Nesting

Global and Local Variables

From the previous examples, you are familiar with how to declare a variable in VB.NET:

```
Dim X As Integer = 1
```

However, did you realise there are different ways of declaring a variable? Variables can either be declared as **global** or **local**. Global variables can be accessed by other routines and functions, and thus are declared outside of any routine/function. Local variables however, can only be accessed by the routine/function that they are declared in and thus their life expires when the routine/function ends.

Local Variable Example

```
Shared Sub Main
   Dim Name As String
   Name = "Joe"
   WriteLine(Name)
End Sub

Shared Sub UberProgram
   Name = "James"
   WriteLine(Name)
End Sub
```

Global Variable Example

```
Shared Name As String

Shared Sub Main
   Name = "Joe"
   WriteLine(Name)
End Sub

Shared Sub UberProgram
   Name = "James"
   WriteLine(Name)
End Sub
```

The example above illustrates how the local variable **Name** (which belongs to Sub Main()) cannot be accessed by other functions/procedures, in this scenario Sub UberProgram(). Global variables, however (as illustrated on the right), are visible to all methods. Therefore the second example demonstrates how both Sub Main() and Sub UberProgram() can now use the **Name** variable.

Data Types

Learning the data types available in a programming language is important. You have already used two data types – **Integer** and **String** – but there are many others available, i.e. Boolean, Double, Float, Char, etc. Each data type has a specific purpose; using them correctly will increase your program's performance and reduce programming errors. Conduct some research into VB.NET data types and see what other data types are available to use. Some examples are shown below:

Dim A As String = "Hello There!"
Dim B As Integer = 2
Dim C As Decimal = 200.234
Dim D As Char = "C"
Dim E As Boolean = True
Dim F As Date = "23/01/2011"

Nesting

There are times when a programmer requires extended functionality of their iteration and selection techniques. This functionality can be extended by 'nesting' these techniques inside other techniques. For example, an 'if' statement can be nested inside a 'do while loop' and, further still, nest another 'do while loop' inside the 'if' statement. Below are some examples of nesting – try them yourself!

The example below shows an 'if' statement nested inside another:

```
Dim X As Integer = 1
Dim Y As Integer = 2

    If X = 1 Then
        If Y = 2 Then
            WriteLine("I like cake")
        Else
            WriteLine("I don't like cake")
        End If
    ElseIf X = 2 Then
        WriteLine("I like cheese")
    Else
        WriteLine("Invalid Input")
    End If
```

Here is a second example; however, this time it illustrates an 'if' statement inside a 'do while loop':

```
Dim X As Integer = 0
    Do While X < 1
        Dim Y As Integer = 0
        Y = ReadLine()
            If Y = 1 then
                X = X + 1
            End If
    Loop
```

Here is a third example. This example demonstrates a loop inside a loop:

```
Dim X As Integer = 0
    Do While X < 5
        WriteLine("")
        Dim Y As Integer = 0
            Do While Y < 5
                Write("*")
                Y = Y + 1
            Loop
        X = X + 1
    Loop
```