Visual Web Development

Terry Marris November 2007

17 Classes

We see how to create our own classes.

17.1 The Concept

My friend is:

ann
small - 1.52 metres
female
pretty and generous

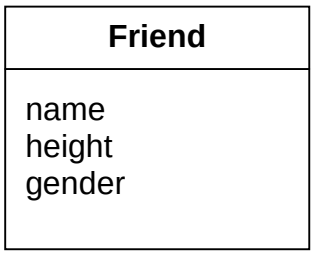
Attributes are derived from what an object has: in our example they include name, height, gender and character.

Three friend objects might be:

name = ann height = 1.52 gender = f	name = sam height = 1.65 gender = m	name = joy height = 1.55 gender = f

Each object is distinct and all share the same attributes, but not necessarily with the same values.

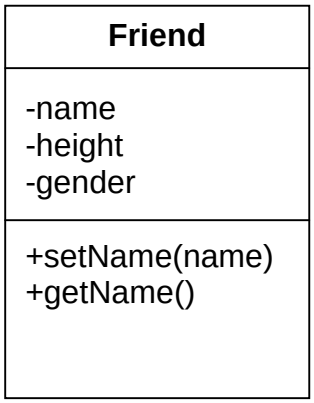
A class represents the collection of similar objects, each sharing the same attributes.



A class has a name (Friend) and a set of attributes (name, height, gender).

17.2 The Specification

The specification view of a class is concerned with the interface between its attributes and the outside world. In general access to attributes is controlled by operations.



Setting operations update the values stored in attributes. Getting operations retrieve the values stored in attributes. setName() allows you to change a friend's name. getName() tells you what a friend's name is.

The - sign indicates that the attributes are private and cannot be accessed by none Friend objects.

The + sign indicates the operations are public and can be used by none Friend objects to access, in a controlled way, the private attributes.

17.3 The Implementation

The class is named Friends. It is usual to use singular names for class names e.g. Friend not Friends. But Friend is a VB word.

```
Public Class Friends
```

The attributes are known as fields or instance variables in an implementation.

```
    Private name As String
    Private height As Double
    Private gender As Char
```

New is a VB word and, in this context, defines a procedure that is responsible for constructing new instances (i.e. objects) of the Friends class.

```
    Public Sub New(ByVal name As String, ByVal height As Double,
                   ByVal gender As Char)
        Me.name = name
        Me.height = height
        Me.gender = gender
    End Sub
```

The Me qualifier is used to distinguish between the field names and the parameter names.

getName() tells you what value is stored in an object's name attribute.

```
    Public Function getName() As String
        Return name
    End Function
```

setName() updates the value stored in an object's name attribute.

```
    Public Sub setName(ByVal name As String)
        Me.name = name
    End Sub
```

toString() returns all the values stored in the fields as one long string.

```
        Public Overrides Function toString() As String
            Return name & ", " & height & ", " & gender
        End Function
    End Class
```

We are obliged to use Override since toString() is used by VB for other classes and we are providing our own specialised version.

17.4 Usage

We create three new instances of the Friends class.

```
Dim myFriend = New Friends("ann", 1.52, "f")
Dim yourFriend = New Friends("sam", 1.65, "m")
Dim ourFriend = New Friends("joy", 1.55, "f")
```

And display them in a text box, each on their own line.

```
txtFriends.Text = myFriend.ToString() + ControlChars.CrLf + _
                  yourFriend.ToString() + ControlChars.CrLf + _
                  ourFriend.ToString()
```

ControlChars.CrLf is a VB way of inserting a newline in text.

Then we update ourFriend's name from joy to jon.

```
ourFriend.setName("jon")
```

And to confirm that the name was changed successfully we get and display ourFriend's name.

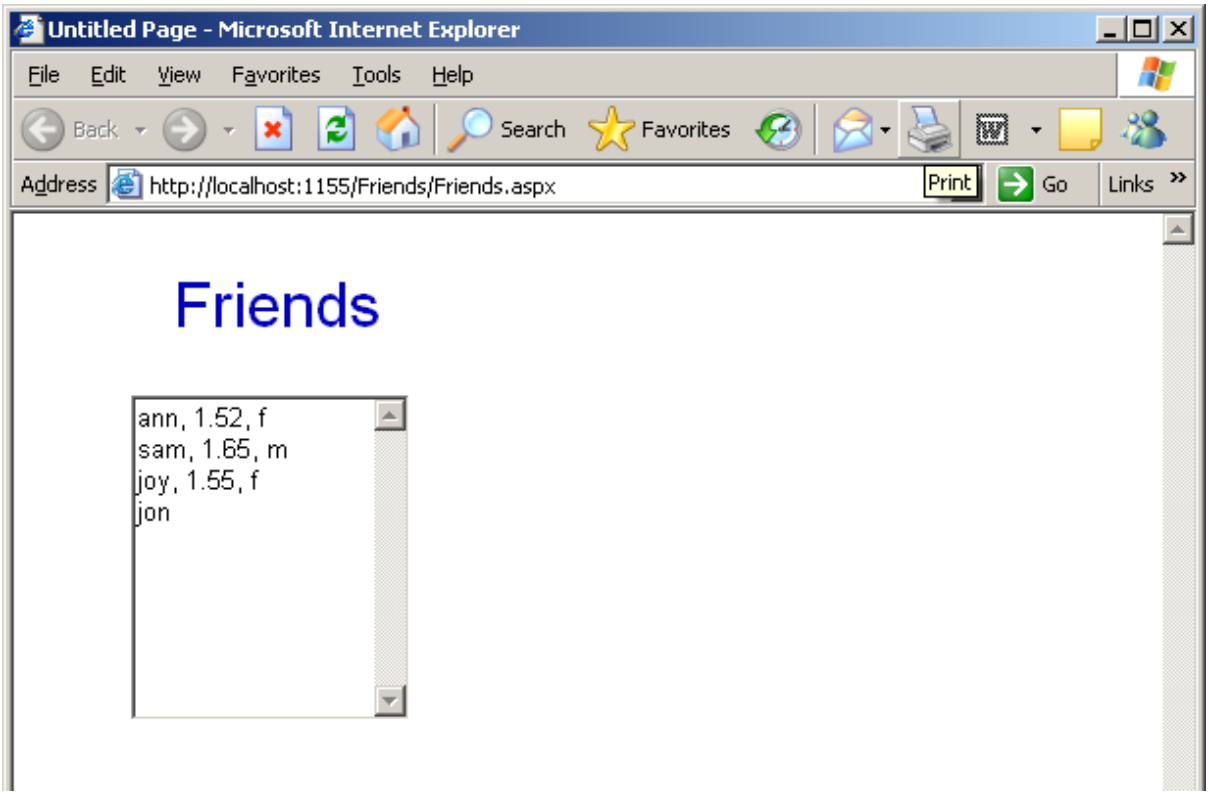
```
txtFriends.Text = txtFriends.Text + ControlChars.CrLf + _
                  ourFriend.getName()
```

The entire code is shown below.

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Public Class Friends
        Private name As String
        Private height As Double
        Private gender As Char

        Public Sub New(ByVal name As String, ByVal height As Double, ByVal
gender As Char)
            Me.name = name
            Me.height = height
            Me.gender = gender
        End Sub

        Public Function getName() As String
            Return name
        End Function

        Public Sub setName(ByVal name As String)
            Me.name = name
        End Sub
        Public Overrides Function toString() As String
            Return name & ", " & height & ", " & gender
        End Function
    End Class

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim myFriend = New Friends("ann", 1.52, "f")
        Dim yourFriend = New Friends("sam", 1.65, "m")
        Dim ourFriend = New Friends("joy", 1.55, "f")
        txtFriends.Text = myFriend.ToString() + ControlChars.CrLf + _
                yourFriend.ToString() + ControlChars.CrLf + _
                ourFriend.ToString()
        ourFriend.setName("jon")
        txtFriends.Text = txtFriends.Text + ControlChars.CrLf + _
                ourFriend.getName()
    End Sub
End Class
```

17.5 The Output

The output from the code shown above in §17.4 above is:



17.6 Exercises

1. Try out the Friends program shown above.

2. Implement and test methods to update and retrieve the values stored in both height and gender.

17.7 Conclusion

There is a lot more to classes than we have covered here. And you can have classes for as many different objects you can think of: bank accounts, library borrowers and weather statistics for example. But we have seen how to encapsulate data in objects and that an object is an instance of a class. Next, we look at dates.