Using Selection

'If' Statements (Selection)

'If' statements are fundamental in all programming languages; they allow selection. In other words, they allow alternative paths in the program to be followed or avoided. The principles of an 'if' statement are the same in all languages; only the syntax differs. 'If' statements use operators to determine if a condition is true or false. Operators include < (less than), <= (less than or equal to), = (equal to), > (greater than) and >= (greater than or equal to). The logic of an 'if' statement is shown below:

```
If [this condition is true] Then
      Do this code
End If
```

Open Notepad or Notepad++ and type the following code (pay attention to the indentation used). Once the code has been typed up, compile and run the program.

```
Imports System.Console
Public Class UsingIFS
        Shared Sub Main()
        WriteLine("Enter a number")
                Dim Y As Integer
                Y = ReadLine()
                If Y > 3 Then
                        WriteLine("Hello World")
                End If
        End Sub
End Class
```

A second (and more) condition(s) can be added using the keyword AND. When using the AND keyword, both conditions must be true before the code contained in the 'if' statement will run. The keyword OR can be used instead to make the code execute if either of two (or more) conditions is true. Amend the previous code so that it includes a second condition; compile and run the code to ensure it works as expected. The syntax for using multiple conditions is shown below:

```
If Y > 3 AND Y < 7 Then
```

Program Output:

```
Enter a number
4
Hello World
```

Test Your Skills

- ✓ *Amend the previous code so that the program writes 'Hello World' if the number is between 5 and 10, and 'Hello Universe' if the number is between 11 and 20.*

If, ElseIf and Else

ElseIf can be used to extend an 'if' statement (by adding additional conditions to check if the previous condition returns false). This method is more efficient than writing multiple separate 'if' statements. The Else keyword can be used to catch any other possibilities if all conditions return false. The logic of an extended 'if' statement is shown below:

```
If [this condition is true] Then
      Do this code
ElseIf [this condition is true] Then
      Do this code
Else
      If all conditions are false do this code
End If
```

Open Notepad or Notepad++ and type the following code (pay attention to the indentation used). Once the code has been typed up, compile and run the program. Note that the instruction Main() will call Sub Main() again, in other words looping the program (this is not ideal, but it works). MsgBox is a simple message box that is displayed to the user.

```
Imports System.Console
Public Class UsingIFS
    Shared Sub Main()
        WriteLine("Enter a number")
            Dim y As Integer
            y = ReadLine()
                If y > 3 Then
                    MsgBox("This is bigger than 3")
                ElseIf y > 6 Then
                    MsgBox("This is bigger than 6")
                ElseIf y > 10 Then
                    MsgBox("This is bigger than 10")
                Else
                    MsgBox("Any other response")
                End If
            Main()
    End Sub
End Class
```

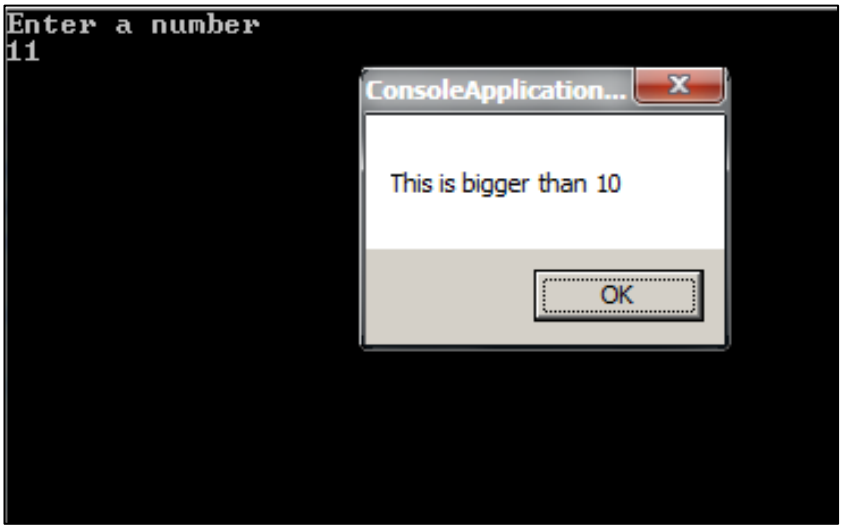
Test Your Skills

- ✓ Once the code has been compiled and run, notice that it does not work as expected, i.e. when entering the value 11, the message 'This is bigger than 3' is displayed, as opposed to 'This is bigger than 10'. See if you can amend the error before continuing.

The previous code did not function as expected because the order of an 'if' statement is important. In the previous example, when the user entered the value 11, the message box 'This is bigger than 3' was displayed to the user, as opposed to 'This is bigger than 10'. This is because an 'if' statement checks all conditions in order (it does not 'pick' the most applicable) and as the first condition is true (10 is bigger than 3), the first message box is displayed. Therefore, to resolve this problem, the 'if' statement must be reversed (as shown below):

```
Imports System.Console
Public Class UsingIFS
    Shared Sub Main()
        WriteLine("Enter a number")
        Dim y As Integer
        y = ReadLine()
        If y > 10 Then
            MsgBox("This is bigger than 10")
        ElseIf y > 6 Then
            MsgBox("This is bigger than 6")
        ElseIf y > 3 Then
            MsgBox("This is bigger than 3")
        Else
            MsgBox("Any other response")
        End If
    Main()
    End Sub
End Class
```

Program Output:



Using CASE

CASE is an alternative method of selection to an 'if' statement. The principles of CASE are similar to that of an 'if' statement, and both techniques can often be used interchangeably. However, CASE is particularly useful when testing the content of a variable. The logic of a CASE statement is shown below:

```
Select Case [Variable Name]
      Case [if this is the 'case' then]
            Do this code
       Case Else
            If all conditions are false do this code
End Select
```

Open Notepad or Notepad++ and type the following code (pay attention to the indentation used). Once the code has been typed up, compile and run the program.

```
Imports System.Console
Public Class UsingIFS
    Shared Sub Main()
        WriteLine("Please enter a name")
        Dim Name As String
        Name = ReadLine()
          Select Case Name
                Case "Joe"
                      MsgBox("Your name is Joe")
                Case "Neal"
                      MsgBox("Your name is Neal")
                Case Else
                      MsgBox("Don't know name..")
            End Select
    End Sub
End Class
```

Program Output: