Object Oriented Programming

in Visual Basic .NET

Labs

The lab sessions that accompany Object Oriented Programming in Visual Basic .NET are intended to guide you through the main topics featured in the book as practical exercises. In general, you should work through the labs in the set order. You may also find revision of certain labs is useful as you take on further projects.

Lab1 is a simple orientation exercise designed to familiarize you with the main parts of Visual Studio. It may seem unnecessarily long compared to the other labs, but you should make sure you are comfortable with this lab before you go on to the second and subsequent ones. It covers the fundamental techniques you will need to use when working on any Visual Basic project.
Lab1: Setting up Visual Studio .NET / Creating a simple project

In this lab, you will cover the following topics:

- Starting up Visual Studio .NET and selecting a project type
- Setting up a projects folder for all your Visual Basic work
- Creating a small console program
- Testing and debugging features
- Creating a short Windows program
- Adding and configuring Windows controls
- Creating a Release version and deploying it

1. Starting Up Visual Studio .NET

To proceed, you will need to have Microsoft Visual Studio .NET, or Microsoft Visual Studio .NET 2003 installed on your computer. Provided it is properly installed, you can proceed as follows:

   a) At the Windows desktop, Press the Start button, select the Programs entry in the Start menu, locate the Microsoft Visual Studio .NET (or Microsoft Visual Studio .NET 2003) menu entry, open this and click on Microsoft Visual Studio .NET (or Microsoft Visual Studio .NET 2003). (In future, instructions like this will be written as Start→Programs→Visual Studio .NET→Visual Studio .NET). Visual Studio should display a ‘splash graphic’ as it loads, and then open on the desktop. Visual Studio should open with the Projects page in the central pane as shown in fig L1.1. (If it does not, select the Projects page by clicking on the Projects tab at the top of the central pane).
b) Click on the New Project button in the Projects page. This will open the New Project dialog box (see fig L1.2), from which you can select a type of project to create.

2. Creating a folder for your VB .NET Projects

At this stage, you could simply enter the name of the new project in the Name box and press the OK button to begin. However, since this is your first project, a far more organized approach is to start by nominating the disk folder you want your
projects to be stored in. The default drive and folder is the one Visual Studio is installed in, but if you were to use this, you would be keeping your own programs and the Visual Studio software in the same place. It is better to store your projects on a separate drive or partition of your computer, or on a mapped network drive if possible, since keeping applications software (e.g. Visual Studio) and your own work (e.g. Visual Basic programs) in separate places makes it easier to upgrade the software, and back up your own work.

For example, I store all my Visual Basic .NET projects on drive D: (a separate partition on my main hard disk drive) in a folder called Programs – i.e D:\Programs. Students in my classes store their projects in a similarly named folder on their machine’s H: drive, which is a mapped network drive.

![Figure L1.3: The home folder for my VB .NET programs](image)

To create a new project in a folder of your choice:

a) Click on the Browse button in the dialog box and navigate to the drive and folder you intend to store your projects on (e.g. C:\Programs). Select the folder you want to use or click on the New Folder icon to create one, and then press the Open button.

b) You should now have returned to the New Project dialog box. Make sure the project type selected in the left hand pane is Visual Basic Projects, choose **Console Application** from the right hand pane (see fig L1.4), and enter the name **HelloVB** in the Name box. Now click on OK to start up the new project in Visual Studio.
3. Setting up the skeleton Console Project

Whenever you start a new project in Visual Studio, a skeleton project is created for you. This project contains no working code and default (and therefore meaningless) names are given to constructs within the project. The skeleton Console Project you have just created appears as shown in fig L1.5 in the Visual Studio IDE.

Figure L1.5: The skeleton Console Project

Figure L1.5 shows a Visual Studio that has been reduced in size to fit better on the page, so the central code pane is too small to be of much use while programming.
However, it does show all of the significant features of the IDE and the newly created project. Two panes in particular contain important views of the new project. The central code pane shows all of the code in a new Visual Basic console project, which comprises 1) a Code Module –
```vbnet
Module Module1 ... End Module
```
and 2) the program’s entry point
```vbnet
Sub Main() ... End Sub
```
To the right of the code window is the Solution Explorer window, which shows a tree of components that make up the solution. Here you will find all of the files included in and referenced by your program. Note that the selected file in this pane, Module1, is the file in which the code in the central pane is stored.

Before we add any code to the project, we should rename components of it to be more in keeping with the project we will be writing:

a) Change the name of the code module file by right clicking on the file name (Module1) in the Solution Explorer, selecting Rename from the pop-up menu, and entering a replacement name – Hello.vb.

b) Change the name that will be used in code to reference the module by clicking in the code pane on the line `Module Module1`, and changing this to read `Module Hello`.

Both of these name changes have their own significance in the project you are working on. The file name used for the module (Hello.vb) will enable you to recognize the file that contains the program code in Windows Explorer, so that you could copy the file, send it to someone else or incorporate it in another project. The name used at the top of the module is an Identifier, meaning that it can be used in program code to refer to a specific item – in this case the module that contains the program code. We will learn the significance of this later.

.NET programs can contain more that one code module, and so it is important to let Visual Studio know which module contains the code that is the entry point of the program – i.e. the first line of code to be executed. In this case, there is only one module to worry about, but we have changed its name (to Hello). We need to let Visual Studio know about this:

c) Change the name of the Startup Object for this project. Make sure the Hello entry in the Solution Explorer is selected, and then from the menus select Project→Properties.... In the Properties dialog box that appears, the
**Common Properties/General** page should be open, and you will be able to select the actual module name (**Hello**) from the drop down list under **Startup Object**.

### 4. Adding code to make the program do something

At this stage, you need not be too concerned about how the program you are writing works; you should be concentrating on getting used to using Visual Studio.

a) Enter the code *exactly* as shown in this lab sheet and the project should work. Note that the first two and last two lines are already written, and you need only add the extra lines:

```vbs
Module Hello

Sub Main()
      Dim name As String
      Console.Write("Enter your name: ")
      name = Console.ReadLine()
      Console.WriteLine("Hello {0}", name)
   End Sub

End Module
```

b) Save the project by pressing the button or choosing **Save All** from the **File** menu.

### 5. Testing and Debugging

The program as written can be tested by selecting **Debug** ➔ **Start** from the menus. It will run by opening a console window, displaying the prompt to the user to enter their name and waiting until the user has entered some text and pressed Enter. However, from here, the remainder of the program will execute and the program will terminate very quickly, without giving you a chance to view the results.

To run the program so that you can view its final output before the console window closes:

a) Select **Debug** ➔ **Start without debugging** from the menus

b) Enter your name as prompted and press Enter. The program should run to completion and the console window should remain on screen with a **Press a key to continue**… message
Often, you will find a program will not run at all, or will run but produce the wrong results, or may even crash (terminate unexpectedly and generate a Windows error message). If the program will not run at all, the problem is a syntax error, which means that there is an error in your code that it could recognise and point out to you. However, if it does run and yet contains an error, the best way to find the error is to **debug** the program. This means running the program under the control of the Visual Studio debugger, which allows you to execute one statement at a time, inspect the values stored in variables and nominate specific **break-points** – statements that your program will stop at, and **watch-points** – variables or expressions whose value you can monitor.

To run the program under the control of the Visual Studio debugger:

a) Select **Debug**→**Step Into** from the menus (or press the F11 key). Visual Studio will compile the program and begin to execute it, but will pause on the first line (**Sub Main()**), which will now be highlighted in yellow.

b) Press F10 to execute the current line and move to the next. Note that the line **Dim name As String** was stepped over since it is not an executable line, and that the following statement was highlighted. Each press of F10 will execute the current line in the program.

c) Press F10 another two times. The first press will execute the statement  
**Console.Write(“Enter your name”)** which will write the message in quotes on a console window. The second press will start the execution of the statement - **name = Console.ReadLine()**. This will in turn bring the console window to the front (it has until now been hidden behind Visual Studio). A cursor will be flashing, as the program waits for you to enter a name. Type your name followed by the Enter key. If you now bring Visual Studio back to
the front by clicking on any visible part of it, you can continue to follow the program’s execution.

d) Hover the mouse cursor over the word name in the program code window. A small pop-up window should appear to inform you of the current contents of this variable.

![Code Window]

Figure L1.7: A code-window showing pop-up information (the contents of the 'name' variable)

e) Select **Debug** ➔ **Continue** or press F5 to make the program run to the end.

### 6. End

Having completed this lab, you should be able to:

- Create a new Visual Basic project and save it in its own folder
- Find the Code and Solution windows
- Add program code to the project
- Execute the project and use the debugger to step through lines of program code and inspect variables.