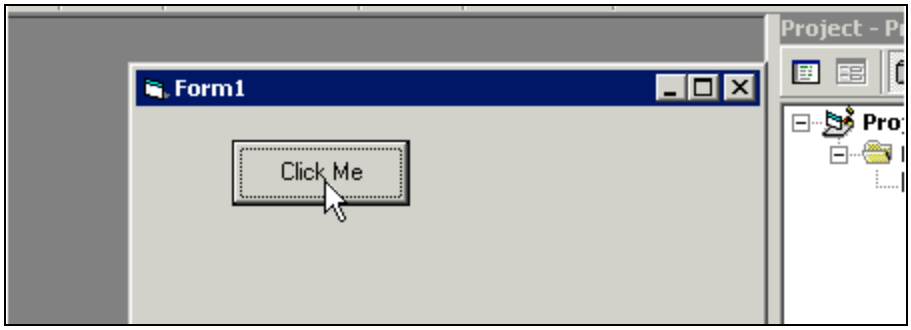
Caption is what actually shows up on the button, and it doesn't necessarily have to be the button's name. This is just one of the properties for the command button.



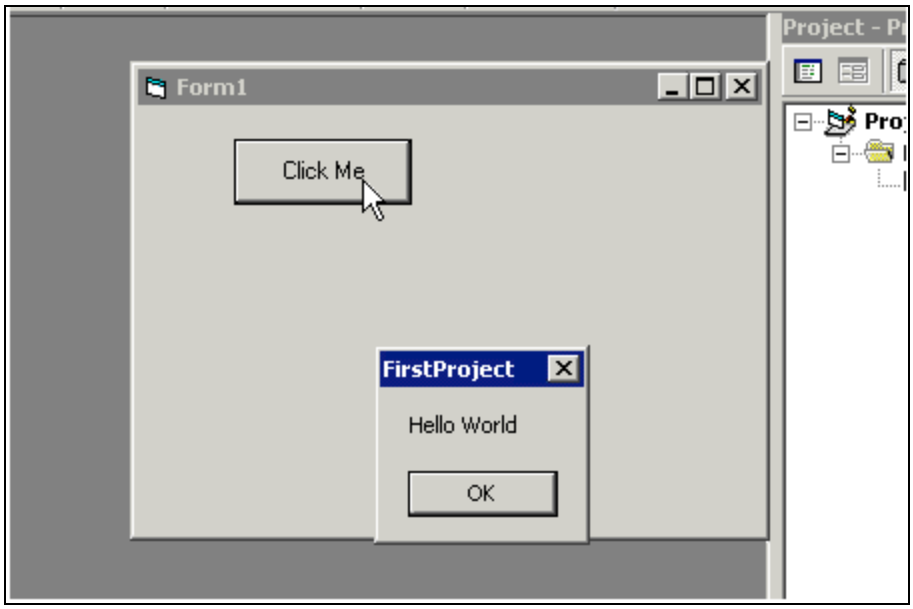
Let's click on the caption and change this to "Click Me."



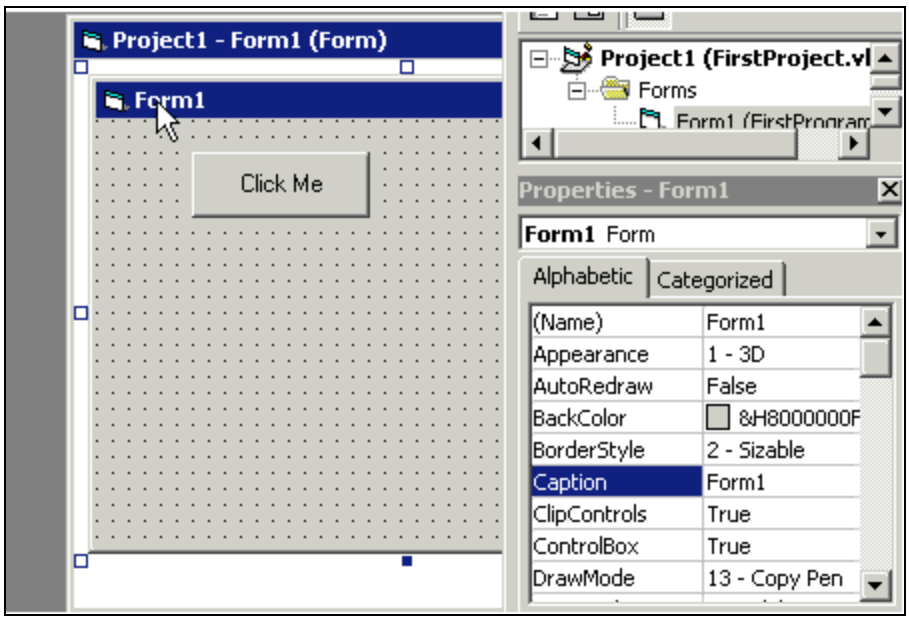
Notice the caption on the button changes as well. The button will say whatever you put in the caption property. That's how the caption property works. Now let's go ahead and run our program. Click on the start button, and look at that! We changed the button so it says, "Click Me" instead of Command1.



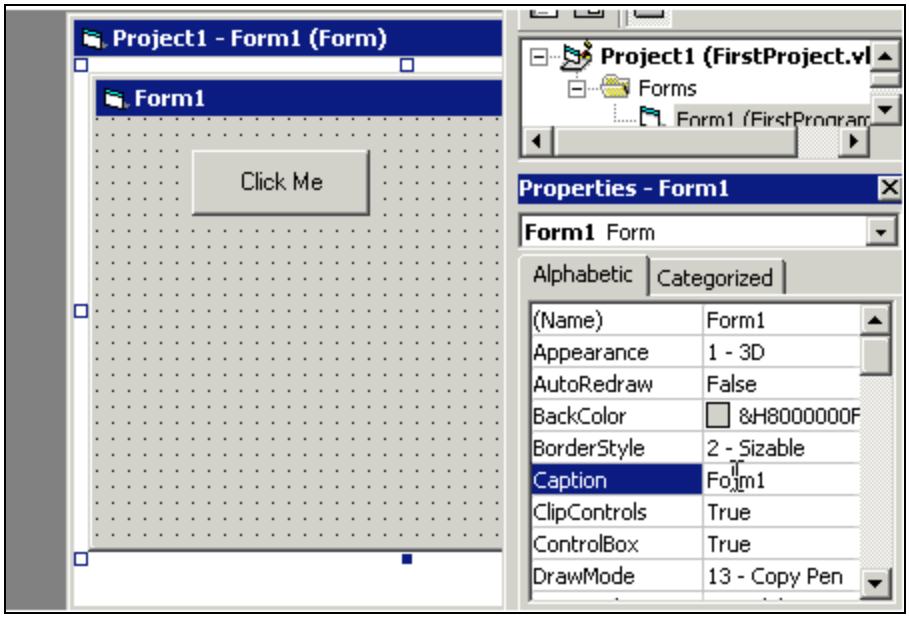
If you click on it, it will generate the "Hello World." We've changed one of the button's properties.



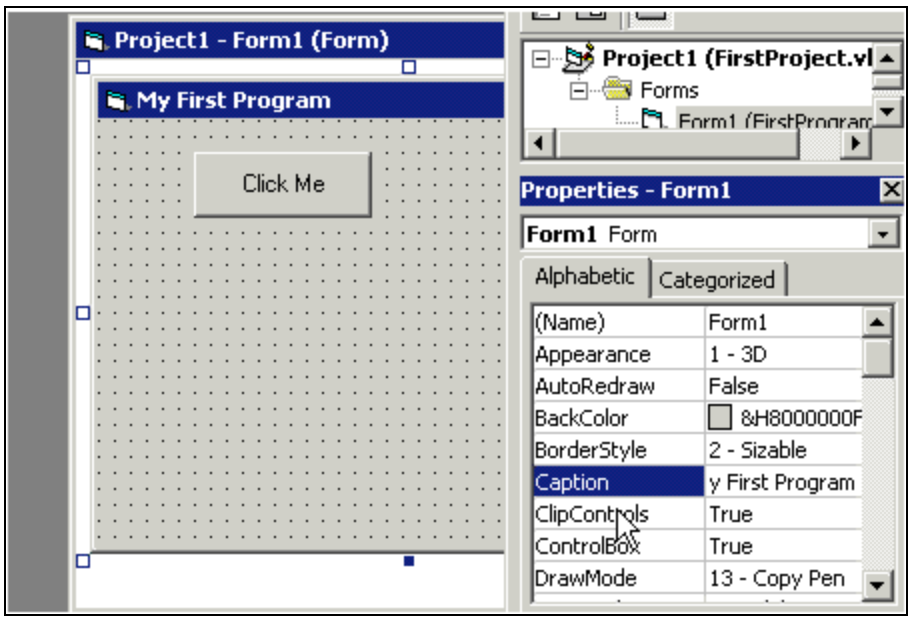
Let's go ahead and close the program. Now what about the form itself? Click on the Form's title.



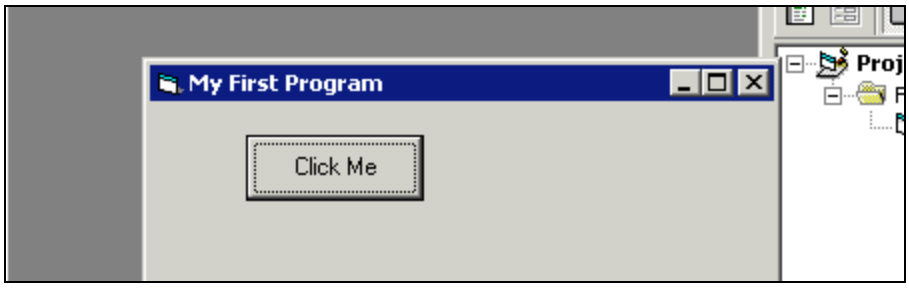
And notice that we have the properties for form one. Remember, almost every object in a Visual Basic program has its own properties. A button has its own set of properties. The form has its own set of properties.



Now you can click on the caption for the form and make this say, “My First Program,” for example.

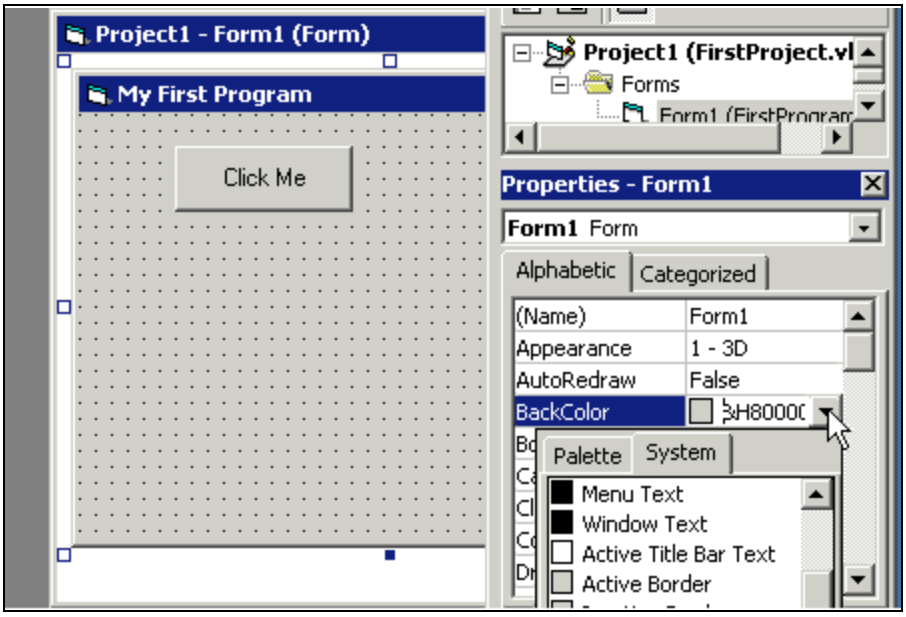


Let’s run our program now.

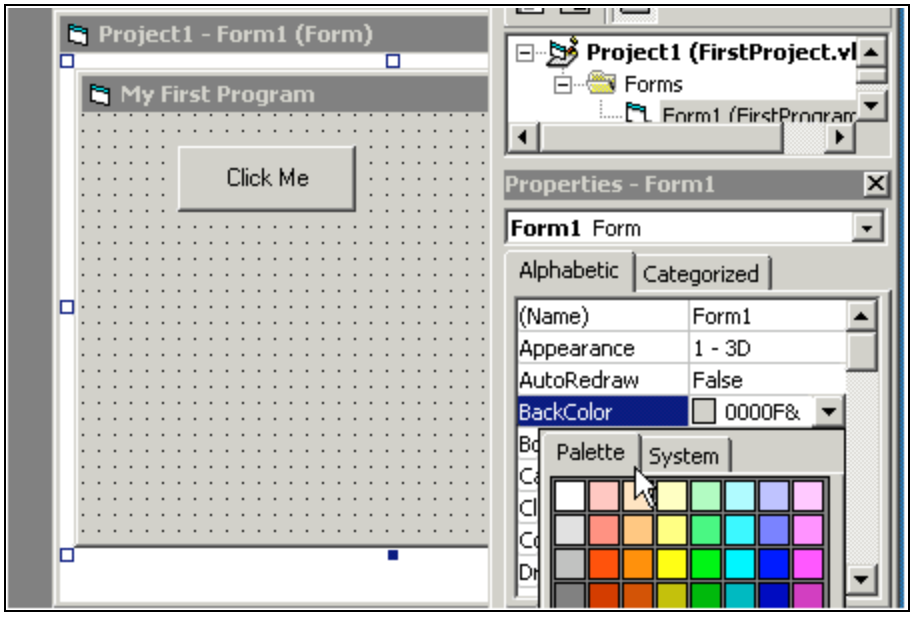


And there we go! It’s a little more user-friendly now. It says, “My First Program” across the title bar. That’s the caption property for the form. Let’s close our program again.

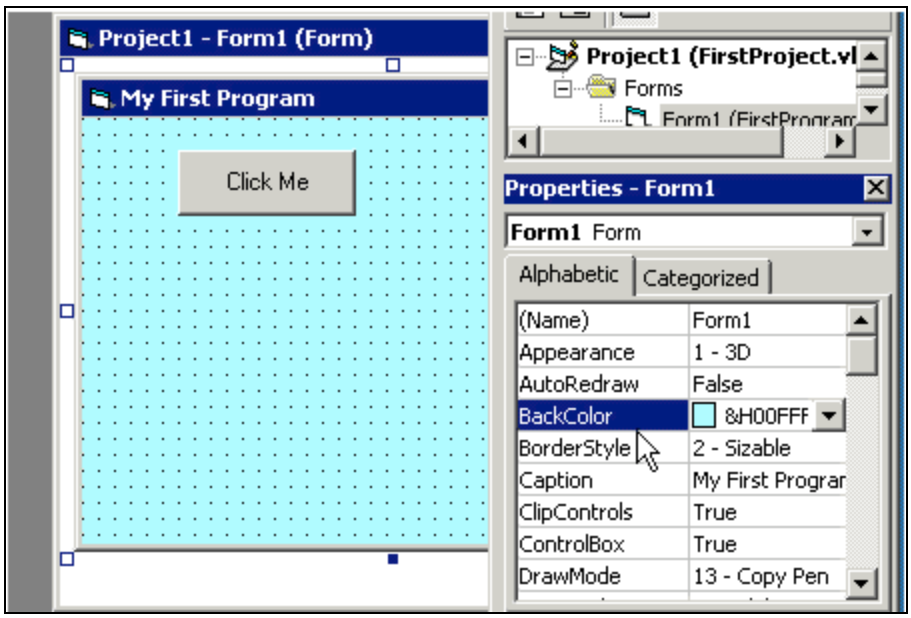
Now there are many other properties for command buttons and forms. One of the other properties for example is back color.



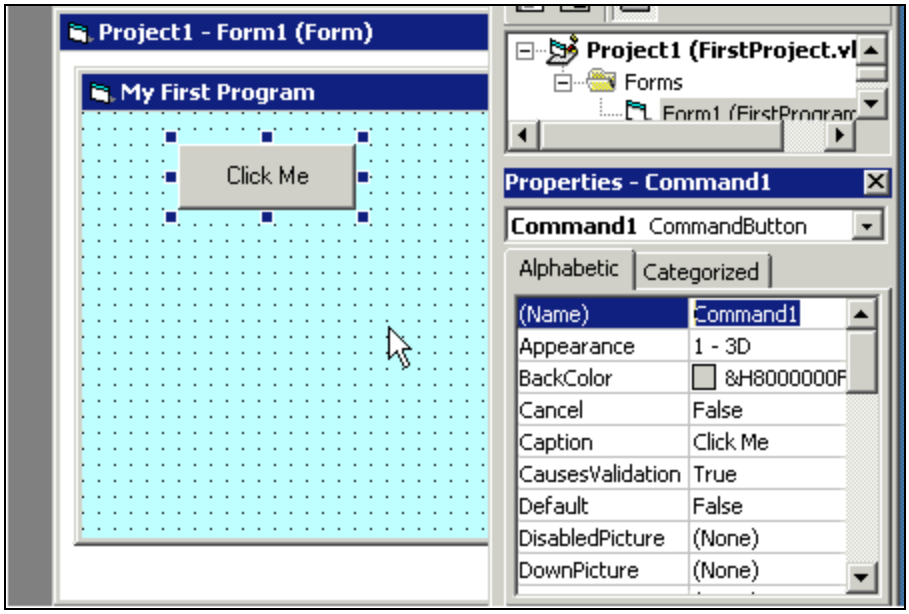
Click over on the BackColor properties and you'll see a little drop down. Click on the drop down and it will open up a list of colors. Select the palette tab and you'll actually get a color palette.



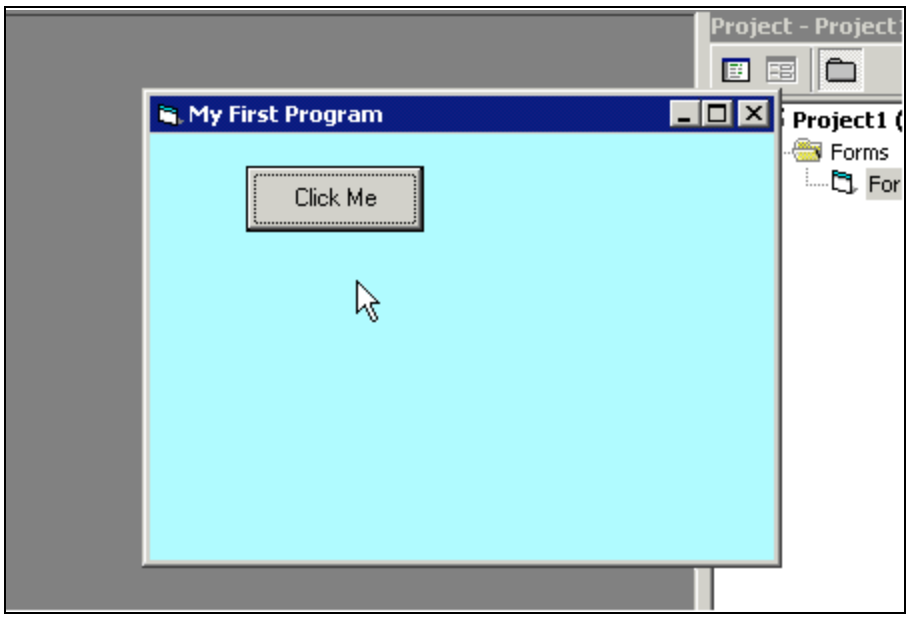
You can click one of these different colors like light blue.



Notice how we've changed the back color or the background color for the form. If you scroll down the properties list, you'll see all kinds of different properties. There's enable, fonts, ForeColor, icon, and more. One of the other properties you can change is the name of the object itself. For example, if we click on our command button, we can actually change the name of this button from Command1 to something else.

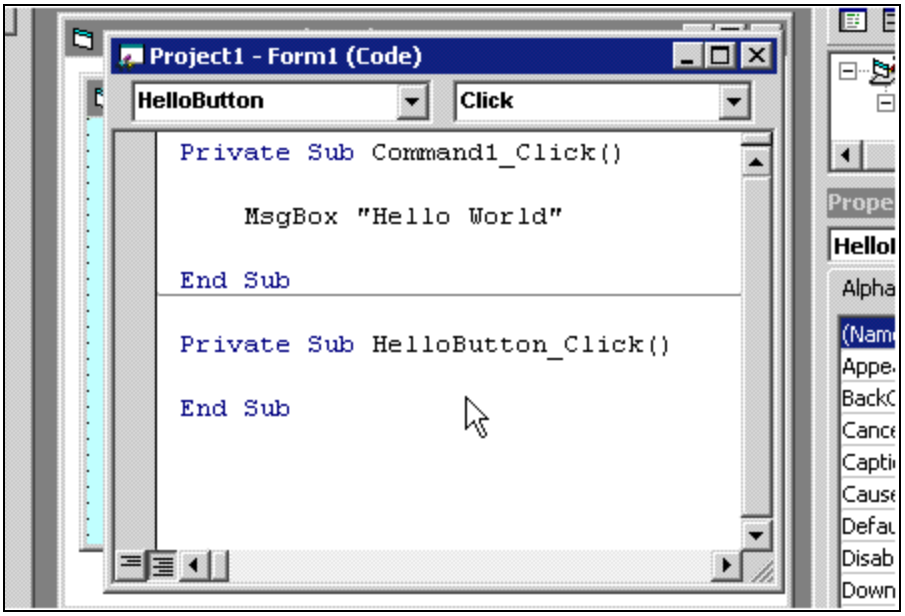


Let's change this button's name to "HelloButton." Let's run our program again.

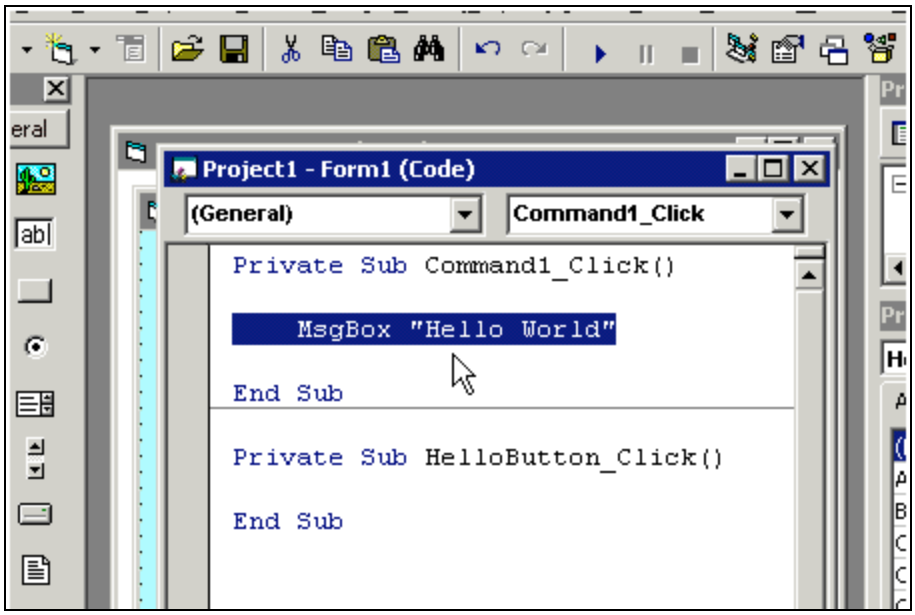


Notice our button says, "Click Me." Click on it. But wait a minute – something's not right. Our button's not working anymore. We just broke our button! Let's close our program and figure out what happened.

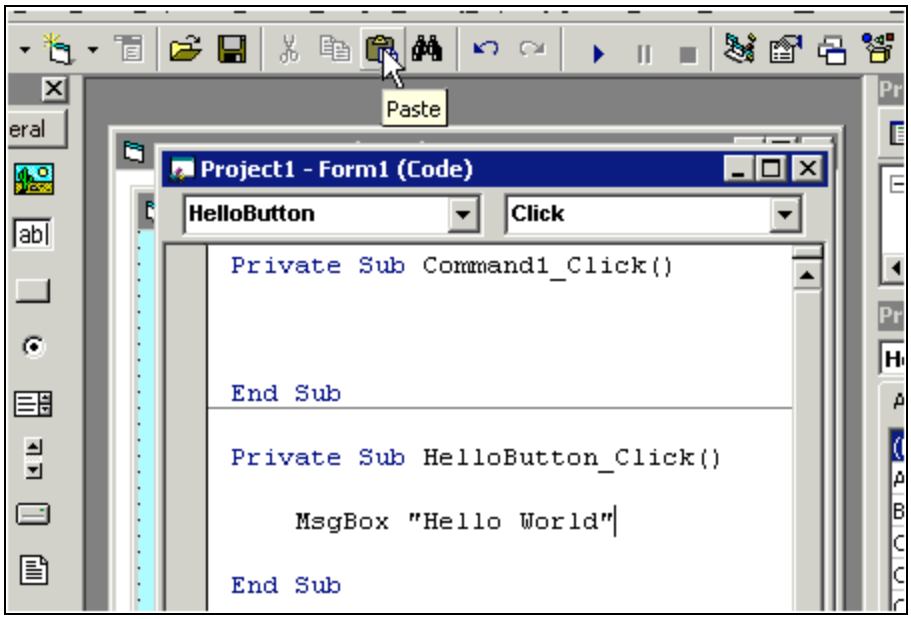
Back in our Visual Basic Design Editor, let's double click our button and check its programming code.



Okay, we double clicked and now Visual Basic put us down in a new private sub! What's going on?! Well you have to be careful if you change the name of an object. We changed the name of the button. We renamed it from Command1 to HelloButton. When you do that, any programming code that happens to be in that object is lost. It's still there as you can see, but Visual Basic loses track of it. All you have to do, is take the code on top, cut it out…



and then paste it below.



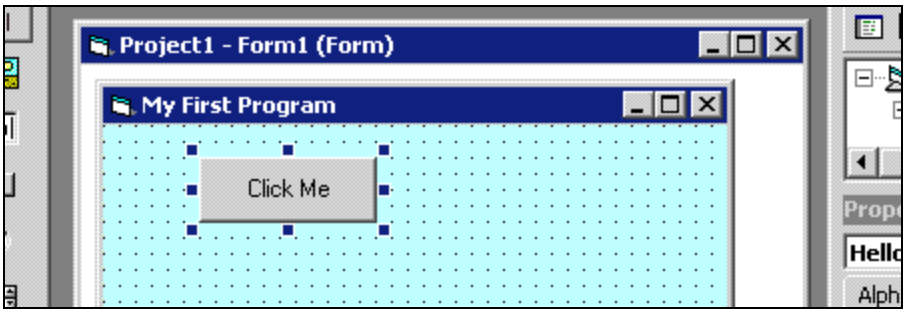
And there we go. And we can delete the extra lines of code by simply highlighting them, and hitting delete on our keyboard.

So keep in mind that if you change the name of your button, you've got to change your code too.

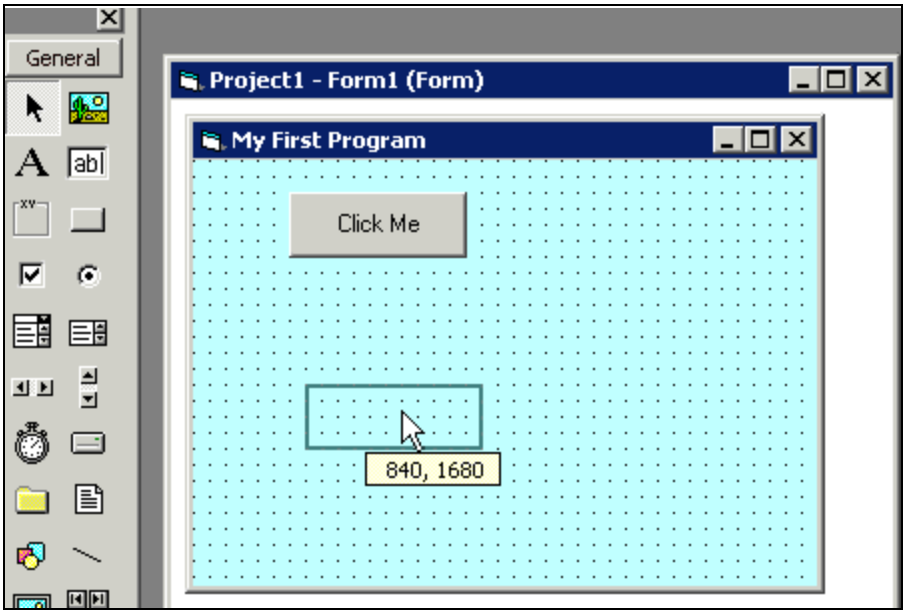
Now we've made our first program and we've learned how to change some of the properties of the different objects in our program: our command button and our form's properties.

Lesson 5. Text Boxes and Labels

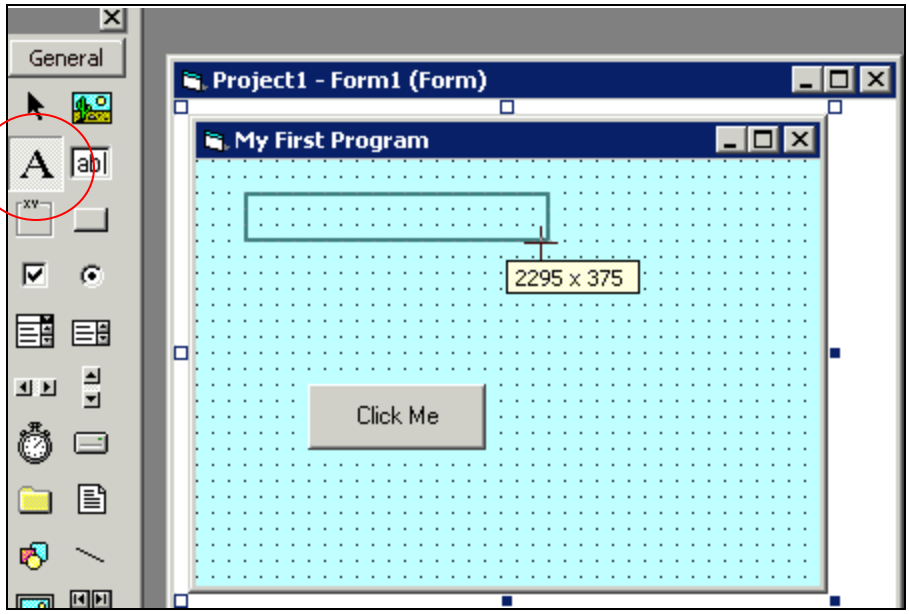
In this lesson, we're going to learn how to make our program a little more functional with some new controls. Specifically, we're going to learn about **text boxes, and labels**.



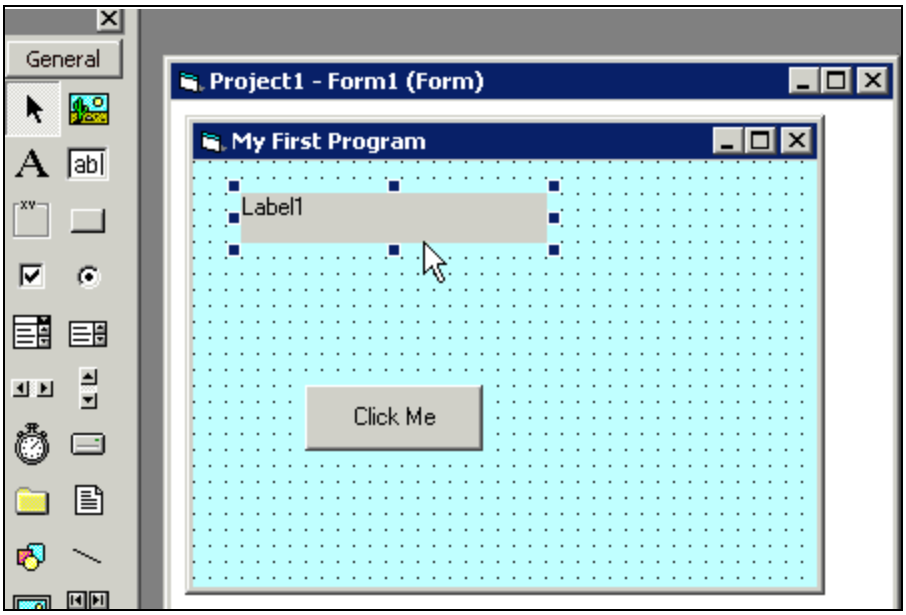
Here we are, back in our good ol' little program. Let's now add some additional **controls** to this program. Let's add a **label** and a **text box**. A **label** is just used for displaying some information on your form. You can put words in there, numbers… whatever kind of text you want to have. A **text box** is used to actually store data. Let's see how these work. First, let's move the little "Click Me" button down a little bit. Click on it and hold the mouse button down and drag it down.



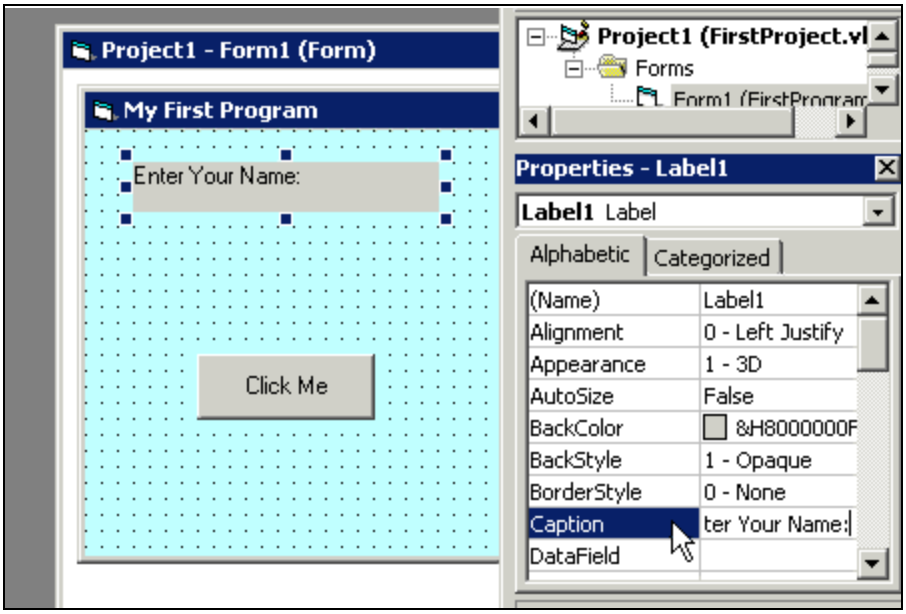
Now grab a label. Click on the "A." Then come on over to your form and click and draw a box.



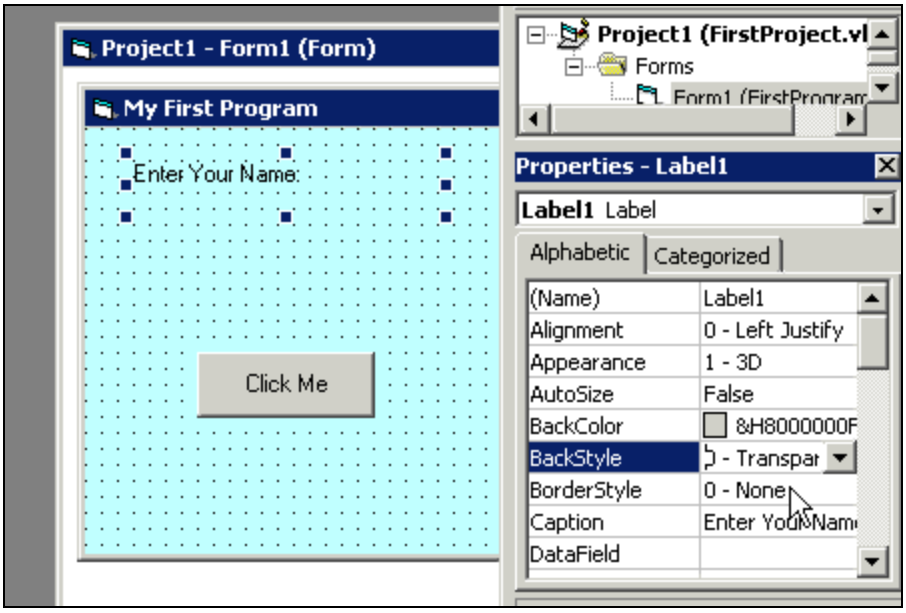
Let your mouse go. And now we have an object that says "**Label1.**"



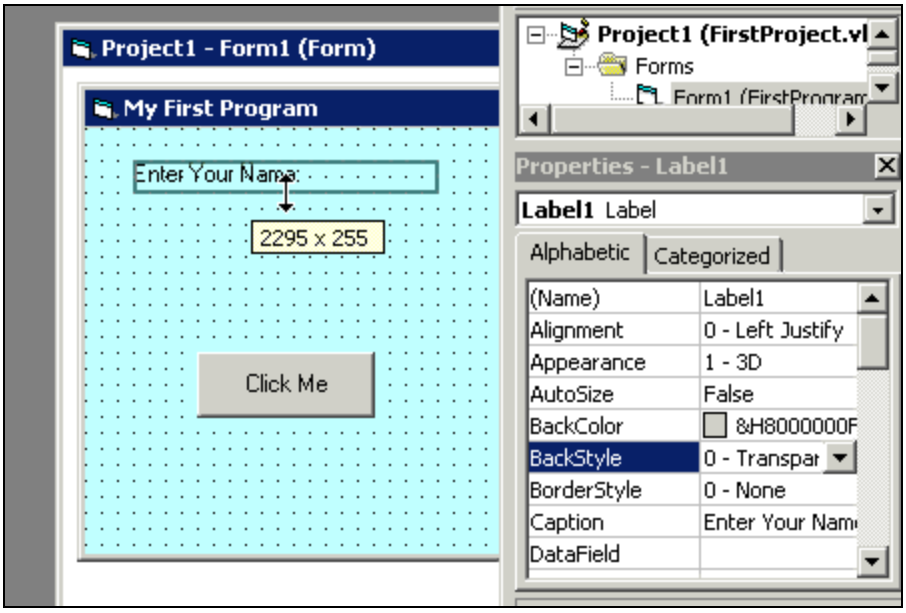
In the properties for **Label1**, change the caption so that it says, “Enter Your Name” and then a colon.



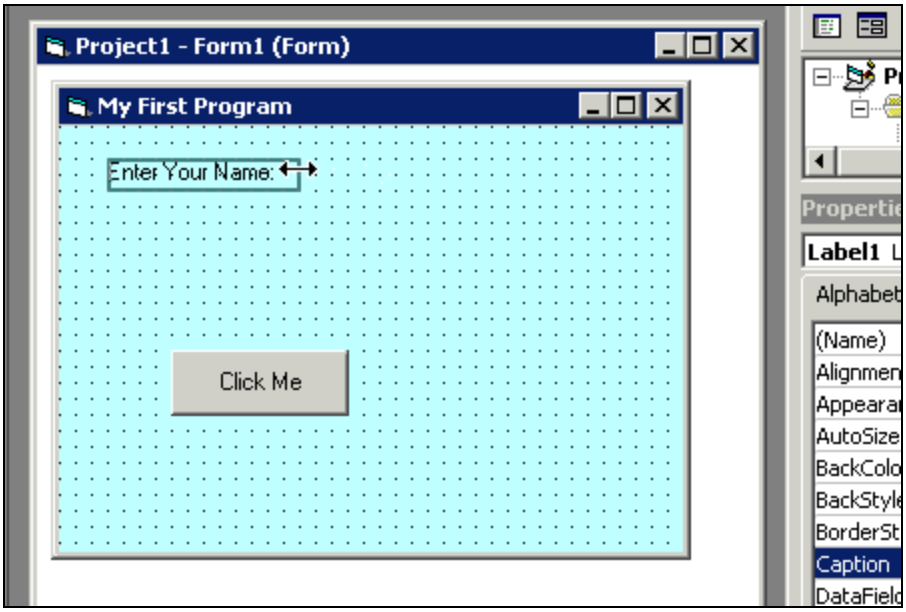
That’s basically just a message that you want to give to the user. Now notice how the background of the label is gray. That’s because it’s specified in the properties box. We could change the back color of this label to match the background color of the form. But then if we change the form again, we’ve got to change the label. So an easier thing is to change it from opaque to transparent.



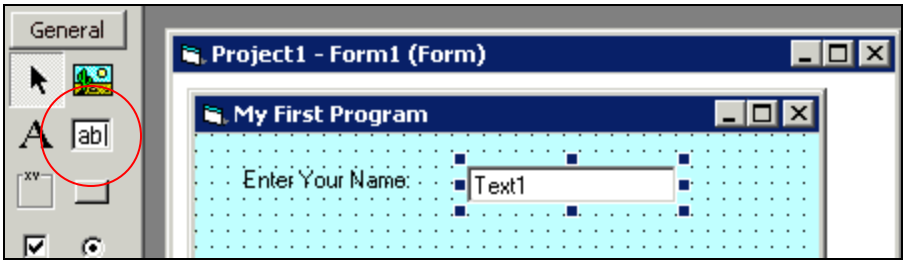
Notice how that happened. The BackStyle changes from opaque to transparent and now the background of the label is transparent. So if we change the color of the form again, we don't have to keep changing all our labels.



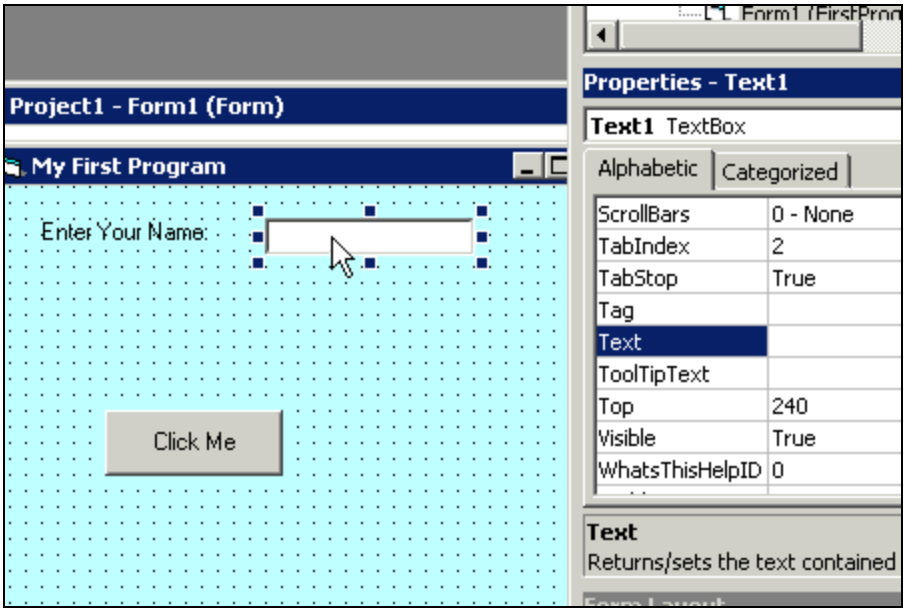
The label is a little big. So grab the box from the bottom and shrink it just a little bit. Let's also resize it this way.



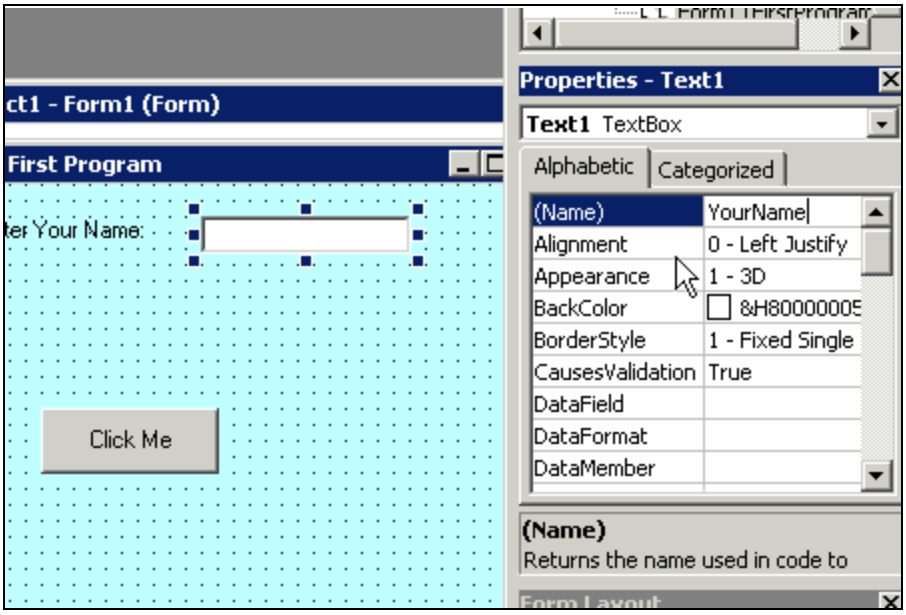
Let's put a text box next to it so that the user can type in their name. Click the "ab" text boxbutton on our toolbox. Then draw a box out on our form.



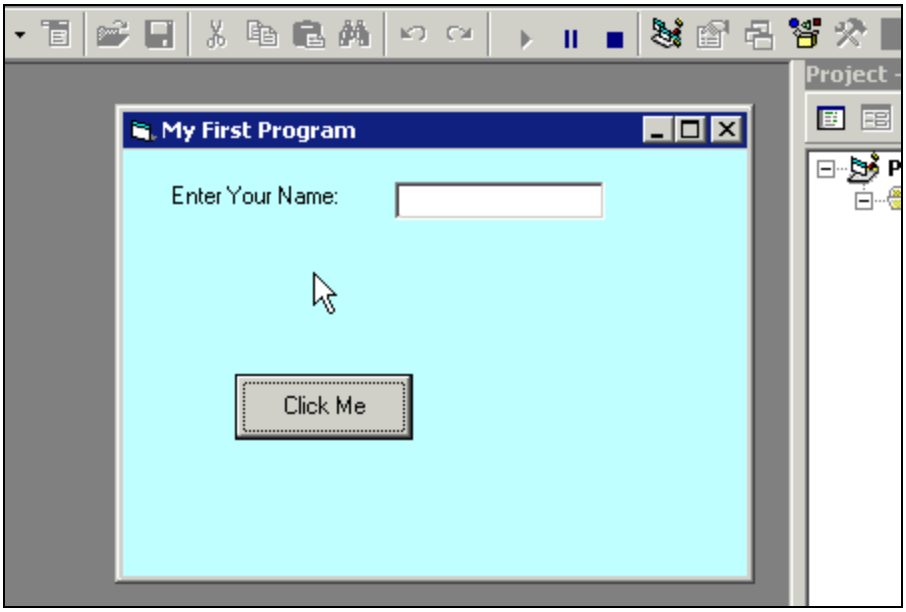
There's a text box. Text boxes have a property called, "Text." By default, it says "Text1." Let's highlight it and delete it. That way our text box will start blank.



Also, scroll up to the very top of the properties list and while we're at it, let's change the name of the box so that instead of saying "Text1," let's say it says, "YourName."



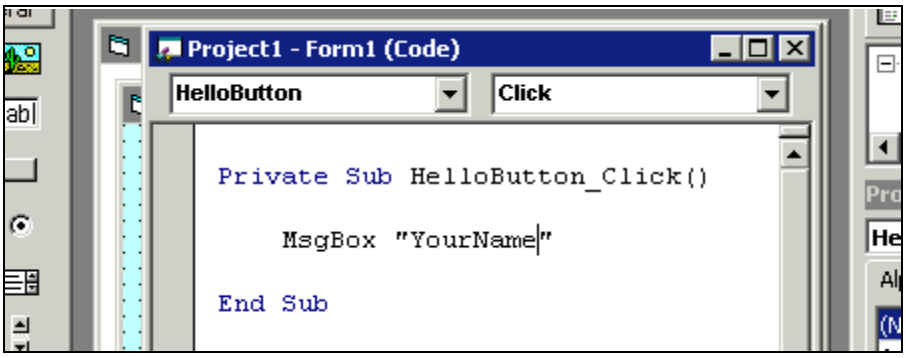
Now this is the actual name of the box itself. The box (the object itself) is now called, "YourName." It's important that you don't put any spaces between "Your" and "Name." So let's see what we've got so far. Click on the start button.



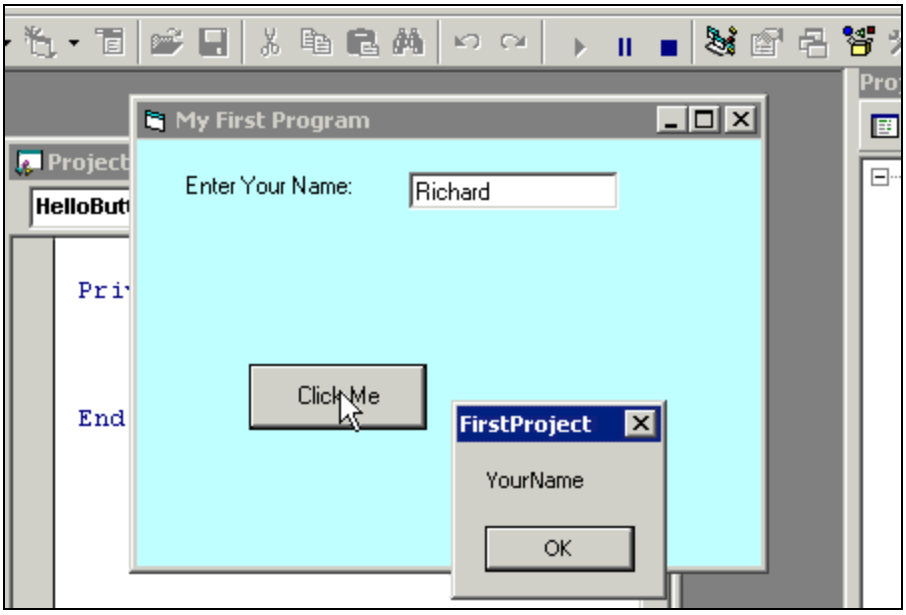
Notice how we have our label, “Enter Your Name:,” and a text box. You can click in this text box and type in your name. And you can click on the “Click Me” button if you want to, and it still says “Hello World”.

Now, wouldn’t it be nice if that box could say, “Hello (your name)” instead of “Hello World”? Let’s see if we can do that. First, let’s see if we can get the message box to just say your name. Let’s close our program.

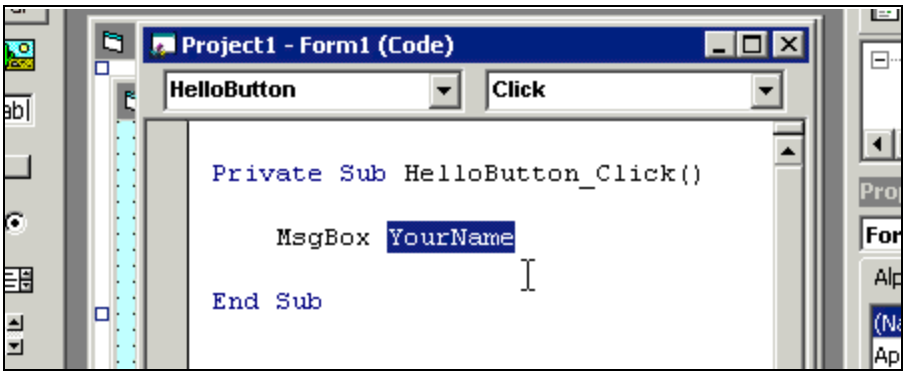
The name of the text box is “YourName.” So instead of having the message box print, “Hello World” on the screen, let’s have it print whatever is in the “YourName” box. Double-click on our “Click Me” button and that will open up our code window. Let’s change “Hello World” so it says “YourName.”



Let’s see if we get what we want. Click on the run button. Type in your name and hit the “Click Me” and wait a minute!



It say's "YourNa me" in it – why is that? Hmm… let's take another look at our code. Close the program. Our code says, "YourName" in quotes – oh wait a minute! That's what's happening! The message box command is taking the word "YourName" and printing it in the box, "YourNa me."

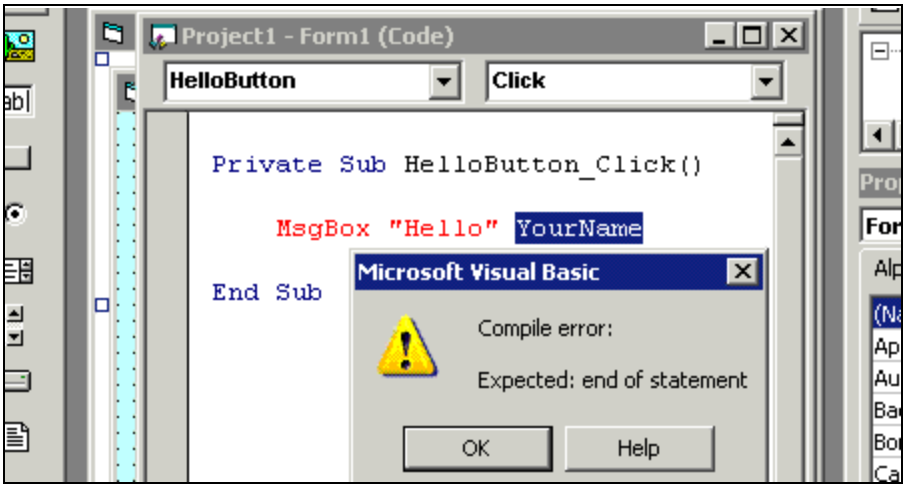


If we get rid of the quotes, now what will happen is MsgBox will now print the **value** that is in the YourName box. Let's see how that works. Click on the start button again. Type in your name. Hit the "Click Me" button. Ah - there we go!

Now the message box command is 'message boxing' your name. What is your name equal to right now? It's equal to your name. And that's how we can use the message box command and a command button to display information on the screen – whatever information a user has typed in. Let's close our program.

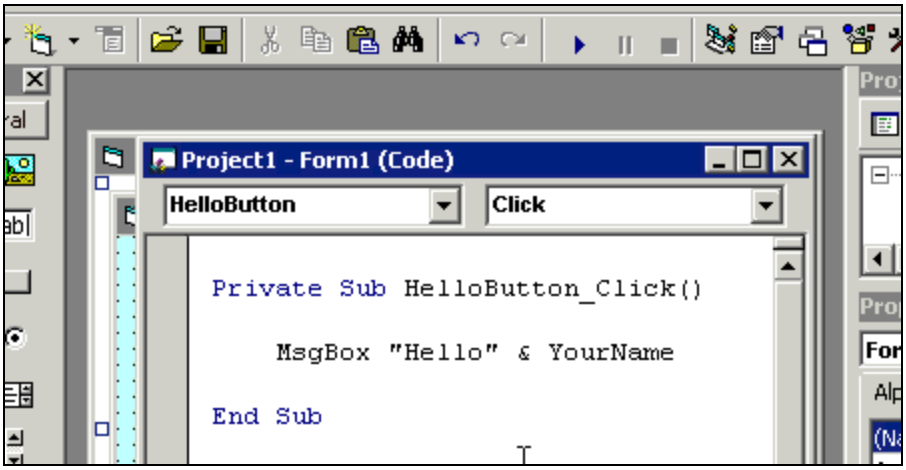
"YourName" (without the quotes) is now a **value**. If it's inside some quotes, it's just going to print the word "YourName" in the box. Now, it's going to send the value for YourName to the message box function instead.

Now we're still only half-way there. Because if you recall, we want this button to say, "Hello (and then your name)." Well here's how you do it. Right in front of YourName, type in open quotes, Hello, close quotes and space. Let's see if this works. Let's run our program.

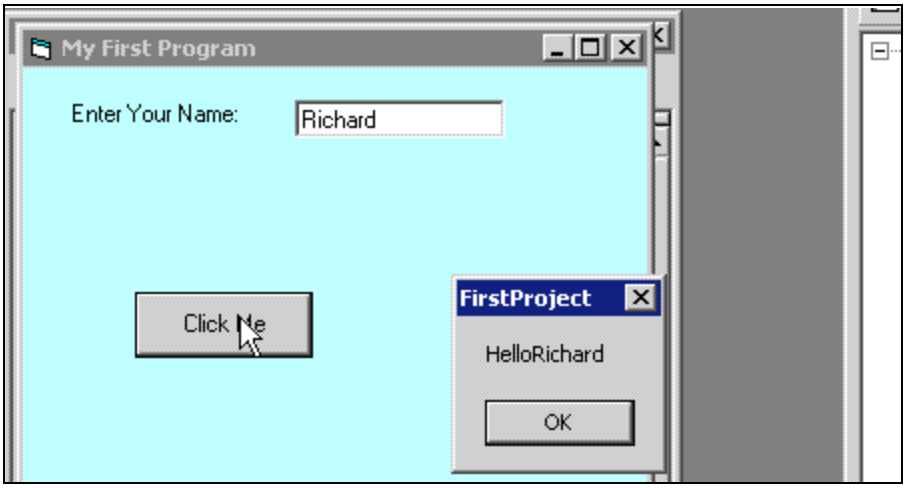


Whoa – what's going on here? Compile error, expected end of statement. **Compile error** means that there is a syntax error in our program somewhere. A **syntax error** means that essentially, we didn't write a line of code correctly. **Expected end of statement** means that Visual Basic expected the statement to end after the word, "Hello." Something's not quite right. Let's click on OK to close this error message. Notice how MsgBox "Hello" is in red. If Visual Basic finds any syntax errors in our code, it's going to highlight them in red.

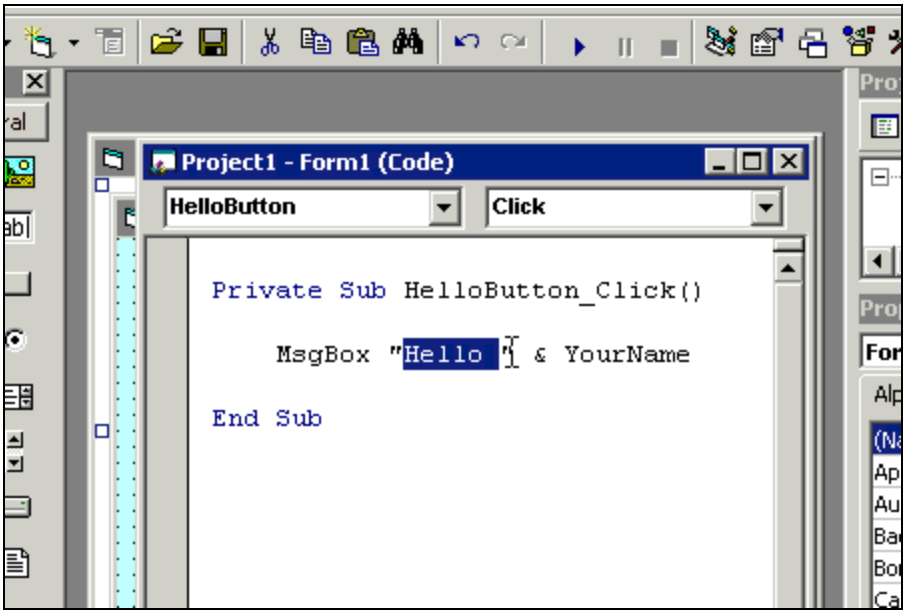
What's happening here is that we've got some text in a string of characters. A **string** is just a fancy word for some text inside of quotes.



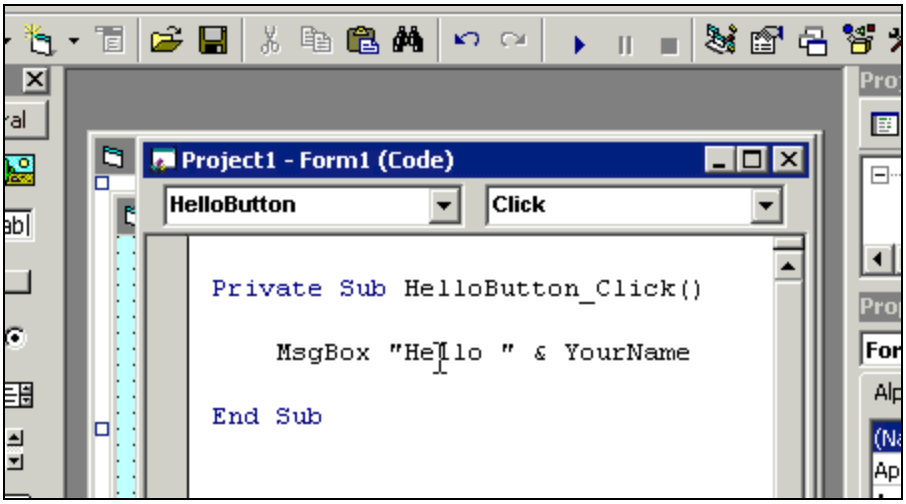
We need some way to put them together. Well, here's a special command for you: Use the ampersand (**&**) to put together "Hello" & YourName and then send them both to MsgBox. Let's see how it works. Let's click on the run button. Type in your name and click on the "Click Me" button.



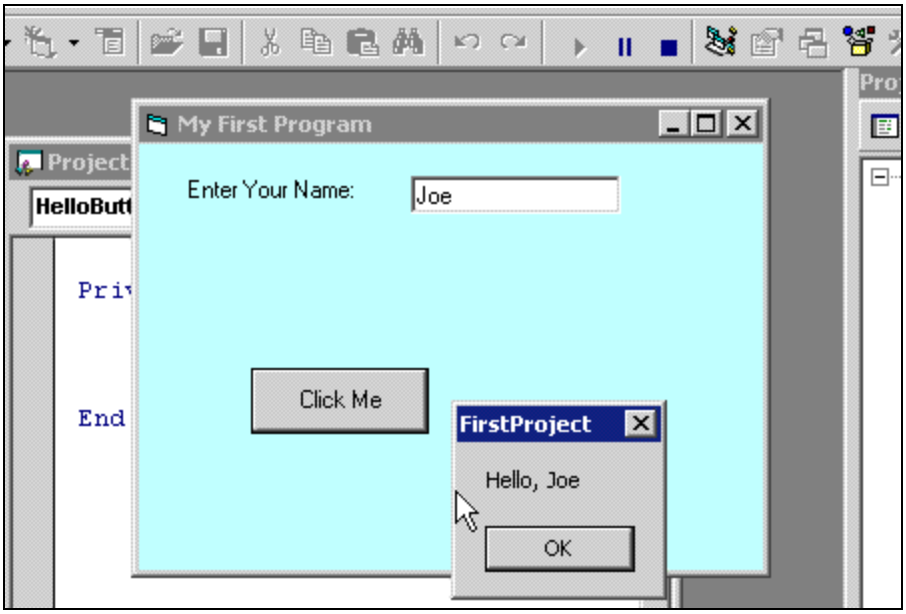
Now it says, "Hello(and then your name)." It took the text "Hello, " and concatenated (that's just a fancy word for putting two text strings together) your name – the value from our box. But notice there's no space between them. How do we get a space in there? Let's close our program and stick a little space between the letter "o" and the closing quote.



Remember, whatever you put inside those quotes, gets put on the screen. If you want to put a comma in there, you can put a comma in there too.



Let's run it. Type in Joe this time. Click on "Click Me."



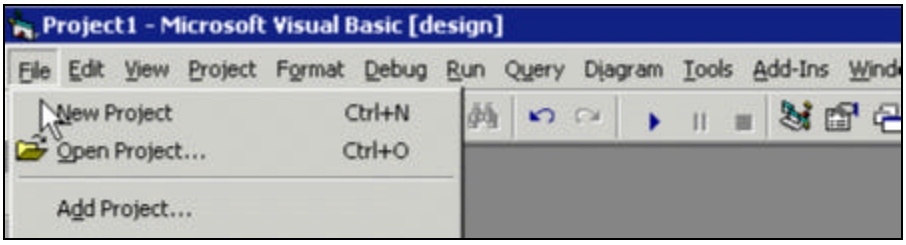
Now it says "Hello, Joe" with a space!

So in this lesson we learned a lot. We learned how to put a label and a text box on the screen. We learned how to concatenate two text strings together. So our program's starting to get a little more functional!

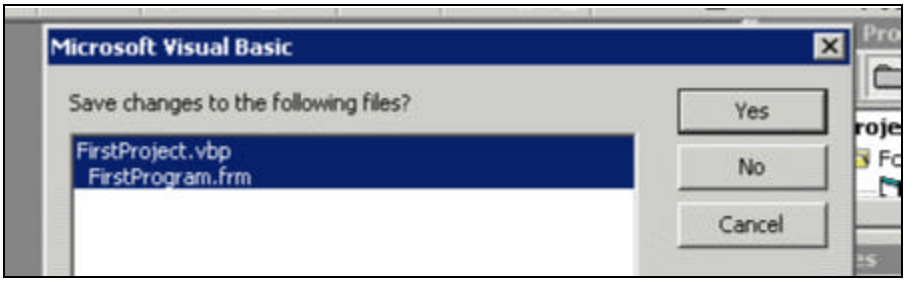
Lesson 6. Simple Calculator

Let's take what we've learned in the last couple of lessons, and put them together to build a program that actually does something useful. Let's build our own simple calculator.

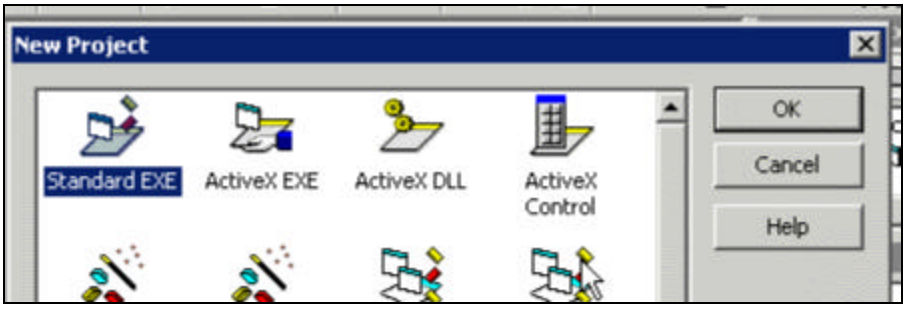
If you already have your Project1 open, let's go ahead and close it. Then click file and then click on New Project.



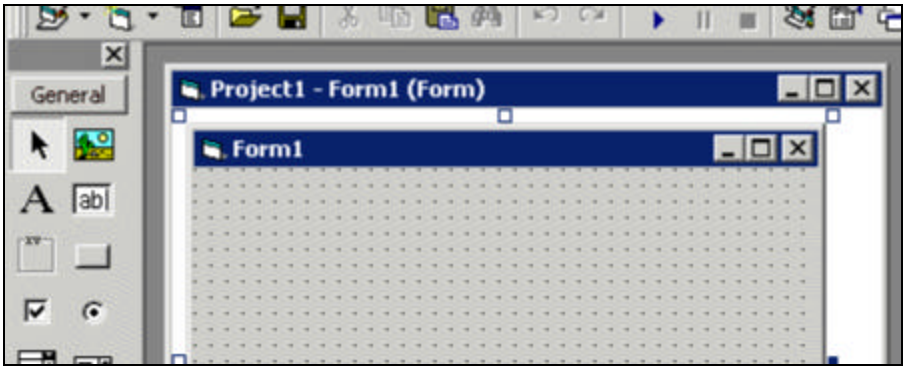
If you haven't saved your changes to the last project you just had open, it's going to ask you if you want to save changes to the following files: our "vbp" our project file, and our "frm", our form file. Go ahead and click yes – that will save those.



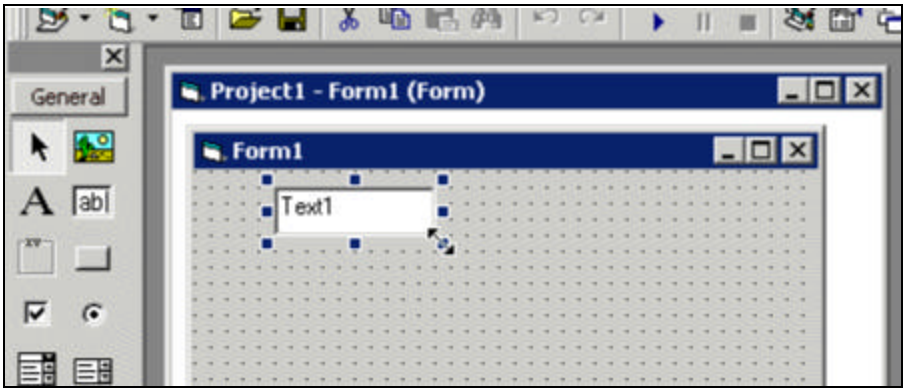
And now we're prompted for the new project. We're going to create a standard "exe" (a standard executable program). Let's click on OK.



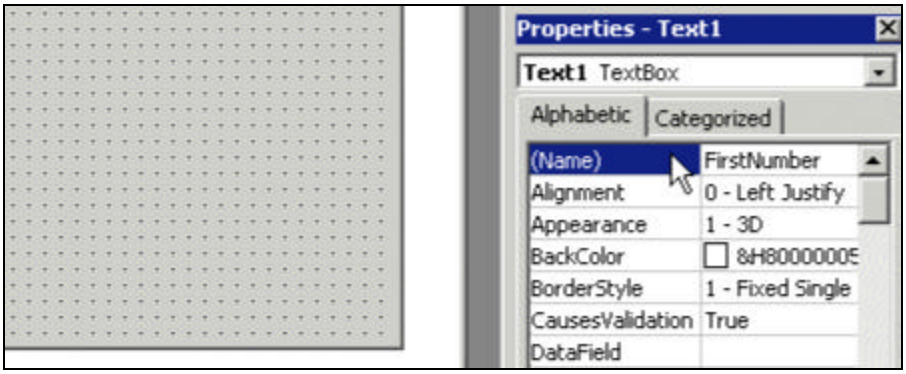
And here we are with a blank, new form and a blank new project. Here's what we're going to do for this project. We're going to build a simple calculator. We want, two text boxes where the user can enter in two numbers – a command button that they can then clic k on – and then a third text box that will have the answer in it.



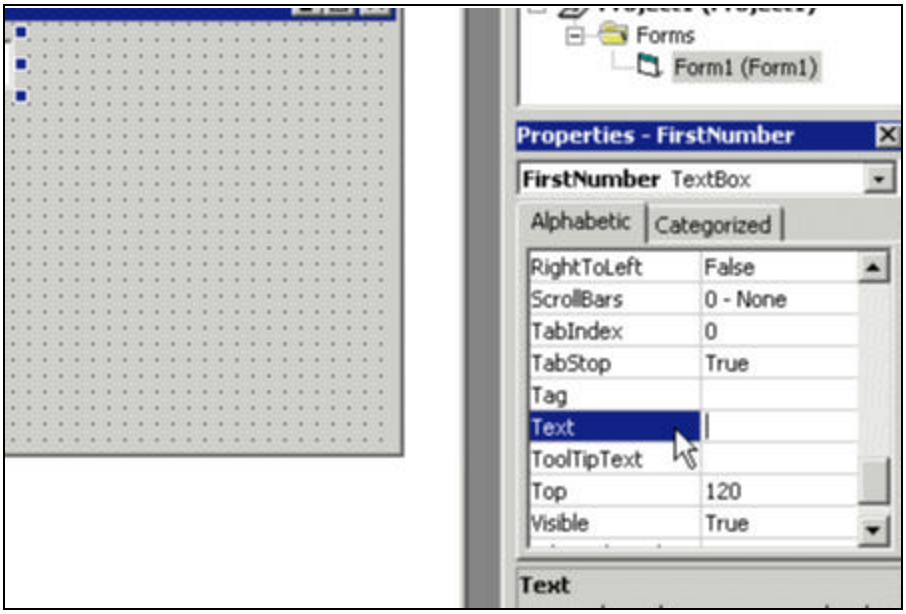
So let's start out by putting a text box on the form.



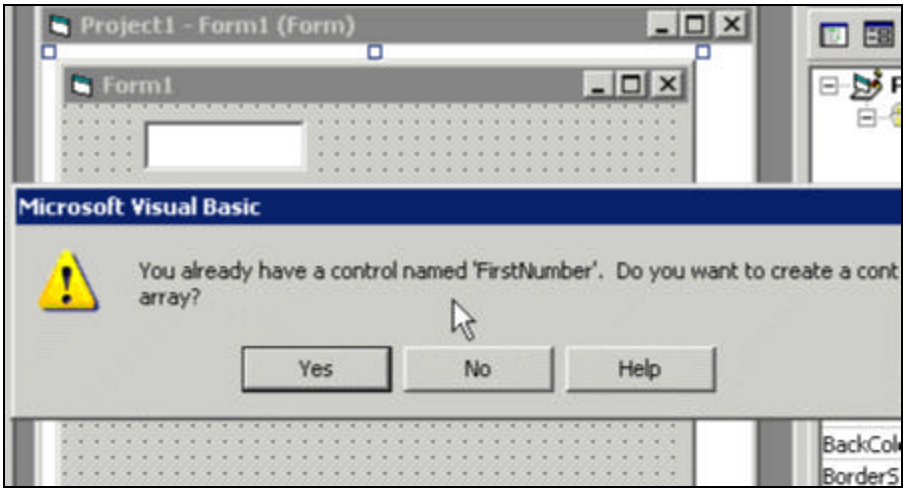
Change the name of the text box to "FirstNumber."



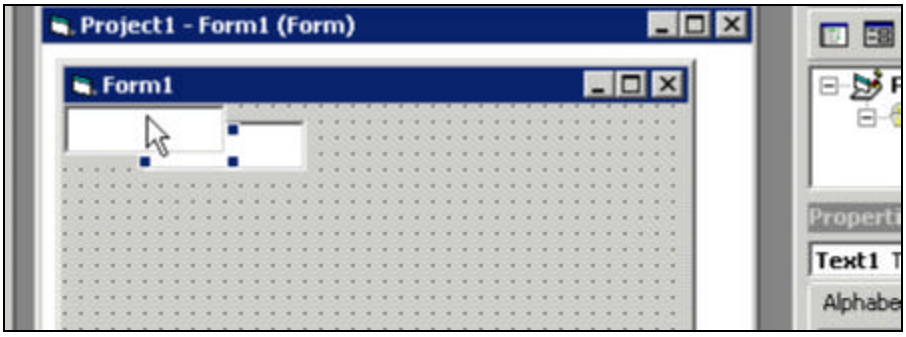
Scroll down and find text and get rid of it.



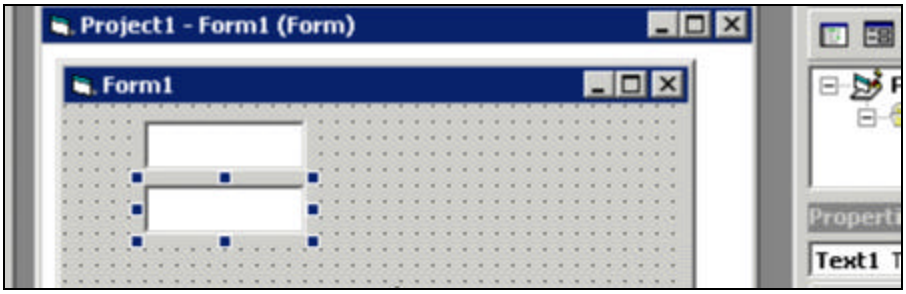
Now we need to make two more of these guys. We need to make one more for a number to go into and then we need to make one more for the answer. So let's just try copying and pasting this thing. Let's click on the text box and hit **copy** and then **paste**.



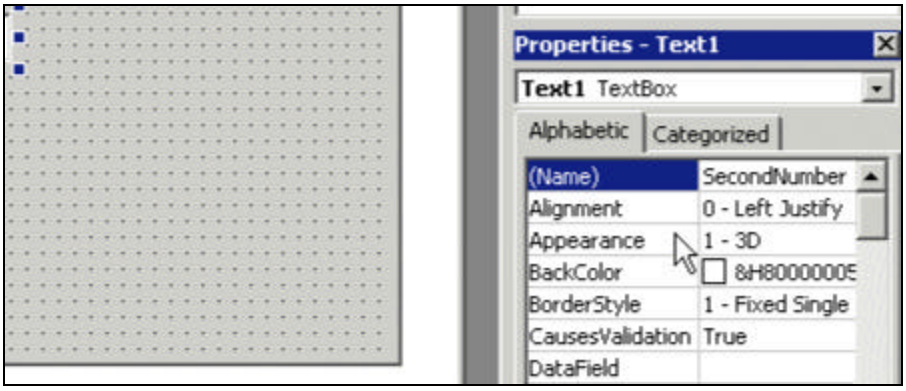
Hold on a second! Visual Basic is saying, you already have a control named "FirstNumber." Do you want to create a **control array**? Hm. We haven't talked about control arrays yet, so for today, let's just say **no**.



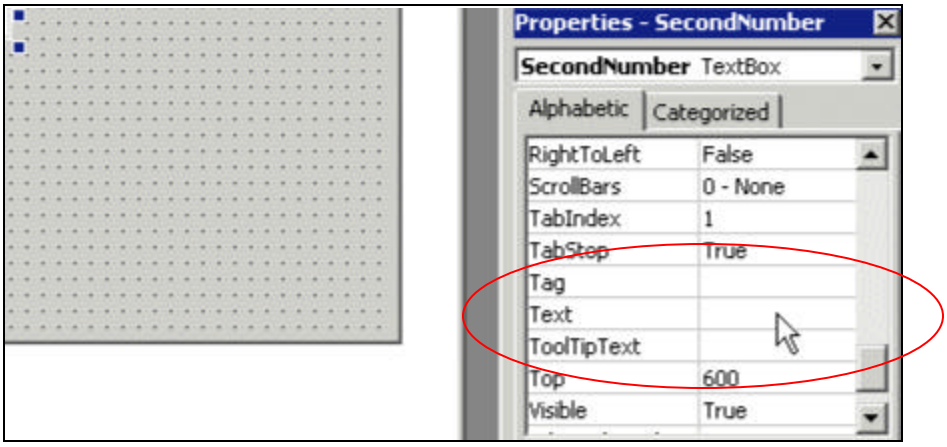
We should get our text box. Let's click on it and drag it underneath the first one.



Go to properties and change the name of the second text box to "SecondNumber."

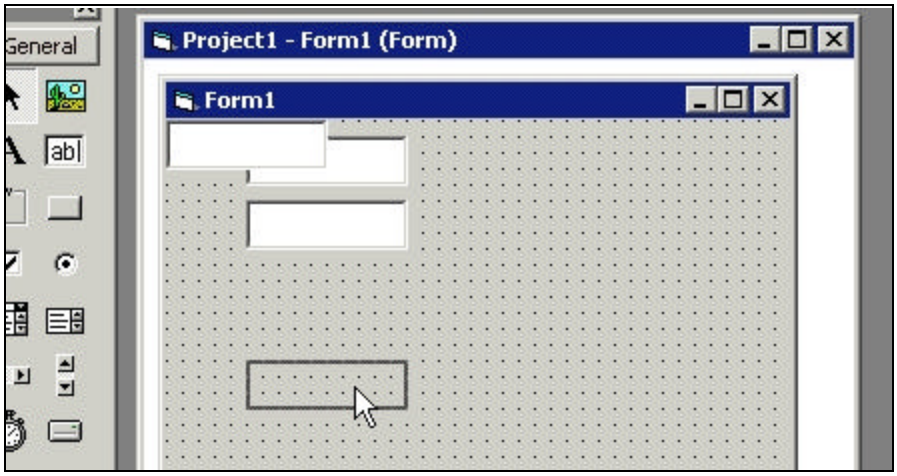


And notice the benefit of copying and pasting…

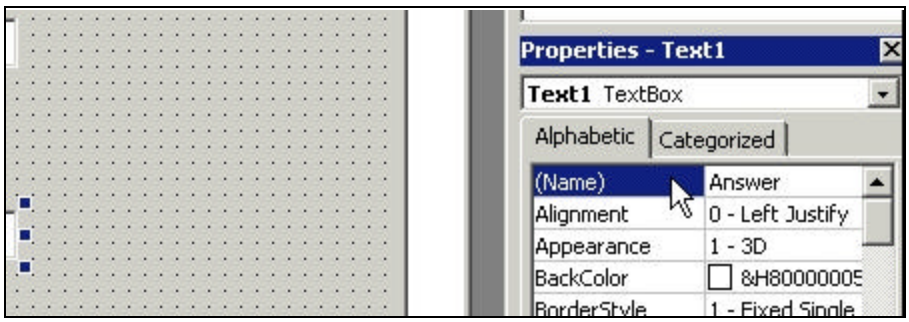


The text field here is blank because we copied the other one! It will copy almost all of the properties except of course its name. Let's make one more copy.

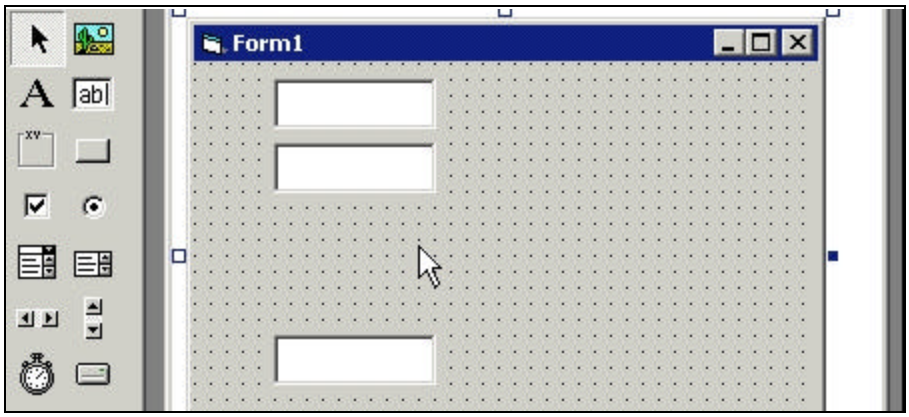
We'll drag this guy down on the bottom. That's where our answer will go.



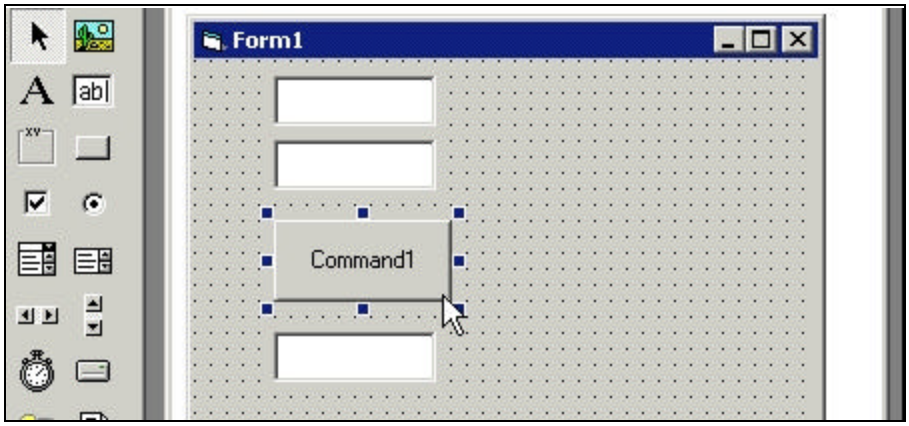
And let's name this guy, "Answer."



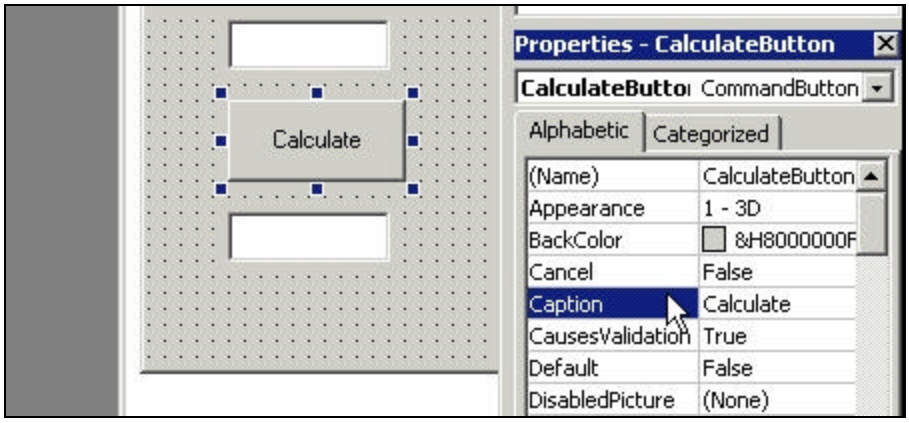
So now we have "FirstNumber," "SecondNumber," and "Answer."



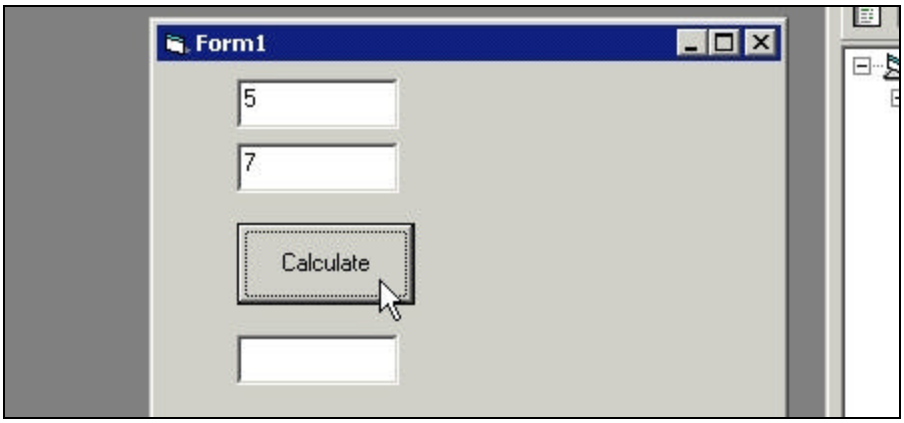
Now we need a command button in the middle here that we can click on. And add these two values up top and put the answer down at the bottom. So lets click on the command button. Draw a button in the middle.



Let's change its properties. Let's give it a good name. Let's call it, "CalculateButton." That will be the name of the button. Let's change its caption to "Calculate."

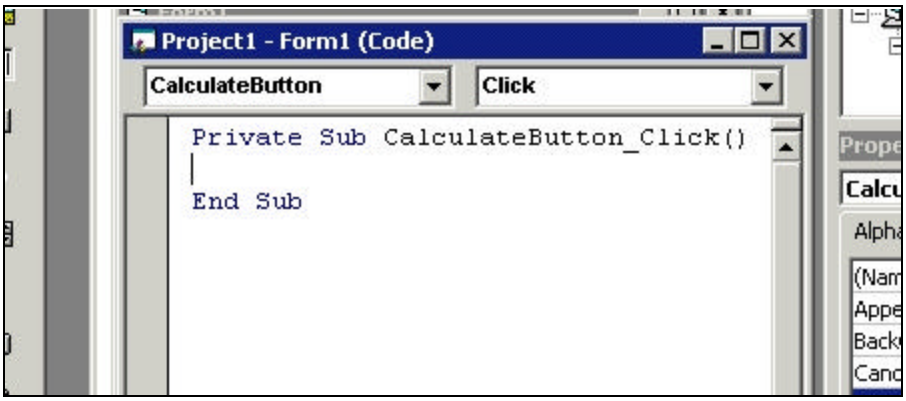


And now, let's go ahead and actually run our program. Type in a value up top, like "5" and then "7" and hit "Calculate."

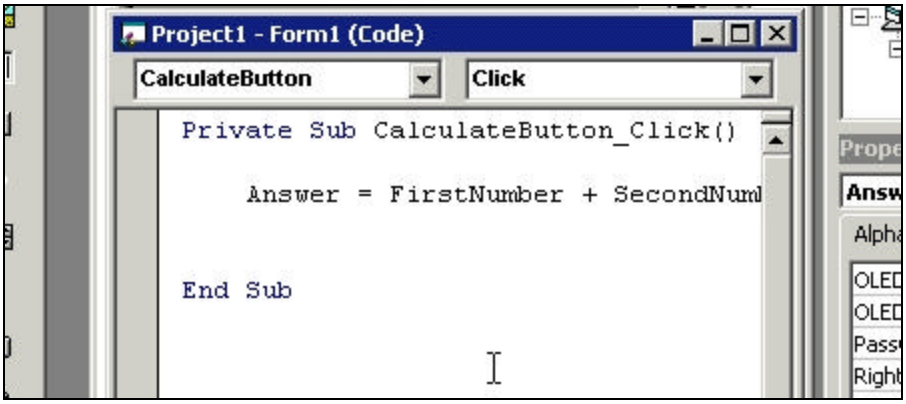


Wait a minute! We forgot to put programming in our button, right? The program just doesn't *know* what we want to do! We have to actually tell it what to do. So let's put some programming in there.

Let's close our program and click on the "Calculate" button to open up the code window. Here we are inside the CalculateButton_Click(). This is a private subroutine which means that only this form can use it. Whatever text we type in between the Private Sub and the End Sub will run whenever the user clicks on CalculateButton on the form.

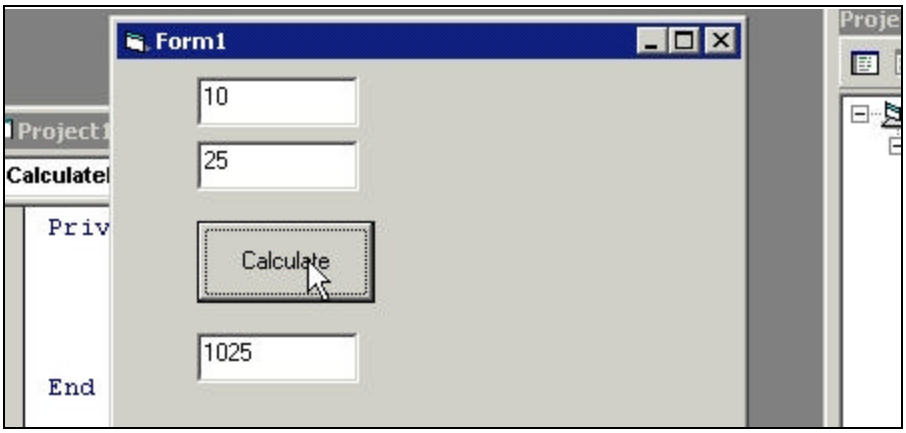


And now let's think, "What do we want to do with this program?" We've got two textboxes, "FirstNumber" and "SecondNumber." We want to then add their values together and put the answer in the answer text box. So here's how we're going to type it in. We're going to say, "Answer = FirstNumber + SecondNumber."



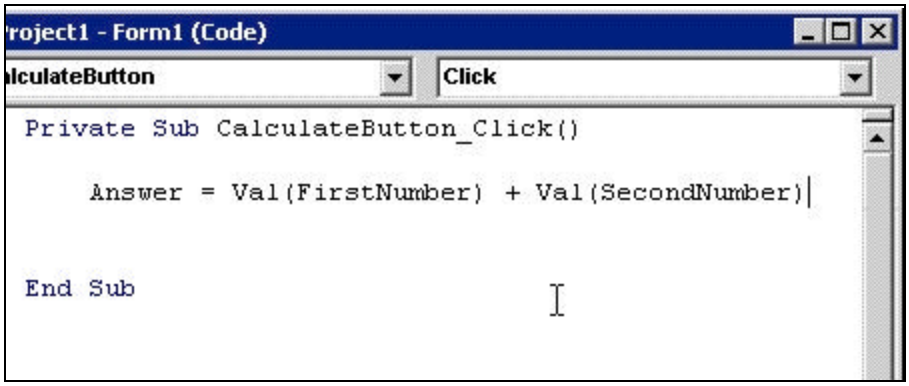
Basically, you do the math backwards. The name of the control goes first, "Answer." In this case, the "Answer" text box is going to be set equal to the FirstNumber text box plus the SecondNumber text box. Let's see how this works now. Let's go ahead and run our program. Click on the start button.

Type in “10.” Tab down and type in “25.” Then click on your “Calculate” button. And... wait a minute. That’s not quite right! We’ve got 1025. Hm.. What happened? Well essentially, Visual Basic is going to treat those text boxes as *text* – not numbers. So anything you type in there, is going to be treated like text. We could type in the top text box, “Richard” and “Rost” in the bottom text box, and then hit “Calculate.” That’s going to add those two things together. It’s going to treat them like *text*.



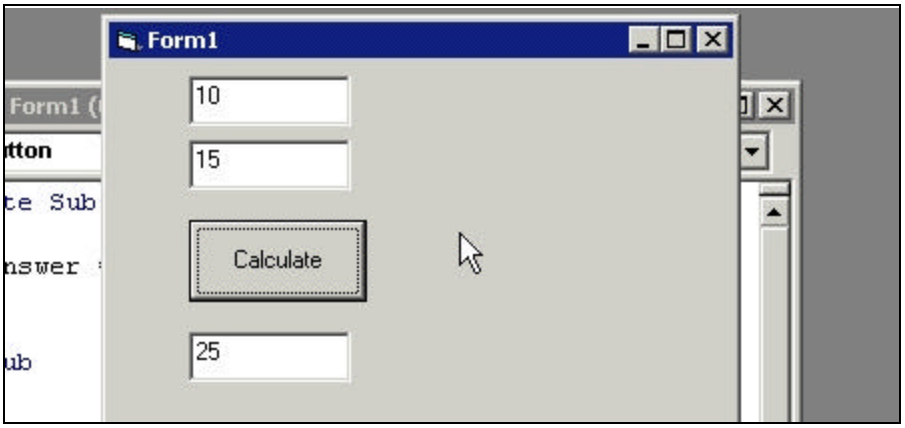
So what we have to first do, is we have to tell Visual Basic, “Hey! Treat these as *numbers* – not as text. Give me the **value** of the numbers in these boxes and don’t treat this stuff as text.” So how do we do that?

Well, let’s close our program to get back inside of our code. Let’s click right in front of “FirstNumber” and type in the word, **Val**. Then open parenthesis. Then click at the end of FirstNumber and put in a close parenthesis. That’s the **Val function**. Basically it’s going to take that FirstNumber field and convert it to a number. Let’s do the same thing for SecondNumber.



So “Answer” is going to be set equal to the **value** of the first number **plus** the value of the second number. Let’s go ahead now and run our program.

Let's type in "10" and then "15" and hit "Calculate."

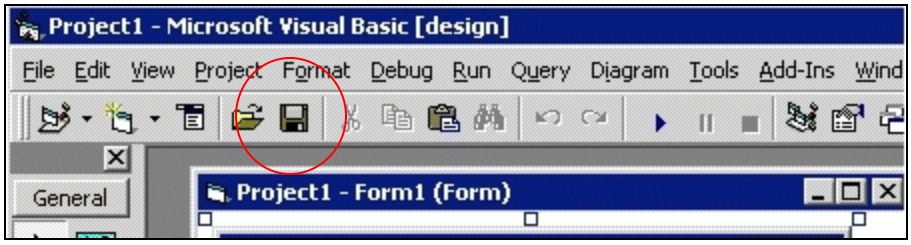


Oh – It's a beautiful thing! Again, give yourself a little round of applause.

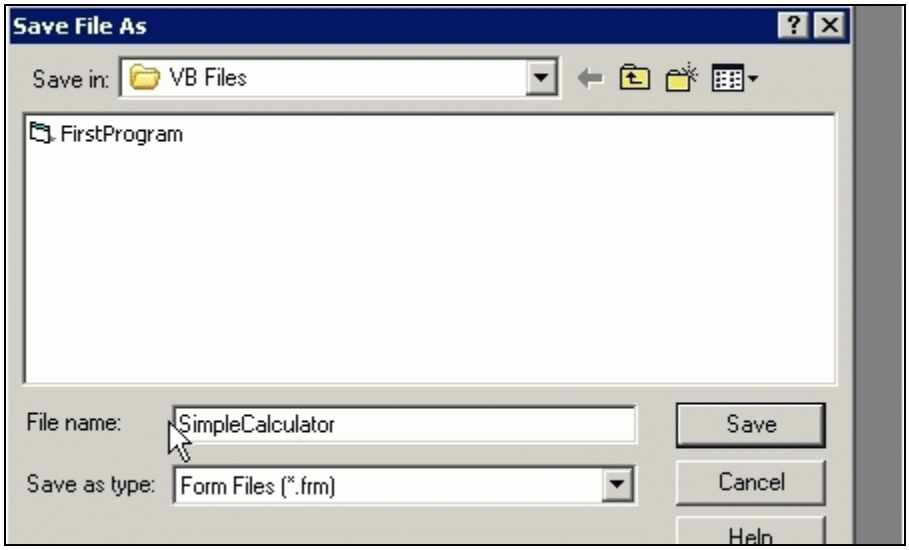
You can see how the **Val** function took the text in the text box and converted it to a number. Let's go ahead and close our program and close our code window. And now we've made our first simple calculator!

Lesson 7. More Calculator

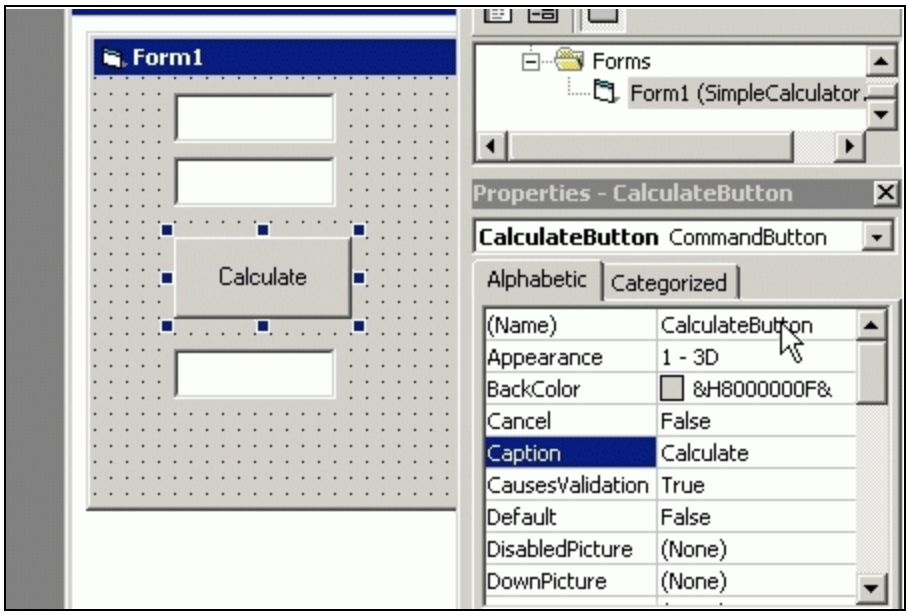
In the last lesson, we made a simple calculator, but all our simple calculator can do is add two numbers together. In this lesson, we're going to make it capable of subtraction, multiplication, and division. Now before we go too far, we forgot to save our program in our last lesson, didn't we? And that can be a problem – especially if the power goes out, or someone comes along and spills coffee on your keyboard! Let's go ahead and save our program by clicking on the **save** button.



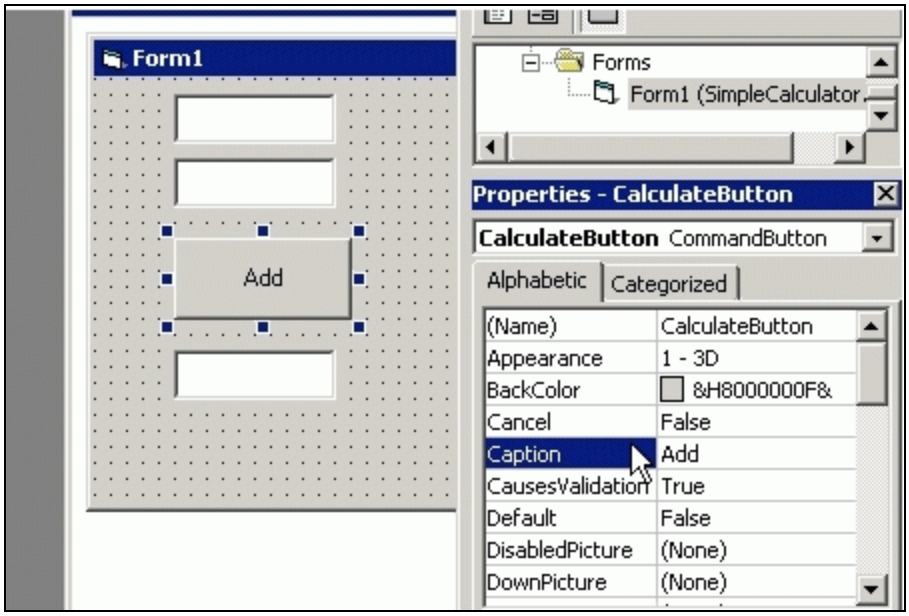
Notice we're still in our VB Files folder.



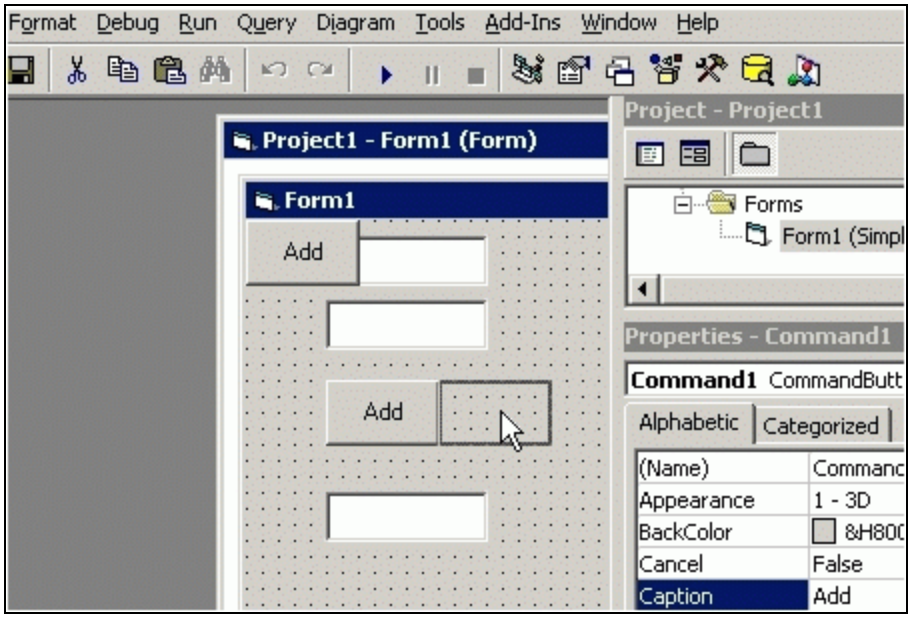
Let's save the form. Let call this "SimpleCalculator" and hit save. We can also save the project as "SimpleCalculator as a "vbp" file. It's okay to have the form and the project named the same thing. Let's change the "Calculate" button so the caption says "add." We'll make some more buttons that subtract, multiply, and divide. Click on the "Calculate" button and move to its properties.



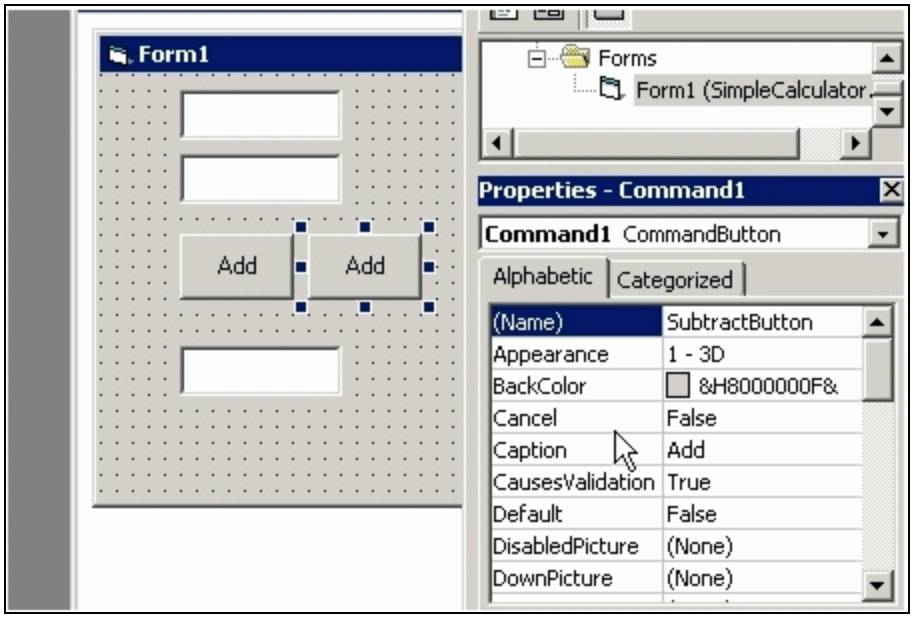
We can leave the name alone. **CalculateButton** is fine – we don't have to worry about changing it to "SumButton." But let's change "Caption" to "Add."



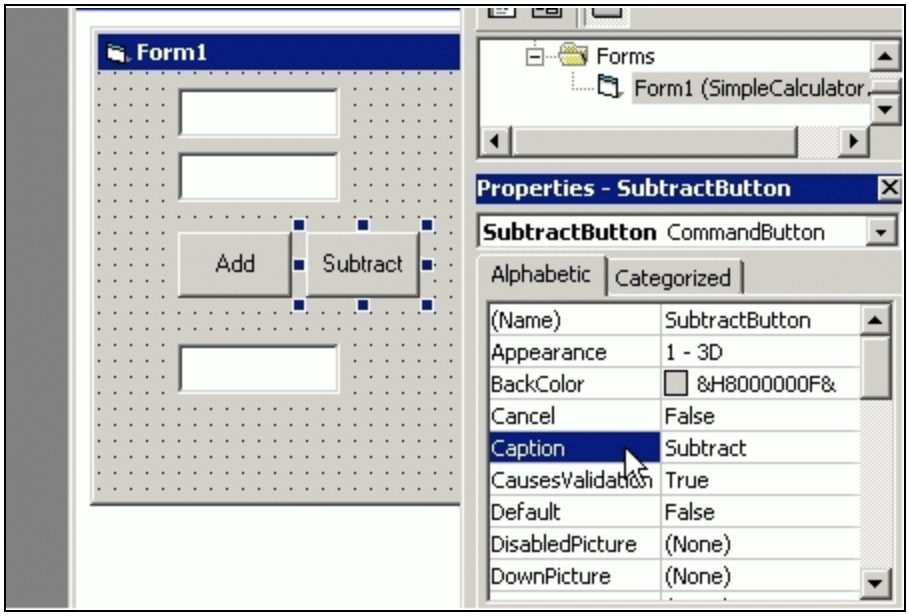
Now the button says “Add.” In fact, let’s make it a little smaller. Now let’s add another button to subtract. We can just copy and paste the “Add” button. Slide it down next to the “Add” button.



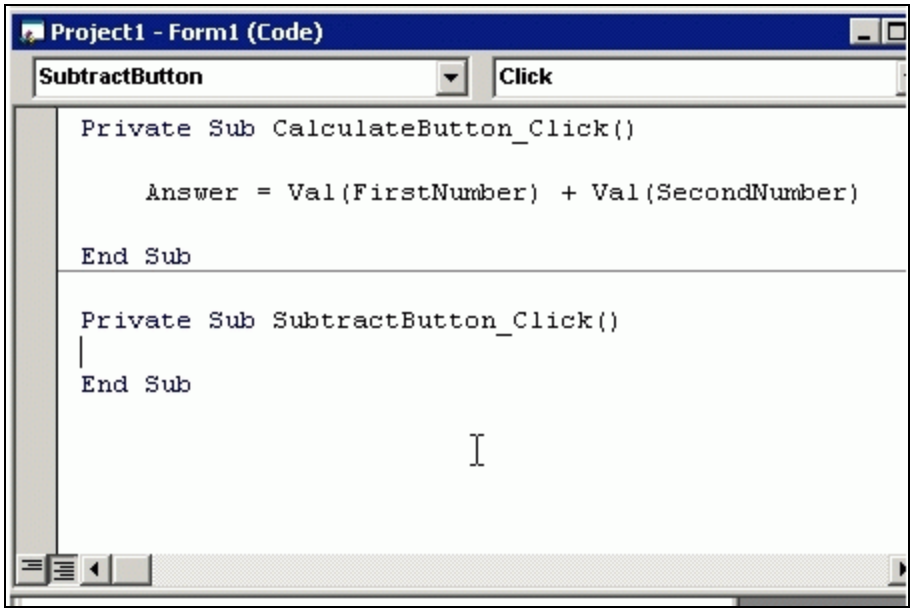
Let’s change this one’s name to “SubtractButton.”



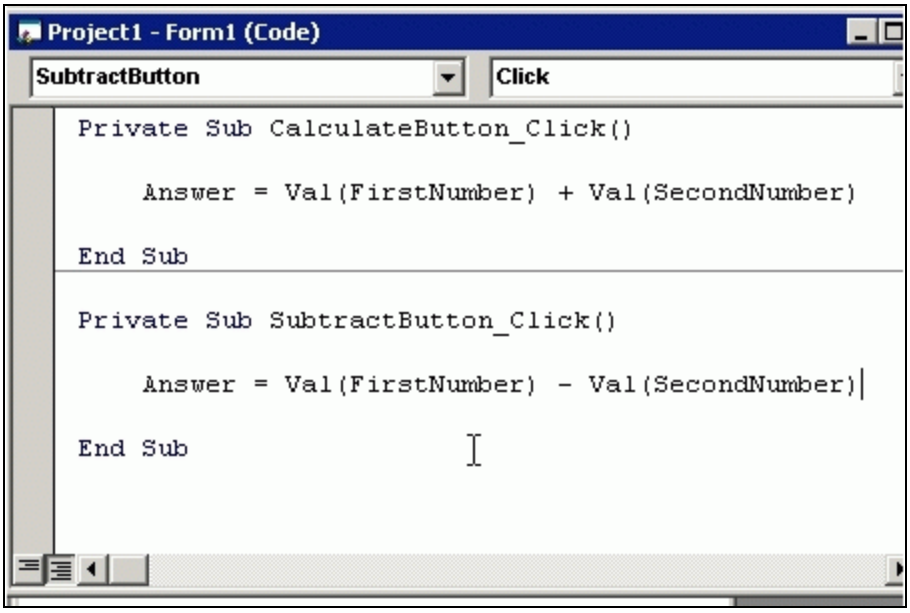
And we'll put the caption "Subtract" on it.



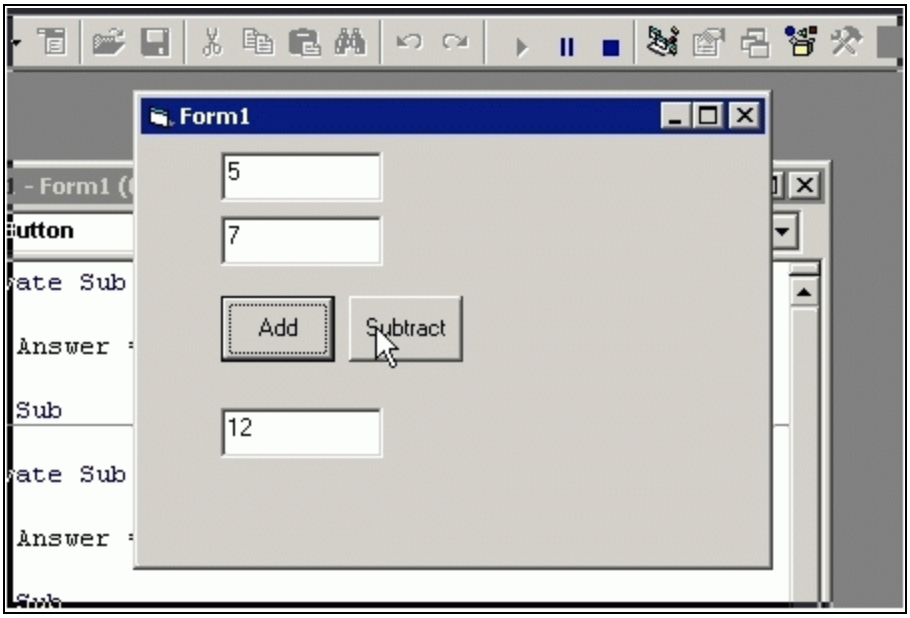
Now let's see its code. Double click on the subtract button.



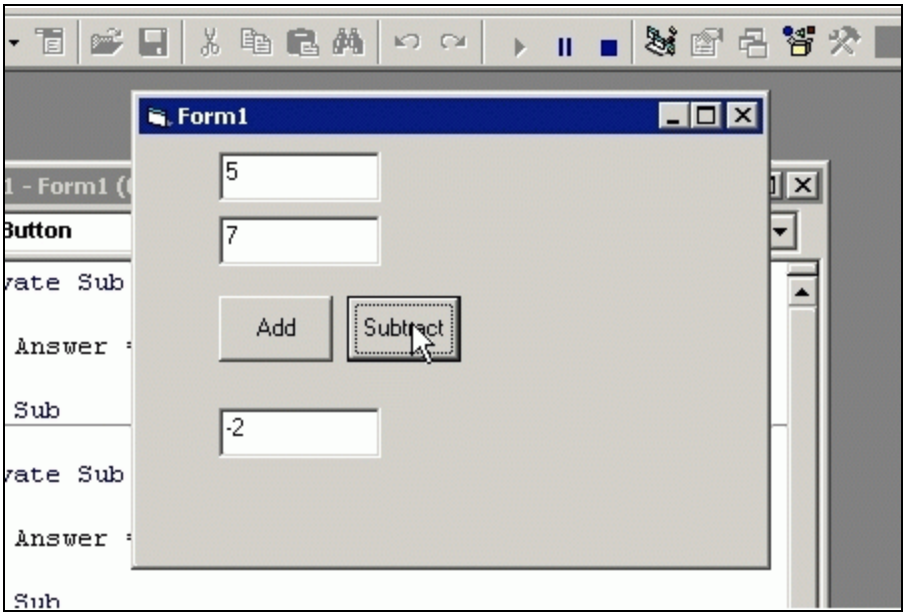
Here we are inside of our code window again. Press the enter key a couple of times. This time its going to be very simply "Answer equals the value of the first number **minus** the value of the second number." Could we have copied and pasted that and just changed the operation? Certainly.



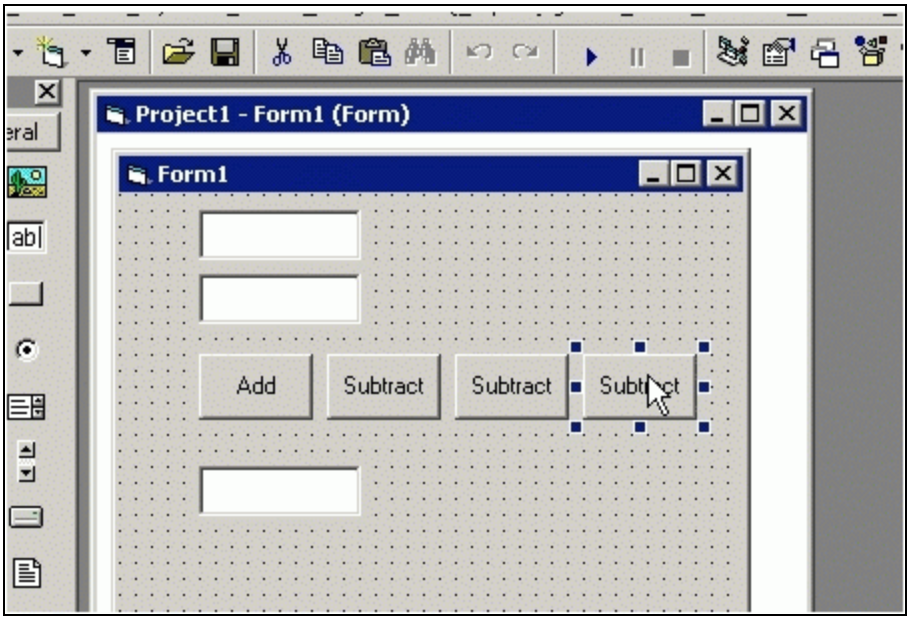
Let's go ahead and try it. Let's run our program.



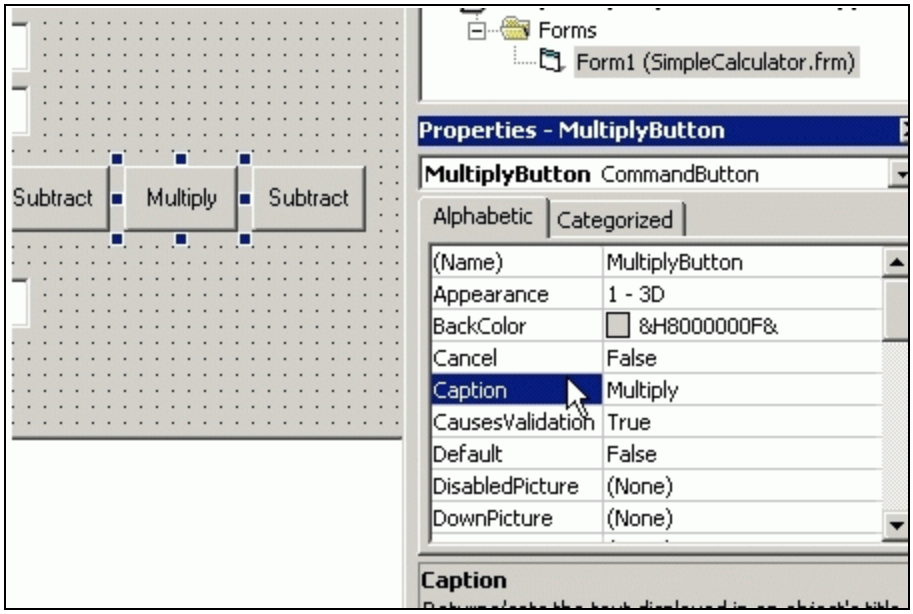
Type in five and seven and click add. We get twelve. Click subtract. We get negative two.



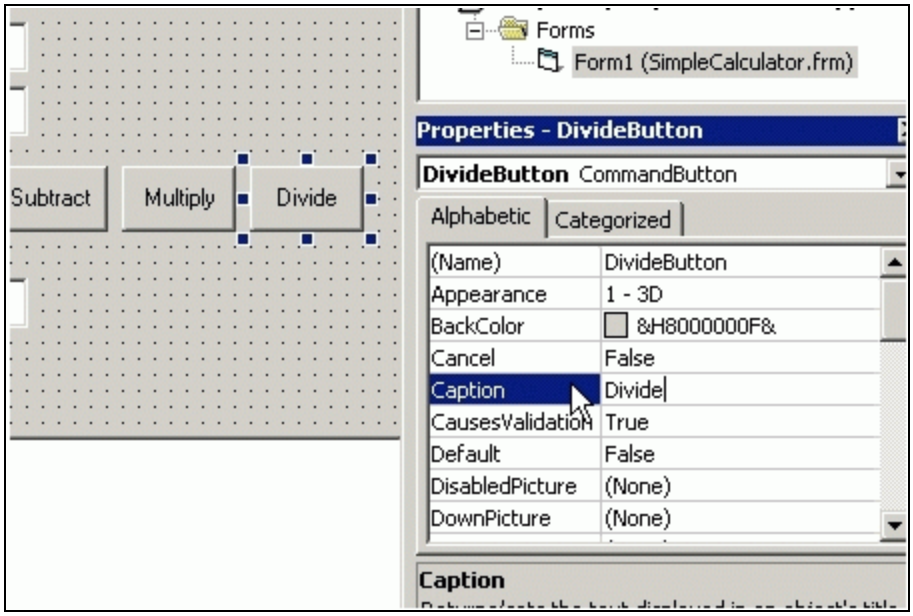
Perfect. Our add and subtract buttons are working perfectly! Let's add two more buttons real quick. Let's add multiply and divide. Close the program and we're going to close the code window for now. Let's copy and paste two more buttons.



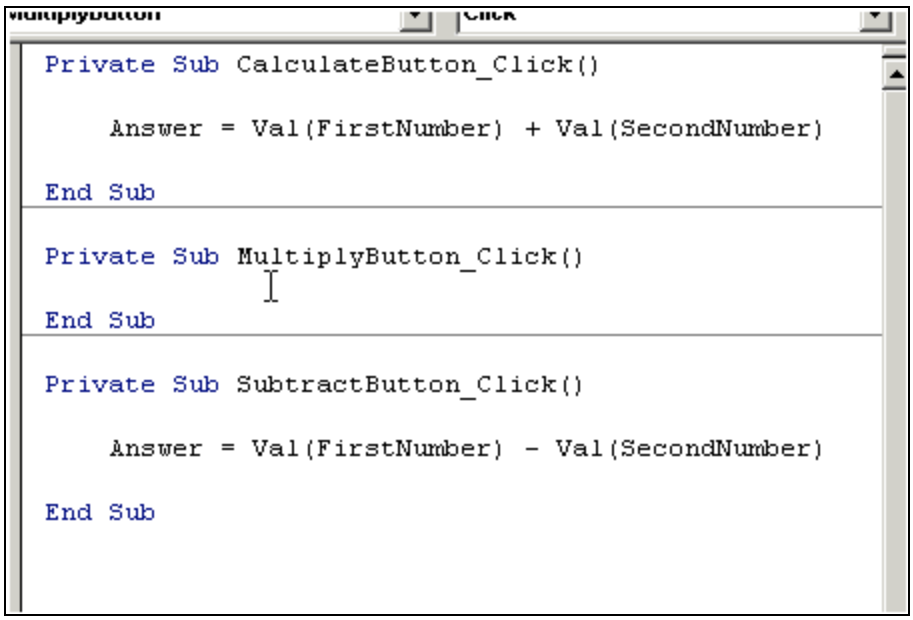
Let's change the properties of these buttons now. The third button's name is going to be "MultiplyButton"



And its caption will be multiply. We'll click on the next button. This one will be "DivideButton," with a caption of "Divide."



Now let's double click on these buttons to adjust their code.



```
Private Sub CalculateButton_Click()

    Answer = Val(FirstNumber) + Val(SecondNumber)

End Sub

Private Sub MultiplyButton_Click()

End Sub

Private Sub SubtractButton_Click()

    Answer = Val(FirstNumber) - Val(SecondNumber)

End Sub
```

We've double clicked on the multiply button. Notice that we're between the other two buttons. That's because Visual Basic arranges the subroutines alphabetically. Notice we've got CalculateButton then MultiplyButton, then SubtractButton. But the code for these is going to be real easy. All we'll do is highlight the top one in the CalculateButton subroutine, copy it with Control + C and then paste it with Control + V in the MultiplyButton subroutine.

We use these keyboard tricks all the time.

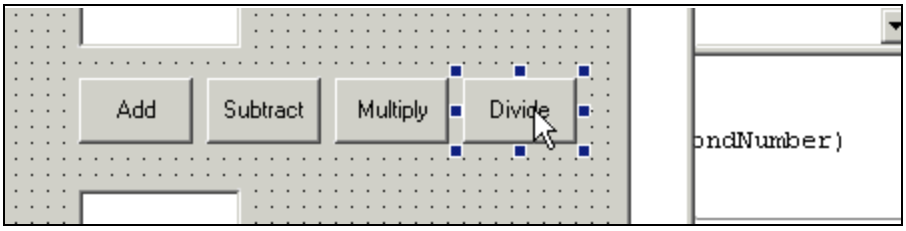
Ctrl-C	Copy
Ctrl-X	Cut
Ctrl-V	Paste
Ctrl-S	Save

Become familiar with these keyboard shortcuts.

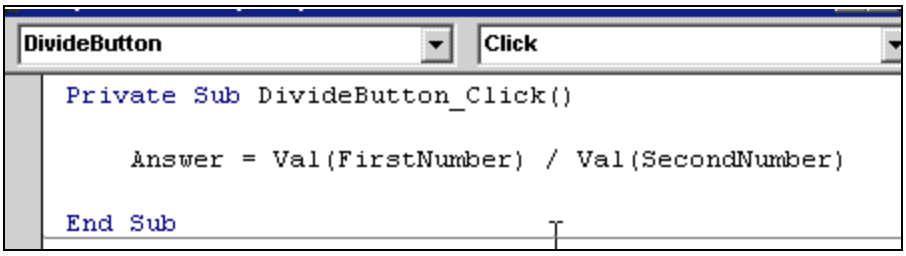
So anyhow we've pasted the code and now we'll change the plus operation to a multiplication operation, which is that asterisk (Shift + 8).

```
Private Sub MultiplyButton_Click()

    Answer = Val(FirstNumber) * Val(SecondNumber)

End Sub
```

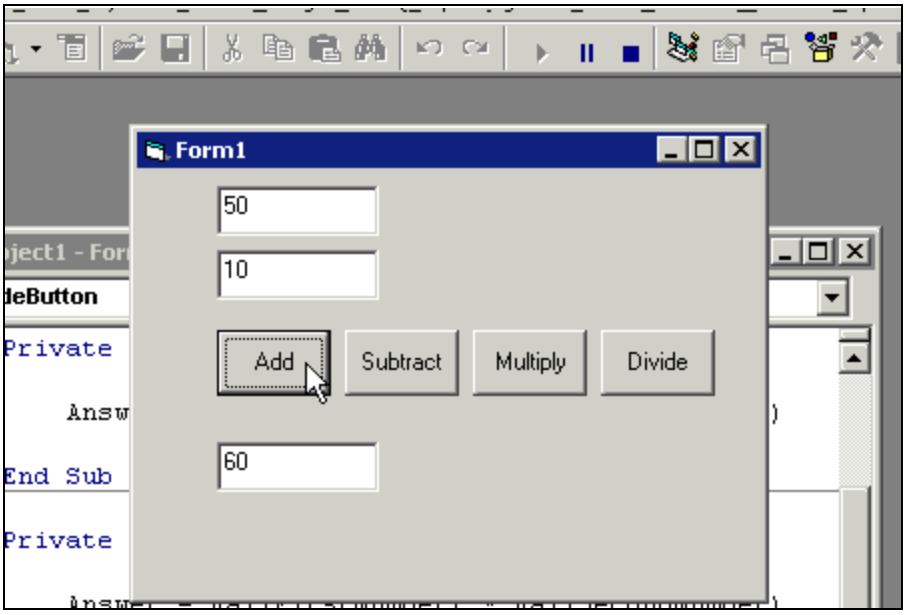
Now as you can see both your code window and your form window on the screen, all you have to do to flip between them is simply click on the form window. Now double click on "Divide" here.



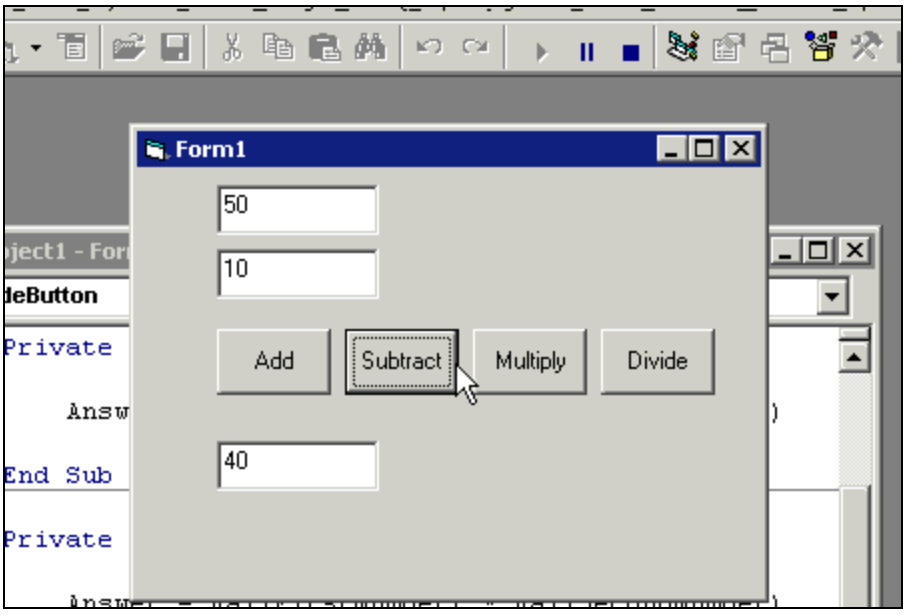
That will put you in a new private sub back in the code window. Now all you have to do is paste (hit Control + V). That code that we copied earlier was still in the Windows clipboard. Division is a forward slash.



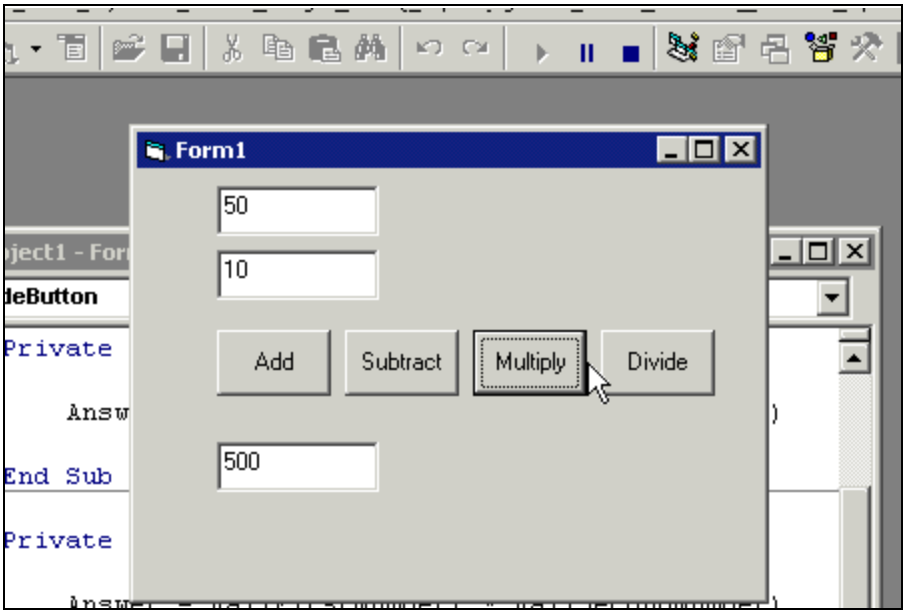
The value of FirstNumber divided by the value of SecondNumber. Let's go ahead and save our program and run it. Let's type in fifty and ten. Let's add them.



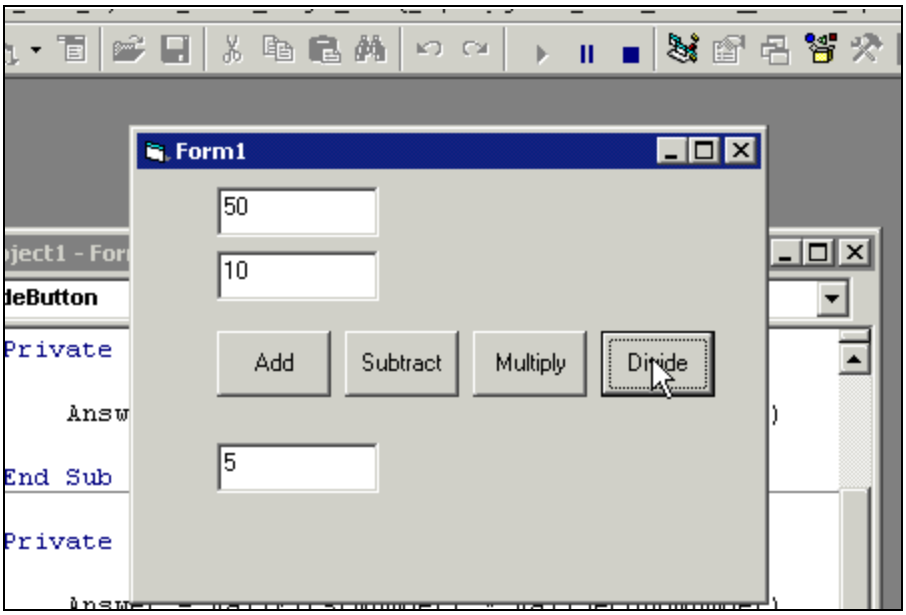
Subtract them.



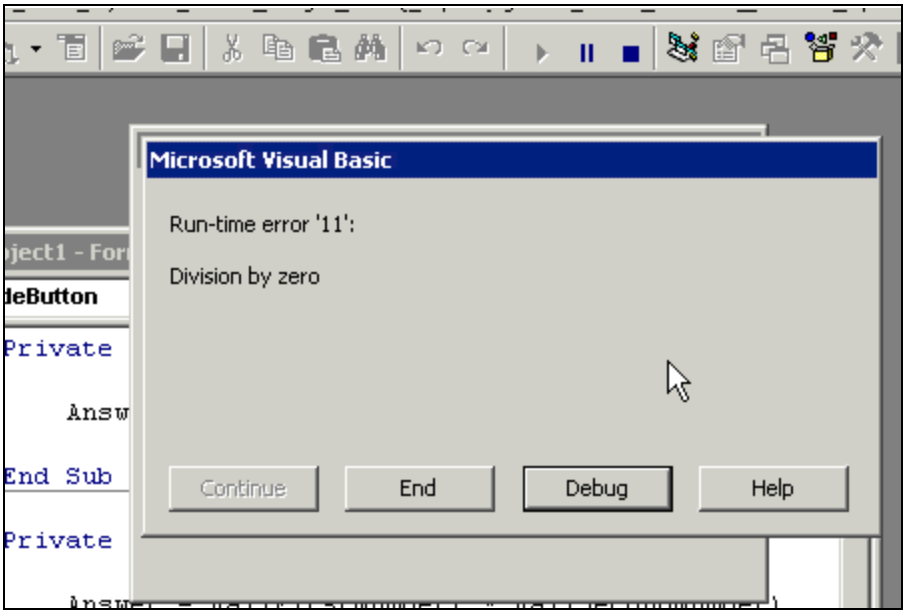
Multiply.



And Divide.



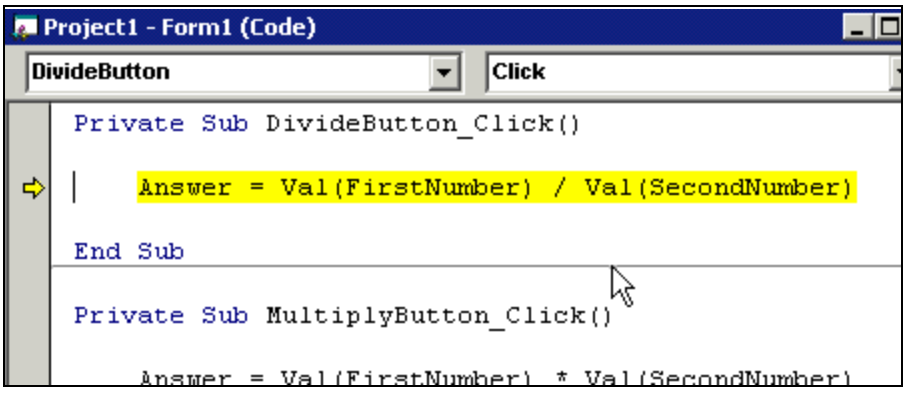
Beautiful! Let's change the ten to zero. Then click Add, Subject, Multiply, then Divide. Uh-oh!



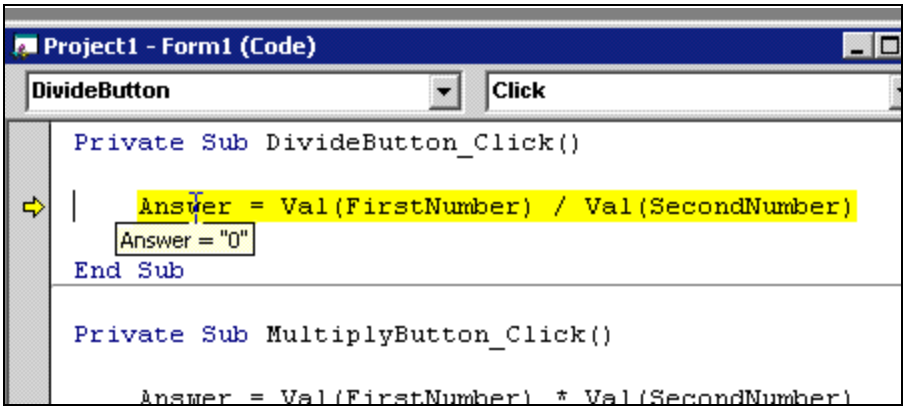
There's an error! This is not a compile error. This is not a syntax error in our code. Our code is fine. What we have here is a run-time error. There's two kinds of errors you're going to get. **Syntax errors** are generally compile-time errors. In other words, when the program gets put together before it actually runs, then Visual Basic program looks at it and says, "Hey! This isn't right. I can't run your program. Something's wrong here." Those are the easy ones to find.

Then you've got your **run-time errors**. Run-time errors are the nasty ones. These are errors in the logic of your code. They don't usually show up until you start running your program. And sometimes they're buried deep so they don't even show up right away. But this is an easy one to fix. It say's "Division by zero.'

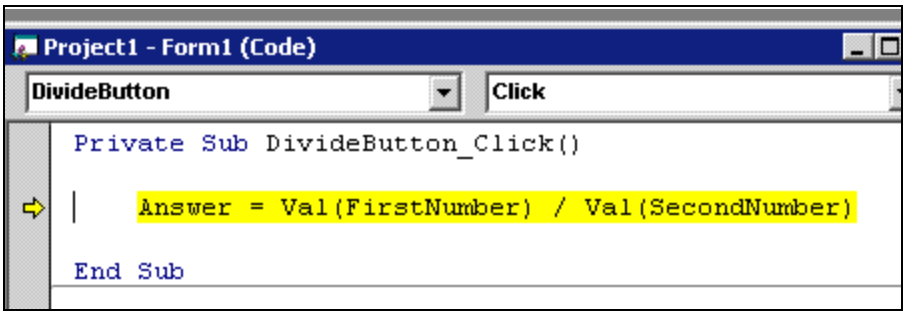
Basically, you've violated one of math's rules. You can't divide by zero. So what can we do? Well we can either End the program or we can Debug the program. Let's go ahead and click on the Debug button.



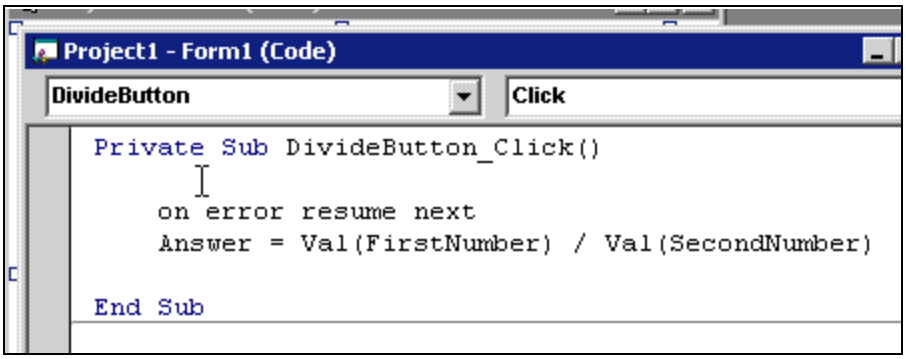
Essentially it brings us into the code that generated the error. If you hold your mouse point over any of these you can see the value of what these different variables are.



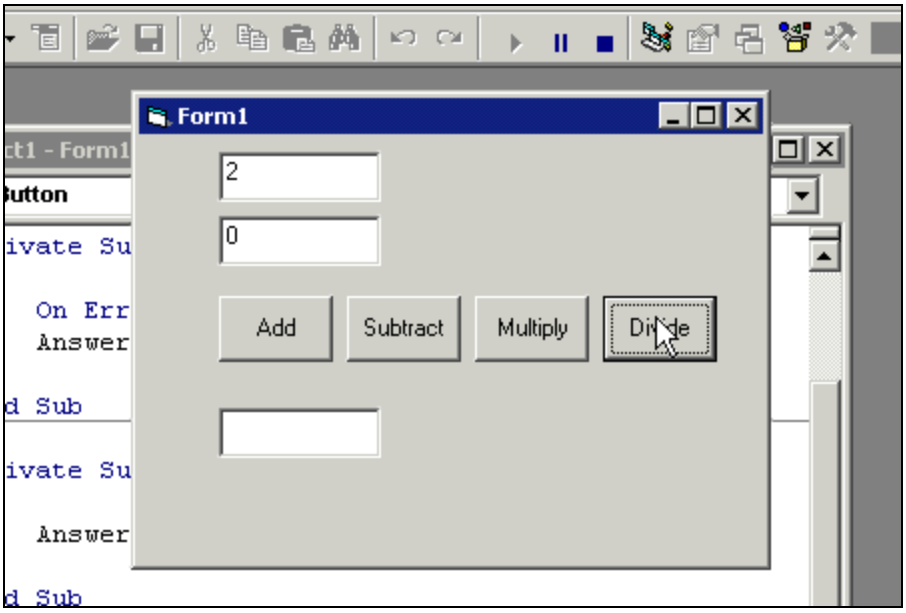
What we're going to do at this point, is we're going to stop our program. Our program is technically still running.



Click on the End button (up top), and that will stop the program. Now there's two ways we can fix this error. The first way is to simply have Visual Basic ignore any errors and continue moving. Click in the DivideButton subroutine and press enter to get a blank line (and then tab in). Type "**on error resume next**".



What does that do? Basically it says, if you encounter an error while running this subroutine, ignore it and move on to the next line. In this case, there is no next line. But it will basically not generate an error. Let's see how that works. Let's run the program. Type in two and zero and hit divide.



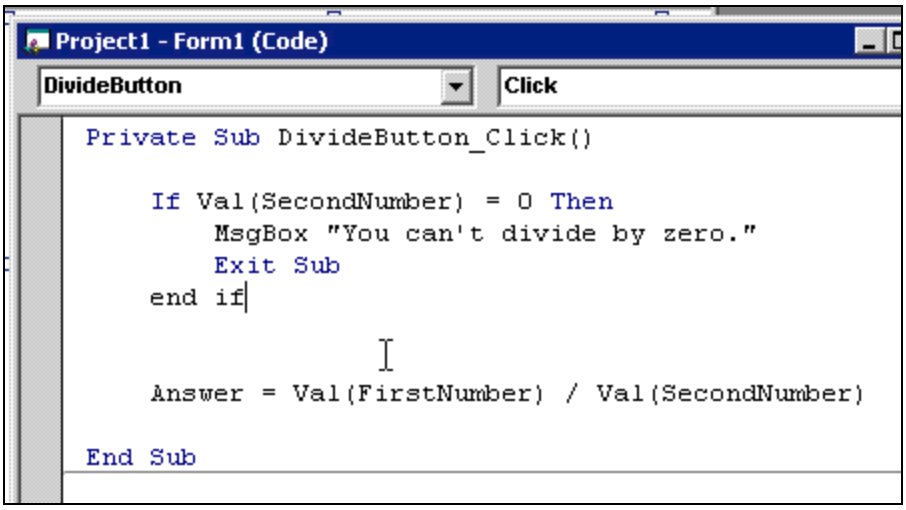
Nothing happens! Like I said, it was easy. But there's nothing telling the user, "Hey you can't divide by zero!" So it would be nice if we could generate our own little error message. Let's stop the program.

Now there are a couple of ways to do this. Let's get rid of this "On Error Resume Next" and let's put in a couple of extra blank lines. What we'd like to do is we'd like to handle this error ourselves. And the way we're going to do that is we're going to look at the denominator (the second number). And we're going to see if it's equal to zero before we try dividing it. If it's zero, we'll pop up a message box that says, "Hey you can't divide by zero!" It it's not zero, we'll let it do the math.

In order for this to work, we have to make Visual Basic smart. We have to make Visual Basic be able to look at the value and make a decision either way as far as what to do. If this is zero, pop up the error message and get out of the subroutine. How do we do it?

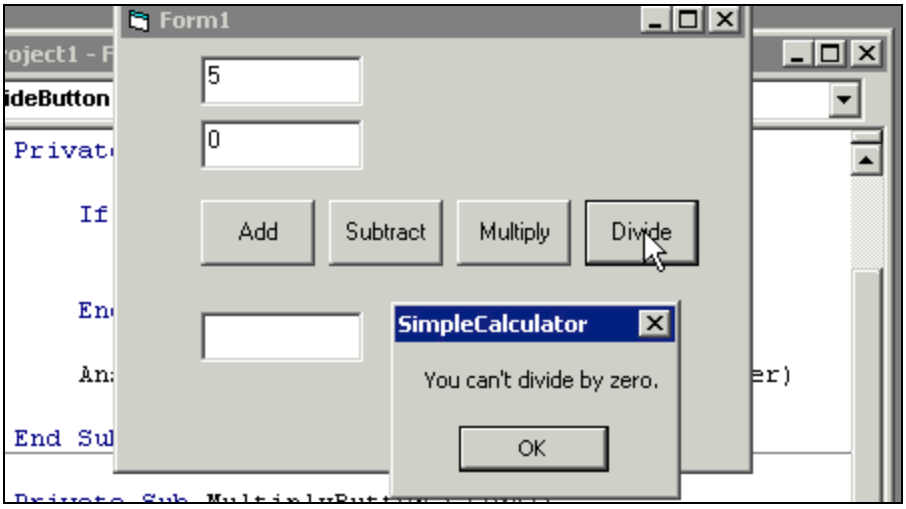
Well we're going to use a new function called, "if." We're going to use an "**if-then statement**." And here's how it works.

Type in: "if the value of second number equals zero, then MsgBox 'you can't divide by zero', exit sub, end if." Like this:

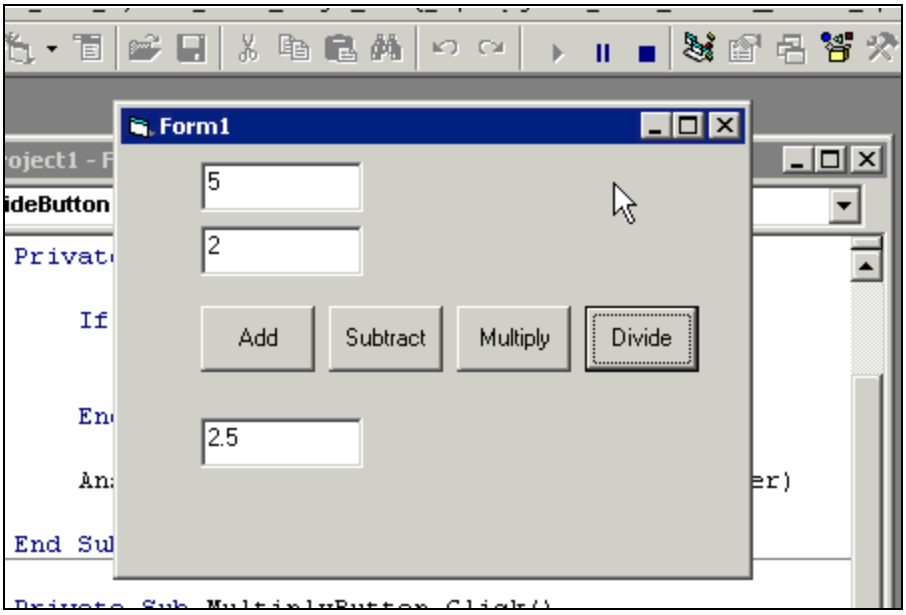


What is all this? Let's take a look at it line-by-line.

"If" (means that we're going to check a value) the value of the second number is equal to zero, then do some stuff. We're going to do all the stuff between the "if" statement and the "end if" statement. You can put any number of lines that you want in there. What's the stuff? Well, MsgBox, you know. Can't divide by zero – that's easy. Exit sub means "get out of Dodge." That means exit this sub – you're done. Don't continue on below. Let's save our program and run it to see what we get. Enter in five and zero and then click "Divide."



Look at that. Isn't that nice? Hit Ok and let's try something with an actual value. Replace zero with two and hit "Divide."



We get 2.5. Let's go ahead and close our program.

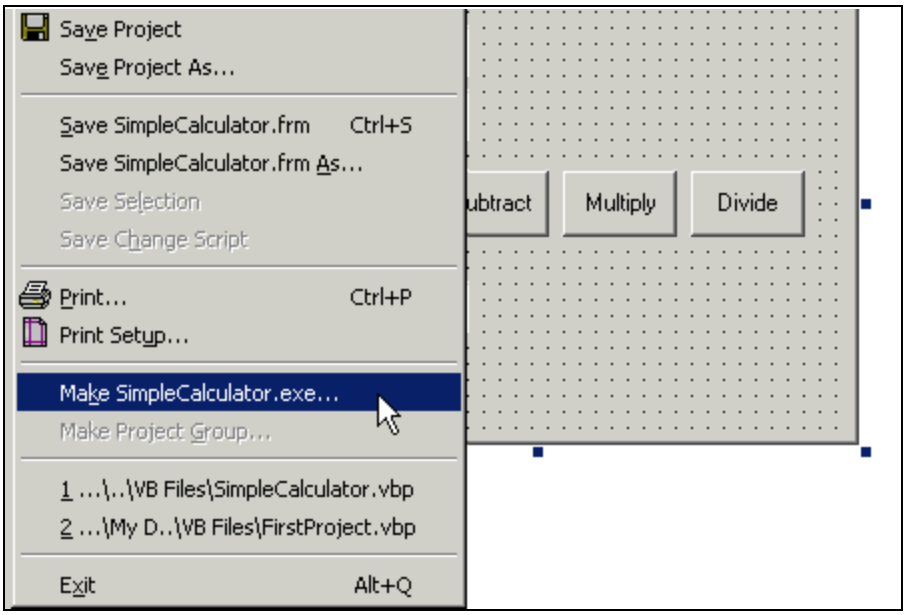
Now you know how to make your Visual Basic program "smart" with an "if" statement.

Lesson 8. Compiling

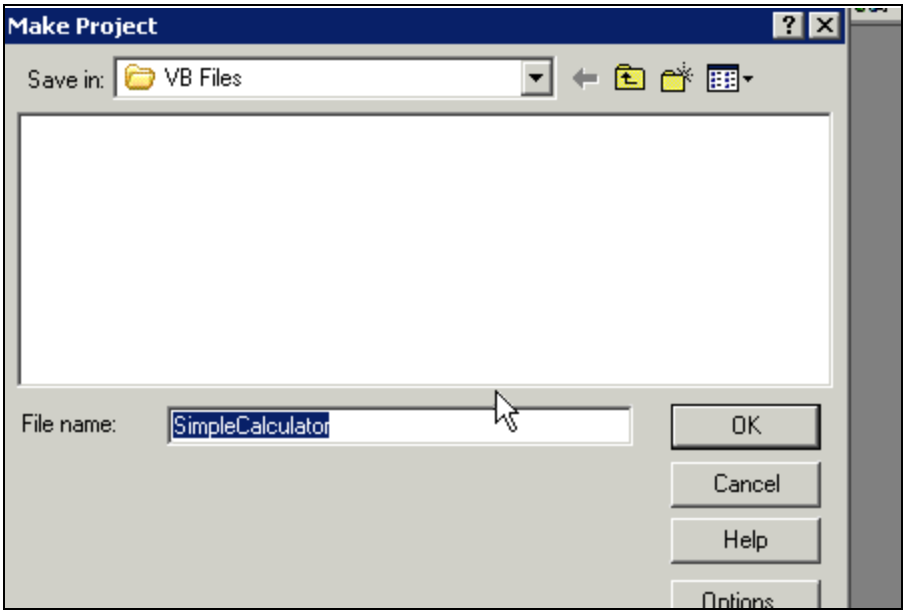
In this lesson, we're going to see how to compile our Visual Basic program into **a stand-alone executable file**.

If we want to run our simple calculator program, we have to come inside the Visual Basic Editor and then run the program. But that's not very useful in our day-to-day life. We don't want to have to stop and run Visual Basic just to run this program. So it would be nice if we have some way to compile this program up into an executable file. In other words, a program file that we can run just from inside Windows. And we can do that!

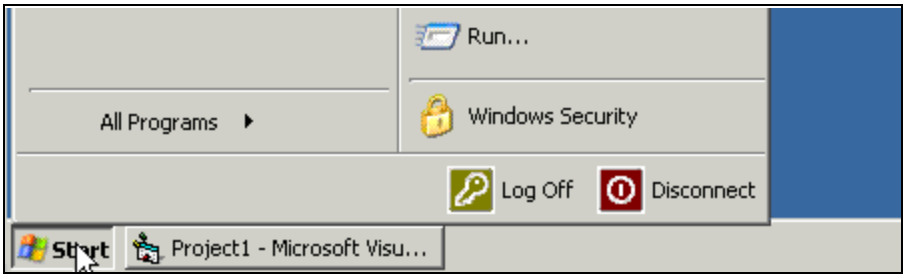
First, make sure you save your project. Hit the save button. Click on file and move down to "Make SimpleCalculator.exe."



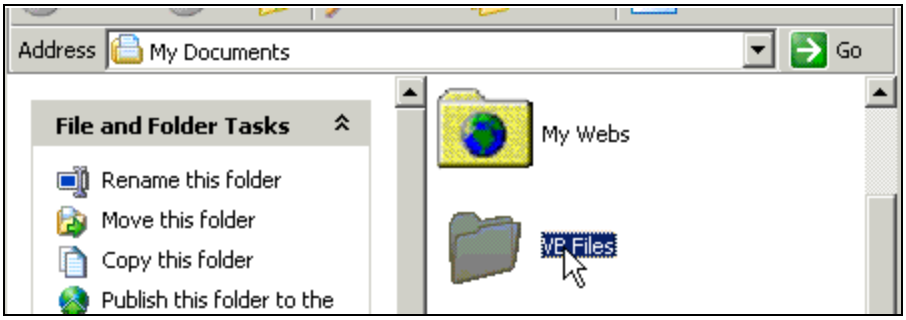
This will compile your program up into an executable program. It'll ask you for a name, such as **SimpleCalculator**. Hit Ok.



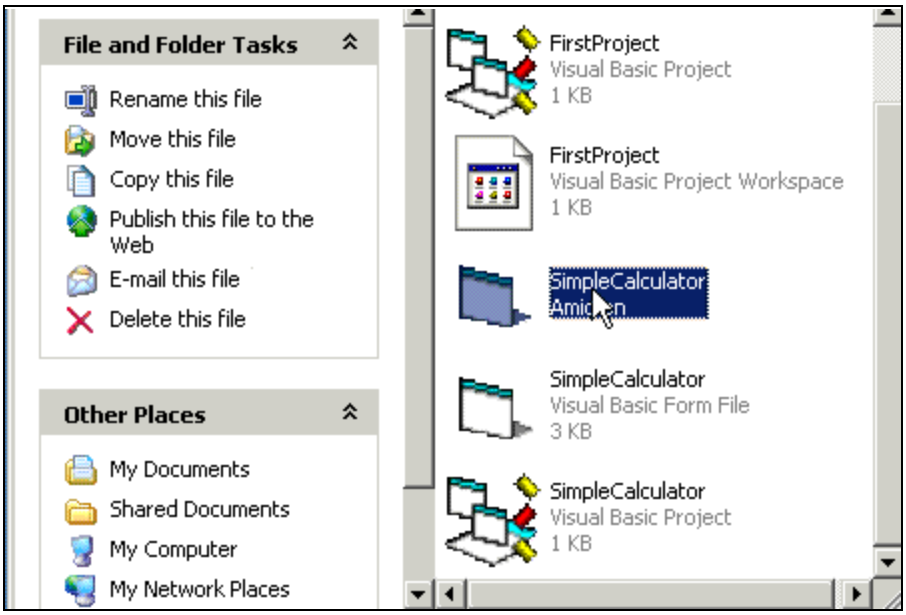
You'll hear Windows do some churning and bubbling. And when it's all done, you'll have an executable program. But where is it? Let's go look for it. Let's minimize our Visual Basic Editor for a moment and go to Start and then to My Documents. Click it.



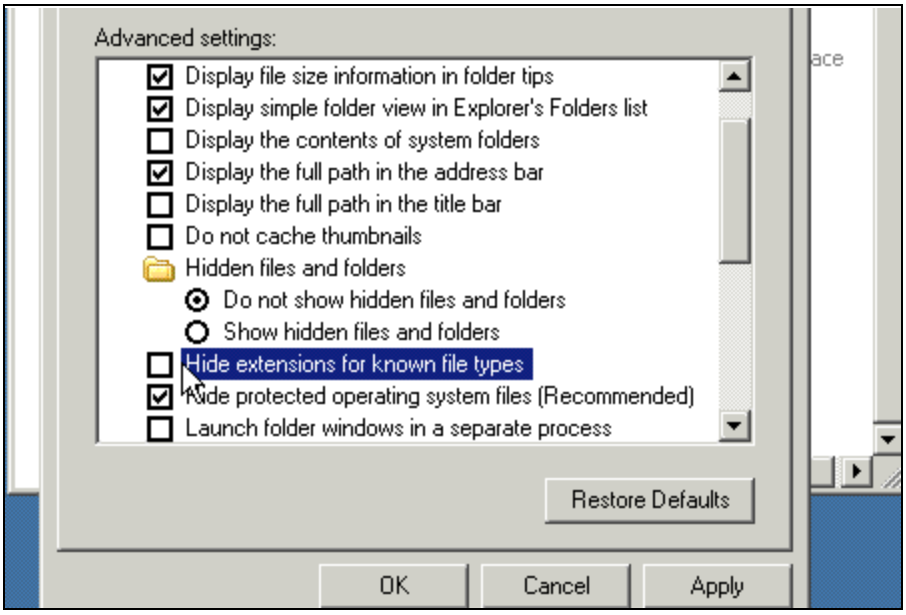
Double click the VB Files folder.



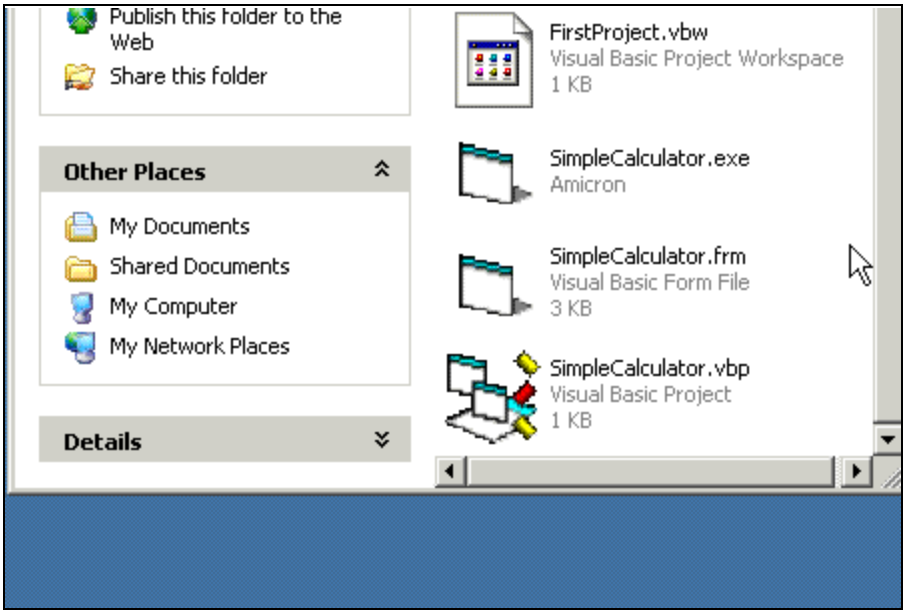
Here's our SimpleCalculator (that's our program), SimpleCalculator Visual Basic Form File, and SimpleCalculator Visual Basic Project file. How do you tell the difference between these guys?



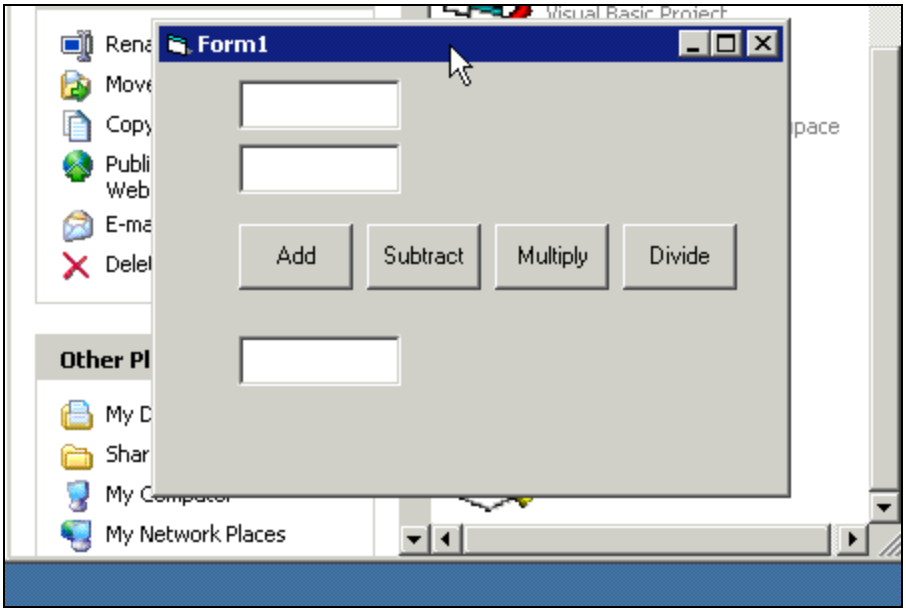
Click on “Tools” and then “Folder Options…” Go to the “View” tab. Scroll down and then uncheck, “Hide extensions for known file types.”



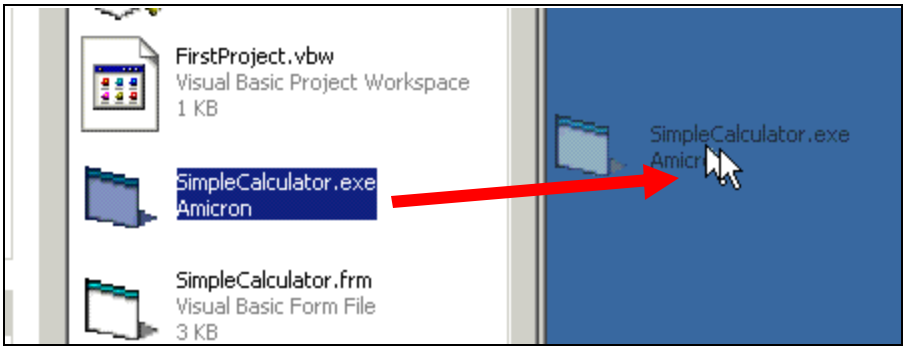
Click on OK. Now you can actually see what these are. Let’s find the “exe” file and double click on it.



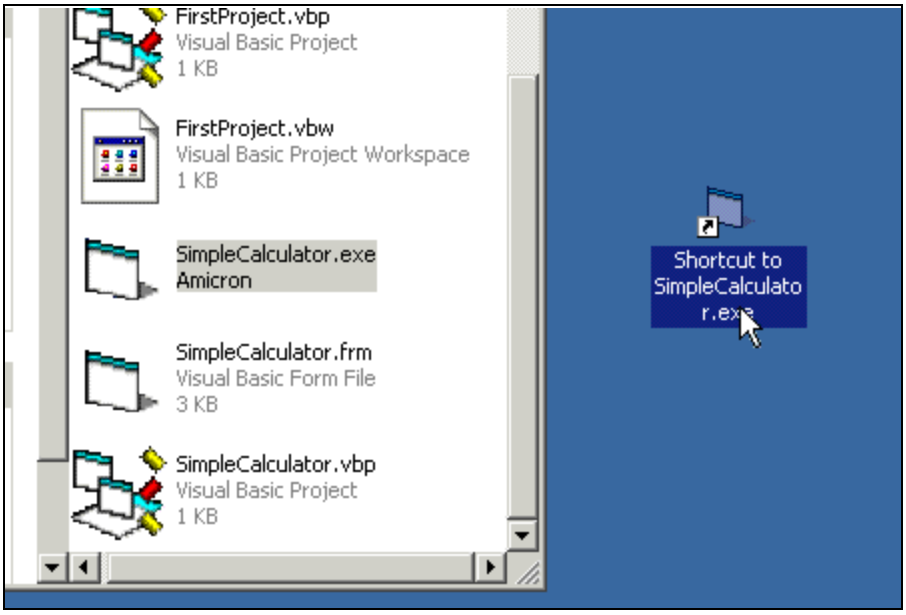
And there we go! Our program runs right from My Computer or Explorer, or wherever you happen to be in Windows. It's now its own standalone Windows application. You don't have to load up Visual Basic to run your program.



You can right click on the executable file and drag it to the desktop and create a shortcut to it.



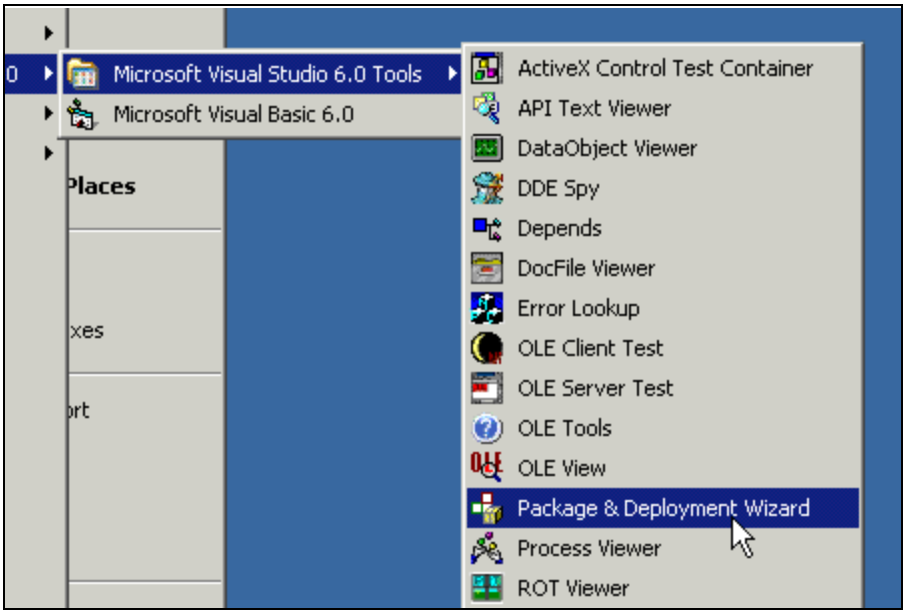
So now your simple calculator is on your desktop. And we can click on it once to rename it. Anytime we want to run this, we can double click on it here on the desktop.



Now one of the things that students always ask me in my classroom is, “Can you share your program?” “Can I take that ‘exe’ program that we just created and give it to a buddy on a floppy disk and say, ‘Here, you can run this.’?”

The answer is – it depends. If he has any other Visual Basic programs on his machine, chances are he has all the necessary files. You may have to give him some additional files, some “dll’s” and some other control files that he might not have on his computer – again, depending on what’s on his machine right now.

If you look in Microsoft Visual studio 6.0 Tools, you will find **a Package & Deployment Wizard**



This guy basically takes your program and packages it up with all the necessary files that you need to distribute your program. Feel free to run through this Package & Deployment Wizard if you'd like to. You only need to bother with it if you want to share your Visual Basic program with other people. You can try giving them your "exe" file first and see if it works, but if not, you might need to run through this.

Keep in mind that since you have Visual Basic on your computer, you don't need to do that, you can run just the "exe" because you have all the necessary files on your machine.

Lesson 9. Review

Let's review what we've learned in today's class.

- We learned about **Visual Basic** and about Windows programs in general.
- We created a couple of different programs.
- We created a simple program to display a message on the screen.
- We built a simple calculator.
- We learned basic commands such as "**msgbox**".
- We learned about different **controls**, **command buttons**, **text boxes**, and **labels**.
- We learned about decision-making, the "**if-then statement**" for example.
- Some real basic **error handling**.
- The "**on error resume next statement"**.
- We learned about **compiling** your program into a standalone executable so you can run it anywhere you happen to be in Windows.
- We talked briefly about **distributing** your program to other users.

Tell us what you think. Log on to **www.599cd.com/Survey** and take a short survey about this course.

> **RICK'S NOTE:** I really do enjoy getting surveys from you! Make sure you visit the web page above and fill out the survey for this class. Let me know if I've moved too fast, and whether or not I covered material that was helpful to you!

Take your **skills check** quiz at **www.599cd.com/Test**. If you pass, you can print out a Certificate of Completion.

What's next? Visit **www.599cd.com** for our complete list of courses.

Need Help? Visit **www.599cd.com/TechHelp** for technical assistance.

Make sure you're on our **Mailing List**. Go to **www.599cd.com/MailingList** for details.

Contact Us. If you have any questions, go to **www.599cd.com/Contact** for information on how you can contact us by phone, email, or live online chat.

Don't forget to visit our **User Message Forums** online at: **www.599cd.com/Forums**. You can chat with our instructors, other users, and even Richard too. You can ask us all of your questions, get answers, and tell us what you thought of our class.

Amicron Computing
PO Box 1308
Amherst NY 14226 USA
www.599cd.com