Visual Basic for Kids

3. Your First Visual Basic Project

Review and Preview

In the first two classes, you learned about **forms**, **controls**, **properties**, and **event procedures**. In this class, you're going to put that knowledge to work in building your first simple Visual Basic project. You'll learn the steps in building a project, how to put controls on a form, how to set properties for those controls, and how to write your own event procedures using a little BASIC.

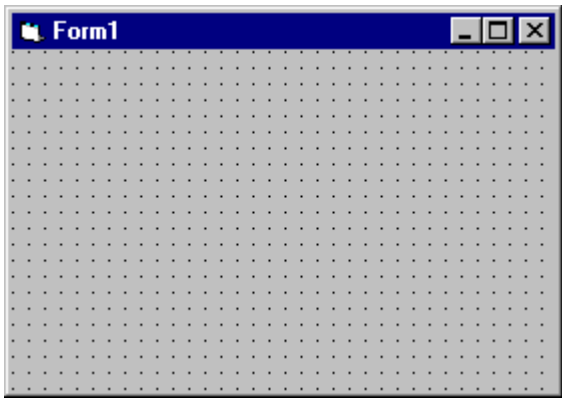
Steps in Building a Visual Basic Project

There are three primary steps in building a Visual Project:

1. Place (or draw) **controls** on the form.
2. Assign **properties** to the controls.
3. Write **event procedures** for the controls.

Each of these steps is done with Visual Basic in **design** mode. Recall the mode is displayed in brackets in the Visual Basic main window title bar.

Start Visual Basic. Find the form that appears when it starts. Click on that form. It should look something like this:



Before building our project, let's review a few 'Windows' techniques. Two things you need to do with controls (the form is a control, remember) is to move them and resize them.

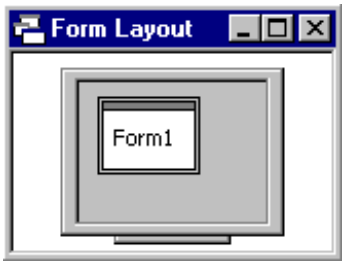
VB4:

- To move the form in a **VB4** project, left-click the title bar area. While holding the left mouse button down, move (or drag) the form to its desired position. Release the mouse button when the form is in the desired position. This is the position the form will have on the screen when your project begins.

- To resize a form in **VB4**, move the cursor to an edge or corner of the form. When a little 'double-arrow' appears, click and drag the corresponding edge or corner to its desired position.

VB5, VB6:

- If you are using **VB5** or **VB6**, the form itself cannot be moved - it is fixed in a window of its own. Placement of the window in this case is controlled using the **Form Layout Window**:



If the form layout window is not present on the screen, click **View** on the main menu, then **Form Layout Window**. Now, click on the form in the little screen and drag it to the desired position. This establishes the location of the form on your computer monitor when your **VB5** or **VB6** application begins.

- To resize the form in **VB5** or **VB6**, notice the form has 'sizing handles' on each edge and each corner. If you move the cursor over one of these handles, a little 'double-arrow' will appear. At that point, you can click and drag the corresponding edge or corner to its desired position. Practice moving and sizing the form. These skills will also be needed next when we place controls on the form.

Placing Controls on the Form

The first step in building a Visual Basic project is to place controls on the form in their desired positions. So, at this point, you must have decided what controls you will need to build your project. Many times, this is a time-consuming task in itself. And, I guarantee, you will change your mind many times. Right now, we'll just practice putting controls on the form.

Controls are selected from the Visual Basic **Toolbox** window. There are two ways to place a control on the form:

1. Double-click the desired control in the toolbox window. The control will be created with a default size and put in the middle of the form.

2. Single-click the desired control in the toolbox window. Now, move the mouse cursor over the form. Notice the cursor changes to a crosshair (+). Place the crosshair where you want the upper left corner of your control to be. Click the left mouse button and hold it down. Now, drag the cursor toward the desired lower right corner of the control. A rectangular outline will be seen. When the outline represents your choice for the control, release the mouse button and the control will appear.

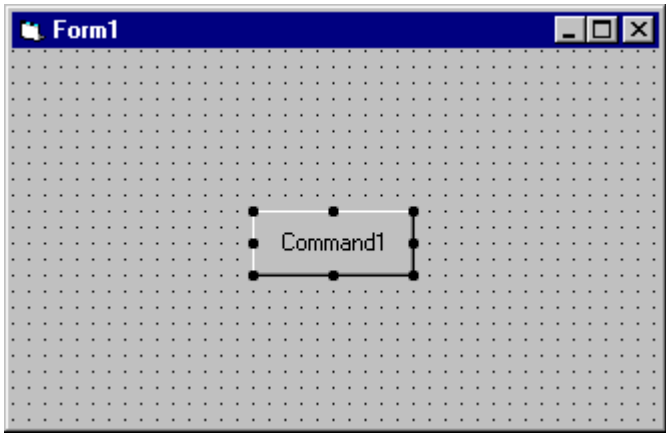
Once, the control is on the form, no matter how it got there, you can still move or resize the control. To **move** a control, left-click the control to select it (sizing handles will appear). Drag it to the new location, then release the mouse button. To **resize** a control, left-click the control so that it is selected. If you move the cursor over one of the sizing handles, a little 'double-arrow' will appear. At that

point, you can click and drag the corresponding edge or corner to its desired position.

Example

Make sure Visual Basic is still running and there is a form on the screen as well as the **Toolbox** (click **View** on the main menu, then **Toolbox** if it is not there). Go to the toolbox and find the **command button** control. It looks like this:

Double-click the control. It should appear in the middle of the form:



Notice the sizing handles around the button. This indicates this is the **active** control. Click on the form and those handles disappear, indicating the form is now the active control. Click on the command button again to make it active. Move the command button around and try resizing it. Make a real big button, a real short button, a real wide button, a real tall button. Try moving the command button around on the form.

Let's put another command button on the form using the second placement method. Go back to the toolbox and single-click the command button control. Move the cursor over the form. You will see a crosshair. Draw the command button on the form using the second method: click when the crosshair is at the upper left corner, then drag the outline until it attains the desired size. Release the mouse button. Notice you can still move this second command button and resize it.

You should become familiar with both ways of placing controls on a form. In time, you will feel more comfortable with one method versus the other. But, always know how to use both. There are times you will only be able to use one method. Spend some time placing controls on the form. Use other controls like labels, text boxes, option buttons, and check boxes. Move them around, resize them. Try to organize your controls in nicely lined-up groups. These are skills that will be needed in building Visual Basic projects.

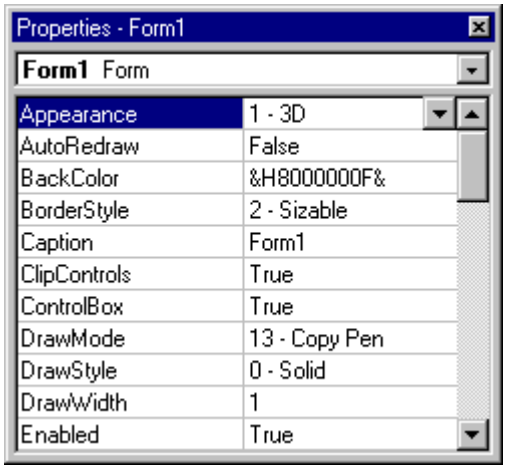
You also need to know how to remove controls from a form. It is an easy process. Click on the control you want to remove. It will become the active control. Press the **Del** (delete) key on your keyboard. The control will be removed. Before you delete a control, make sure you really want to delete it.

Setting Control Properties (Design Mode)

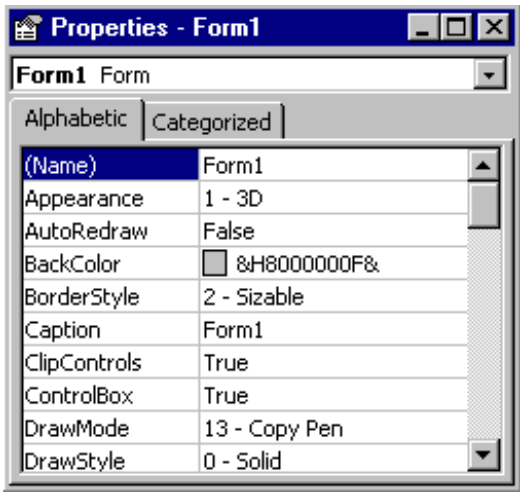
Once you have the desired controls on the form, you will want to assign properties to the controls. Recall properties specify how a control appears on the form. They establish such things as control size, color, what a control 'says', and position on the form. When you place a control on the form, it is given a set of default properties by Visual Basic. In particular, its geometric properties (governing size and location) are set when you place and size the control on the form. But, many times, the default properties are not acceptable and you will want to change them. This is done using the **Properties Window**.

If Visual Basic is not running on your computer, start it now. If it is running, click **File**, then **New Project** (answer No if asked about saving the current form and project). In **VB5** or **VB6**, you will be asked what kind of project you want to start - answer **Standard EXE**. There should be a blank form on the screen. Find the **Properties Window** (press <**F4**> if it's not there):

VB4:



VB5, VB6: Click the **Alphabetic** tab if **Categorized** properties are displayed.



Recall the box at the top of the properties window is the **control list**, telling us which controls are present on the form. Right now, the list only has one control, that being the form itself. Let's look at some of the form's properties.

First, how big is the form? All controls are rectangular in shape and four properties define the size of that rectangle. Scroll down the list of properties and find the **Height** property. This property is the height of the form in a unit of measurement called **twips**. There are 1,440 twips in an inch. So, dividing **Height** by 1,440 will tell you how high your form is in inches. Similarly, the **Width** property gives the form width in twips. Resize the form and notice the Height and Width properties change accordingly. The **Left** property tells you how far the left side of the form is from the left side of your monitor screen. The **Top** property tells you how far down the form is from the top of the screen. Move the form and see these properties change (in **VB5** and **VB6**, you have to move the form in the Form Layout Window). Or, click on the Left or Top property and type in new values in the property side of the list and see the form move on the screen. So, four properties: Left, Top, Width, and Height completely specify the location and size of the form on the computer screen.

Scroll to the **BackColor** property. You probably guessed that this sets the background color of the form. The value listed for that property is probably &H000000F&! You probably don't recognize this, but it is computer talk for gray. We'll look at other ways of setting colors in later classes that makes this a little clearer. To change the BackColor property now, click on BackColor, then click on the drop-down arrow that appears in the property side of the list. (If using **VB5** or **VB6**, next click on the **Palette** tab.) A palette of colors will appear, choose a new color and notice the results.

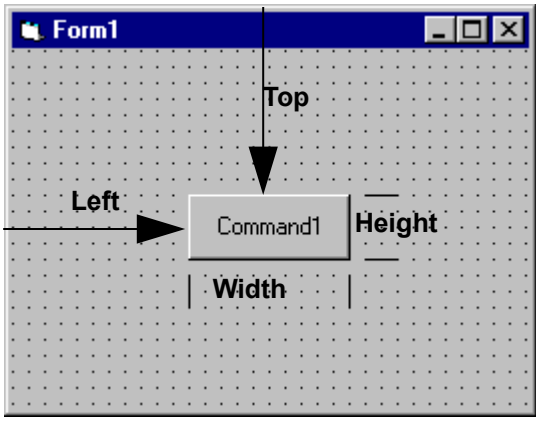
Scroll to the **Caption** property. This property establishes what is displayed in the form's title bar. Click on Caption, then type in something on the right side of the property window. Notice the new Caption appears in the form title bar.

That's all there is to setting control properties. First, select the control of interest from the control list. Then, scroll down through properties and find the property you want to change. Click on that property. Properties may be changed by typing in a new value (like the geometry values and the Caption) or choosing from a list of predefined options (available as a drop-down list, like color values).

Example

Start a new Visual Basic project. A form will appear. Move and resize the form to some desired location and size. Check the **Left**, **Top**, **Width**, and **Height** properties. Set the **BackColor** property. Set the **Caption** property. Place a command button control on the form. Size and place the command button.

Let's look at some of the command button properties. Select the command button in the control list of the properties window. Like the form, the command button is also rectangular. The **Width** property gives its width in twips and **Height** gives its height in twips. The **Left** and **Top** property for controls other than forms are a little different. For a control that is not a form, **Left** gives the position of the left side of the control relative to the left side of the form, not the screen. So, it gives the control position on the form in twips. Similarly, **Top** is the position (in twips) of the top side of the control relative to the top of the form. For a single command button, these properties are:



Another important property for a command button is the **Caption** property. The text appearing on the button is the Caption. It should indicate what happens if you click that button. Change the **Caption** property of your command button. Even though a **BackColor** property is listed for a command button, it cannot be

changed. Put a couple more command buttons on the form. Move and size them. Change their Caption properties.

We have seen that to change from one control to another in the properties window, we can click on the down arrow in the controls list and pick the desired control. A shortcut method for switching the listed properties to a desired control is to simply click on the control on the form, making it the **active** control. Click on one of the command buttons. Notice the selected control in the properties window changes to that control. Click on another button - note the change. Click on the form. The selected control becomes the form. You will find this shortcut method of switching from one control to another very useful as you build your own Visual Basic projects.

Naming Controls

The most important property for any control is its **Name**. Because of its importance, we address it separately. When we name a control, we want to specify two pieces of information: the **type** of control and the **purpose** of the control. Such naming will make our programming tasks much easier.

In the Visual Basic programming community, a rule has been developed for naming controls. The first three letters of the control name (called a **prefix**) specify the type of control. Some of these prefixes are (we will see more throughout the class):

Control	Prefix
Form	**frm**
Command Button	**cmd**
Label	**lbl**
Text Box	**txt**
Check Box	**chk**
Option Button	**opt**

After the control name prefix, we choose a name (it usually starts with an upper case letter to show the prefix has ended) that indicates what the control does. The complete control name can have up to 40 characters. The name must start with a letter (this is taken care of by using prefixes) and can only contain letters (lower or upper case), numbers, and the underscore (_) character. Even though you can have 40 character control names, keep the names as short as possible without letting them lose their meaning. This will save you lots of typing.

Let's look at some example control names to give you an idea of how to choose names. These are names used in the **Sample** project looked at in Class 1 and Class 2. Examples:

frmSample - Form for the Sample project
cmdBeep - Command button that causes a beep
lblPick - Label showing number picked
optBlue - Option button that changes background color to Blue
chkTop - Check box that displays or hides the toy top

This should give you an idea of how to pick control names. We can't emphasize enough the importance of choosing proper names. It will make your work as a programmer much easier.

Setting Properties in Run Mode

To illustrate the importance of proper control names, let's look at a common task in Visual Basic. We have seen one of the steps in developing a Visual Basic project is to establish control properties in design mode. You can also establish or change properties while your project is in run mode. For example, in the **Sample** project, when you clicked on an option button, the **BackColor** property of the form was changed. To change a property in run mode, we need to use a line of BASIC code (you're about to learn your first line of BASIC!). The format for this code is:

```
ControlName.PropertyName = PropertyValue
```

That is, we type the control's name, a dot (same as a period or decimal point), the name of the property we are changing (found in the properties window), an equal sign (called an assignment operator), and the new value. Such a format is referred to as **dot notation**.

The code used to change the **Sample** project form background color to blue is:

```
frmSample.BackColor = vbBlue
```

Notice proper control naming makes this line of code very understandable, even if you don't know any BASIC. It says that background color of the Sample form has been set to blue.

How Control Names are Used in Event Procedures

Another place the importance of proper control naming becomes apparent is when we write event procedures (discussed next). When you put a control on a form, all of the possible event procedures that control can have are added to your project. We have seen that these event procedures are viewed in the code window. The structure for these event procedures is:

Header line: `Private Sub` **`ControlName`**`_EventName()`
[BASIC code goes here]
Footer line: `End Sub`

Note the header line uses the control name. So, with proper naming, we can easily identify each event procedure.

As an example, using **Sample** again, the **Click** event procedure for the **optBlue** control is:

```
Private Sub optBlue_Click()
frmSample.BackColor = vbBlue
End Sub
```

We recognize this is the code that is executed when the user clicks on the optBlue option button. Proper naming makes identifying and reading event procedures very easy. Again, this will make your job as a programmer much easier. Now, let's write our first event procedure.

Writing Event Procedures

The third step in building a Visual Basic application is to write event procedures for the controls on the form. To write an event procedure, we use the code window. Review ways to display the code window in your project. This step is where we need to actually write BASIC code or do computer programming. You won't learn a lot of BASIC right now, but just learn the process of finding event procedures and typing code.

As just mentioned, when you place a control on a form, the event procedures associated with that control become part of the project and can be accessed using the **code window**. Each control has many event procedures. You don't write BASIC code for each procedure - only the ones you want the computer to respond to. Once you decide an event is to be 'coded,' you decide what you want to happen in that event procedure and translate those desires into actual lines of BASIC code. As seen earlier, the format for each event procedure is:

Header line: `Private Sub ControlName_EventName()`
[BASIC code goes here]
Footer line: `End Sub`

The words 'Private Sub' indicate this is a **Sub**routine (another word for procedure) that is **Private** to the form (only usable by the form - don't worry about what this means right now). Developing the BASIC code is the creative portion of developing a Visual Basic application. And, it is also where you need to be very exact. Misspellings, missing punctuation, and missing operators will make your programs inoperable. You will find that writing a computer program requires exactness.

So, the process to write event procedures is then:

- Decide which events you want to have some response to
- Decide what you want that response to be
- Translate that response into BASIC code
- Find the event procedure in the code window
    - Type in the BASIC code

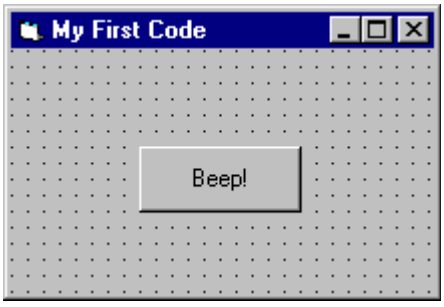
And, it is a process best illustrated by example.

Example

If Visual Basic is not running on your computer, start it and begin a new project.

- Put a single command button on the form.
- Set the **Name** property of the form to **frmFirstCode**.
- Set the **Caption** property to **My First Code**.
- Set the **Name** property of the command button to **cmdBeep**.
    - Set the **Caption** property of the command button to **Beep!**

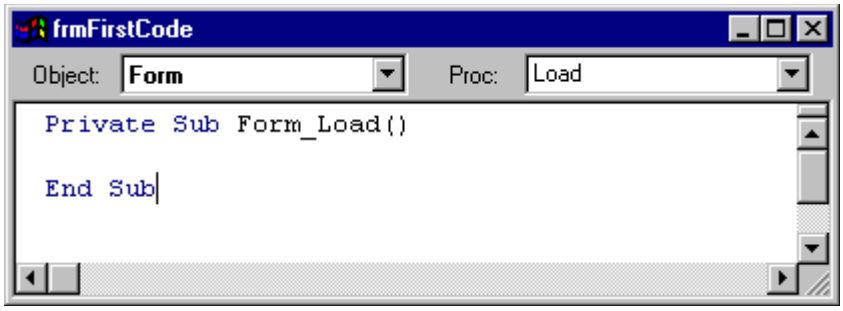
At this point in the design process, your form should look something like this:



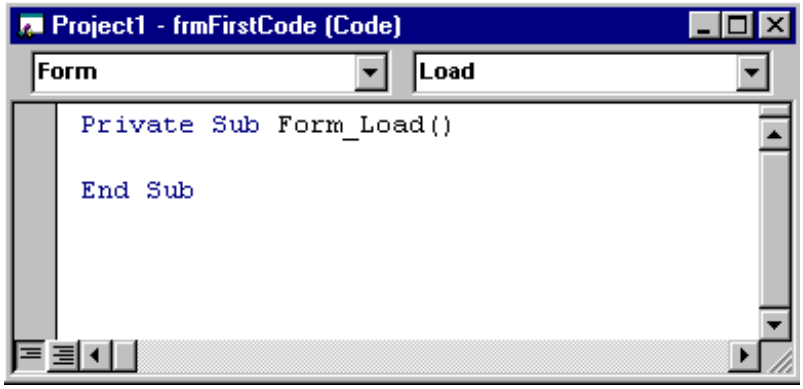
We want to write a single event procedure - the procedure that responds to the **Click** event of the command button. When we click on that button, we want to computer to make a beep sound.

Display the code window (pressing <**F7**> is one way):

VB4:



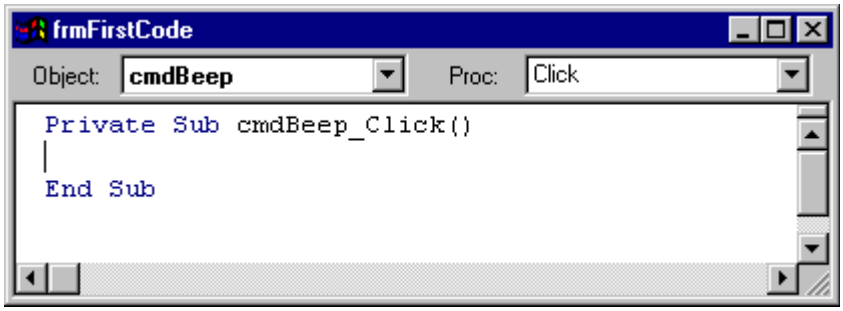
VB5, VB6:



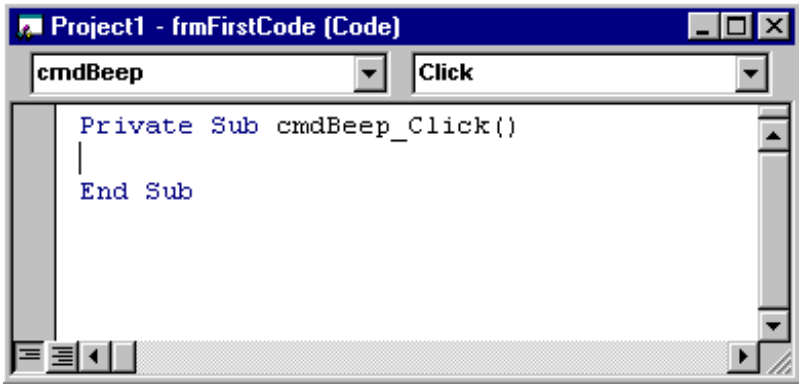
Your code window may not look like this. In the Visual Basic environment, there is a display option called **full-module** view where event procedures are listed one after another, not separately as shown here. If your environment is in full-module view, you need to change it - we will not use it in this course. To get out of full-module view, click **Tools**, then **Options**. Click the **Editor** tab in the displayed window and make sure there is not a check mark next to the **Full-Module View** option. If there is, click the box to remove it.

If the **cmdBeep** object is not shown in the **Object** list, click on that list's drop-down arrow and select cmdBeep (the command button). The code window should now look like:

VB4:



VB5, VB6:



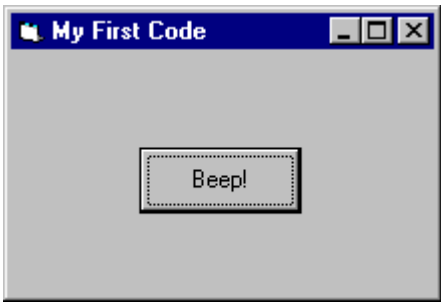
Notice the **Click** procedure for the **cmdBeep** button is now displayed. Many times you will also have to use the **Procedures** list to find the desired procedure - this time, it just happened to be the one displayed (procedures are listed alphabetically). This is where we type the code to make the computer beep.

The code window acts like a word processor. You can type text in the window and use many of the normal editing features like cut, paste, copy, find, and replace. As you become a more proficient porgrammer, you will become comfortable with using the code window. Click on the region between the header and footer lines. Type the single line:

```
Beep
```

This is a BASIC instruction that tells the computer to beep. You have now written your first line of BASIC code.

Your project is now ready to run. **Run** the project (click the **Start** button on the toolbar or press <**F5**>). You may be asked if you want to save some files, say **No** or click **Cancel** for now. The form will appear:



Click the command button. The computer should beep. You caused a **Click** event on the **cmdBeep** control. The computer recognized this and went to the **cmdBeep_Click** event procedure. There it interpreted the line of code (Beep) and made the computer beep. Stop your project. Go back to the code window and find the cmdBeep_Click event. After the Beep line, add this line:

```
frmFirstCode.BackColor = vbBlue
```

Make sure you type it in exactly as shown - remember, computer programs must be exact. Run the project again. Click on the command button. Explain what happens in relation to the control, the event procedure, and the BASIC code. Stop your project.

Summary

You have now finished your first complete Visual Basic project. You followed the three steps of building an application:

1. Place controls on the form
2. Assign control properties
3. Write control event procedures

You follow these same steps, whether building a very simple project like the one here or a very complicated project.

Now, knowing these steps, you're ready to start working your way through the Visual Basic toolbox, learning what each control does. You can now begin learning elements of the BASIC language to help you write programs. And, you can begin learning new features of the Visual Basic environment to aid you in project development. In each subsequent class, you will do just that: learn some new controls, learn some BASIC, and learn more about Visual Basic.