19. Using the Weather Class

Classes are brand new in VB4A and as a result 90% of the earlier exercises should be rewritten to include them. Classes help prevent spaghetti code by giving order to the variables you create by putting them in a unified place and being able to operate on them as you wish. It cuts down on repetitive code and allows for libraries that can be added to projects.

In this case we are making a weather class to hold all the data from Weather Underground. In this exercise to strip it back to the raw structure I use typed in data instead of calling it from the server. That will be added later after the class used becomes more clear.

The process of the class is relatively easy to follow.

Overview – specifics at the end

Make a **new Class Module** named **Weather**.

To pass data to that class in the **Main Activity** create a new instance of the class

```
Dim myweather As weather
```

Pass data through the initialise command to the Class. (these are values for weather conditions)

```
myweather.Initialize( "17","90","ENE","3","-2","15","0")
```

In the Weather class the data is fed in through the initialise sub.

```
Sub Initialize(temp_c,relative_humidity, wind_dir,wind_kph,windchill_c ,
feelslike_c, precip_today_string As String)
```

Then you pass it to the Private variables which are variables that can only be used in the class. Private variables are traditionally prefixed with m_ or just _ at the beginning just to make them obvious.

```
m_temp_C = temp_c
m_relative_humidity = relative_humidity
m_wind_dir = wind_dir
m_wind_kph = wind_kph
m_windchill_c = windchill_c
m_feelslike_c = feelslike_c
m_precip_today_string = precip_today_string
```

Once the private variables have the data you can do what you want with it in subs that you name yourself. This is just an example of using the private variables and creating your own variables to hold data.

```
Sub gettemp_c
Dim temp As String
Dim m_temp As Double
```

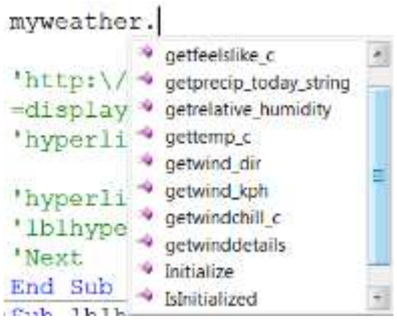
```
m_temp = m_temp_C
If m_temp_C <5 Then
m_temp_C =  m_temp_C &" C - cold"
Else If m_temp_C <10 Then
m_temp_C = m_temp_C &"C - not so cold"
Else If m_temp_C >10 Then
m_temp_C = m_temp_C &"C - Go Running!"
End If
temp = "Temperature =  " & m_temp_C & CRLF  & "Windchill = " &
m_windchill_c &  CRLF & "Feels like = " & m_feelslike_c

Return temp  g
End Sub
```

Return temp holds whatever data you want to go back into the activity. You can use anything as long as it has Return at the beginning.

For example

```
Sub getrelative_humidity
Return m_relative_humidity
End Sub
```

To get the data back in your Activity just use myweather.whatever you want.



```
myweather.
'http:\/
=display
'hyperli

'hyperli
'lblhype
'Next
End Sub
```

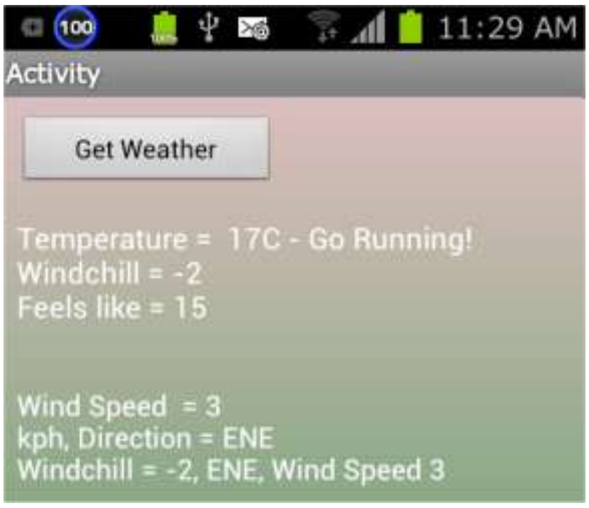
```
lbltemp.text = myweather.getwinddetails
lblwind.text  = myweather.gettemp_c & ", " &
myweather.getwind_dir &", Wind Speed " &
myweather.getwind_kph
```

And that’s it!!!! Code over the page

Activity code

Here is the Activity code for the weather app. Its just a standard program down to the Sub **download**

```
'Activity module
Sub Globals
  Dim btnrun As Button
  Dim lbltemp As Label
  Dim lblwind As Label
End Sub
```



```
Temperature =  17C - Go Running!
Windchill = -2
Feels like = 15


Wind Speed  = 3
kph, Direction = ENE
Windchill = -2, ENE, Wind Speed 3
```

```
Sub Activity_Create(FirstTime As Boolean)
  Activity.LoadLayout("main")
  download
  End Sub
```

```
Sub download
'create an instance of the weather class
Dim myweather As weather
'Pass data to that class
myweather.Initialize( "17","90","ENE","3","-2","15","0")

'Show the output from that class in the labels
lbltemp.text = myweather.gettemp_c
lblwind.text  = myweather.getwinddetails & ",  " & myweather.getwind_dir
&", Wind Speed " & myweather.getwind_kph
End Sub
```

Class Module code

```
'Class module
'I want a class that holds the weather items
Sub Class_Globals
 'internal _private variables used just in this class
Private m_temp_C, m_relative_humidity, m_wind_dir,
m_wind_kph,m_windchill_c, m_feelslike_c, m_precip_today_string As
String
End Sub
```

```
'Initializes the object. You can add parameters to this method if needed.
'data goes in through temp_C and relative_humidity to _temp_C and
_relative_humidity


'Pass the data from the Main Activity to the class in through Sub
intialise
Sub Initialize(temp_c, relative_humidity, wind_dir, wind_kph,
windchill_c, feelslike_c, precip_today_string As String)
'the incoming data is passed to the internal private classes
m_temp_C = temp_c
m_relative_humidity = relative_humidity
m_wind_dir = wind_dir
m_wind_kph = wind_kph
m_windchill_c = windchill_c
m_feelslike_c = feelslike_c
m_precip_today_string = precip_today_string
End Sub
```

```
Sub gettemp_c
Dim temp As String
Dim m_temp As String
m_temp = m_temp_C
If m_temp_C <5 Then
m_temp =  m_temp_C &" C - cold"
Else If m_temp_C <10 Then
m_temp = m_temp_C &"C - not so cold"
Else If m_temp_C >10 Then
m_temp = m_temp_C &"C - Go Running!"

temp = "Temperature =  " & m_temp & CRLF  & "Windchill = " &
m_windchill_c &  CRLF & "Feels like = " & m_feelslike_c
'Return is what is returned, when you call the sub
Return temp
End Sub
```

```
Sub getwinddetails
Dim wind As String
```

```
wind ="Wind Speed  = " & m_wind_kph  & CRLF  & "kph, Direction = "&
m_wind_dir & CRLF &"Windchill = " & m_windchill_c
Return wind
End Sub
```

```
Sub getrelative_humidity
Return m_relative_humidity
End Sub
```

```
Sub getwind_dir
Return m_wind_dir
End Sub
```

```
Sub getwind_kph
Return m_wind_kph
End Sub
```

```
Sub getwindchill_c
Return m_windchill_c
End Sub
```

```
Sub getfeelslike_c
Return m_feelslike_c
End Sub
```

```
Sub getprecip_today_string
Return m_precip_today_string
End Sub
```