

# Scripting Languages G22.3033-002 Summer 2008

## Example solutions for hw02

Assigned Th 5/29/2008, due Fr 6/6 at 4pm. 50 points.

<http://www.cs.nyu.edu/courses/summer08/G22.3033-002/>

### Instructions for example solutions

These are example solutions. Please keep in mind that often, there is not just one correct solution to a question. If you come up with different answers, then it may be that both your answers and these answers here are correct. Of course, these answers here may also contain mistakes. If you spot a mistake, please let me know so I can correct it.

### Example solutions for Homework 2

#### solutions-hw02-1 Properties

- a. There are two properties: `Celsius`, which is defined like a variable, and `Fahrenheit`, which is defined by two property methods `Let` and `Get`. The following code reads and writes both:

```
Dim t As Temperature
Set t = New Temperature
t.Celsius = 25
Debug.Print t.Celsius & " C = " & t.Fahrenheit & " F"
t.Fahrenheit = 25
Debug.Print t.Celsius & " C = " & t.Fahrenheit & " F"
```

These properties encapsulate a temperature value, which is stored internally as `Celsius`, but can be accessed as both `Celsius` and `Fahrenheit`. The user can not tell the difference between the two properties.

- b. To implement a write-once property, I would use a private variable for storing the actual value; a private boolean for keeping track of whether the actual value has already been written; and a pair of `Get` and `Let` methods for reading and writing the value, and checking that the value is written only once. For example:

```
Private myPropValue
Private myPropWritten As Boolean
Sub Class_Initialize()
    myPropWritten = False
End Sub
Public Property Let myProp(val)
    If Not myPropWritten Then
        myPropValue = val
        myPropWritten = True
    End If
End Property
Public Property Get myProp()
    myProp = myPropValue
End Property
```

## solutions-hw02-2 Call-backs

- a. VBA supports call-backs by naming conventions. If a user form contains a control named `myControl`, then the VBA execution engine calls back to subroutine `myControl_Click` when the user clicks a button.
- b. The VBA designers could have implemented call-backs by passing an object on which the runtime system invokes a method. This is a common approach taken by other object-oriented languages such as Java.

## solutions-hw02-3 Associative array (VBA)

```
Option Explicit
Implements StringMap

Private used As Integer
Private keys() As String
Private values() As String

Sub Class_Initialize()
    used = 0
    ReDim keys(2)
    ReDim values(2)
End Sub

Public Function StringMap_Size() As Integer
    StringMap_Size = used
End Function

Public Function StringMap_Contains(key As String) As Boolean
    Dim i As Integer
    For i = 0 To used - 1
        If keys(i) = key Then StringMap_Contains = True: Exit Function
    Next i
    StringMap_Contains = False
End Function

Public Property Let StringMap_Data(key As String, val As Variant)
    Dim i As Integer
    For i = 0 To used - 1
        If keys(i) = key Then values(i) = val: Exit Property
    Next i
    If UBound(keys) = used - 1 Then ReDim keys(2 * used)
    keys(i) = key
    values(i) = val
    used = used + 1
End Property

Public Property Get StringMap_Data(key As String)
```

```

    Dim i As Integer
    For i = 0 To used - 1
        If keys(i) = key Then StringMap_Data = values(i): Exit Property
    Next i
End Property

```

## solutions-hw02-4 Dendrogram (VBA)

You can find the full example solution in the following Powerpoint presentation:

<http://www.cs.nyu.edu/courses/summer08/G22.3033-002/solutions-hw02-4.ppt>

Here, we just describe the most important parts. The GUI is illustrated in the questions. The code sheet requires two event handlers, for the two buttons:

```

Private Sub cmdDraw_Click()
    Dim cvs As Canvas
    Set cvs = New Canvas
    Const dpi As Double = 72
    cvs.xbase = dpi * Cdbl(txtXbase.Value)
    cvs.xscale = dpi * Cdbl(txtXscale.Value)
    cvs.ybase = dpi * Cdbl(txtYbase.Value)
    cvs.yscale = dpi * Cdbl(txtYscale.Value)
    Dim spec As String, x As Double, y As Double
    spec = txtSpec.Value: x = 0: y = 0
    DrawDendrogram cvs, spec, x, y
End
End Sub

Private Sub UserForm_Click()
    End
End Sub

```

The handler code assumes that you have renamed the different form elements appropriately. For example, the text box for the specification is called `txtSpec`. To do the renaming, right-click on a form element, select “properties” from the context menu, then edit the “(Name)” field. While you are there, you should also initialize the “Value” properties with reasonable defaults.