CSCI-GA.3033.003
Scripting Languages

08/06/2012
Objects in VBA
Properties, Call-backs
Outline

• Classes and properties
• Visual programming and callbacks
Defining Classes

Properties
- Public color As String
- Public weight As Double

Methods
- Public Function pluck() As String
  pluck = Me.color & " apple"
  End Function
- Public Sub prepare(how As String)
  Dim dish As String
  Select Case how
    Case "slice"
      dish = "salad"
    Case "squeeze"
      dish = "juice"
    Case Else
      dish = "dessert"
  End Select
  Debug.Print "yum, " & pluck() & " " & dish
  End Sub
Documentation of OO in VBA

- MSDN library
  - Development tools and languages
  - Visual Studio 6.0
  - Visual Basic 6.0
  - Product documentation
  - Using Visual Basic
  - Programmer’s guide
  - Part 2: What can you do with Visual Basic
  - Programming with objects
Defining Properties

- **Syntax**
  - Property Get \( id([\text{arg}^*]) \) ... End Property
  - Property Let \( id([\text{arg}^*,] \text{ arg}) \) ... End Property
  - Property Set \( id([\text{arg}^*,] \text{ arg}) \) ... End Property

- **Remarks**
  - To return value from **Get**, assign to \( id \)
  - **Let** assigns regular value, **Set** assigns object
  - Extra arguments, if any, are for indexing
  - Each routine can be **Private** or **Public**
  - **Public** variable is just shortcut for routines
Inheritance

Concepts

Superclass/ Base class/ Interface

Implements/ “is a” relationship

Subclasses/ Derived classes

Can add additional methods

Can add additional properties
Subtyping Example

Dim someFruit As Fruit
If ... Then
    Set someFruit = New Apple
Else
    Set someFruit = New Banana
End If
' compiler knows that the method exists,
' even if it doesn’t know which version
' will get called
someFruit.prepare("slice")
Inheritance in VBA

Class module “Fruit” (Interface)

Would be abstract in other languages

“Implements” in subclass identifies superclass

Override private with mangled name

Class module “Apple” (subclass)
Pure Interface Inheritance

- Interface inheritance (in VBA):
  - Subclass must provide own implementation for all methods in superclass
  - Can use object of subclass where object of superclass is expected

- Implementation inheritance (not in VBA):
  - Subclass can leave some methods from superclass unchanged
  - Those method implementations are reused

- To reuse code in VBA, must call it by hand
Missing OO Features in VBA

• Implementation inheritance
• Constructors
  – Can write `Class_Initialize()` method
  – Or write subroutine in separate module to allocate new instance and initialize it
• Static fields and static methods

⇒ OO features changed in VB.NET
Outline

• Classes and properties
• Visual programming and callbacks
Using Dialogs

- Dialog = window to request user input
- User form = module defining dialog

```
Load frmLemonStar
frmLemonStar.Show
MsgBox "done, number of lines was " & _
    frmLemonStar.txtNumberOfLines.Text
Unload frmLemonStar
```

- Loading is optional
- Name of module
- Display dialog box, wait for user input, run event handlers
- Unloading is optional
- Retrieve user input
Hungarian Notation

- Convention: start identifier with indication of type

  - chk  Check box
  - cmd  Command button
  - frm  User form
  - fra  Frame
  - lst  List box
  - cmb  Combo box
  - mnu  Menu
  - opt  Option button
  - lbl  Label
  - txt  Text box
Dialog Example

(done, number of lines was 20)
Drag&Drop Dialog Design

- Right-click on project → Insert → User form
- View → Properties Window
  - Change name, caption, maybe width/height
- View → Toolbox
  - Drag and drop controls onto user form
  - Rename, resize, set initial value
  - Set “default” or “cancel” property for buttons
- Right-click on form → Tab order
  - Order when user “tabs through” controls
Writing Event Handlers

Sub paintLemonStar(nLines As Integer)
    Dim n As Integer, i As Integer
    n = nLines - 1
    For i = 0 To n
        Dim l As Shape
        Set l = ActiveWindow.Selection.SlideRange.Shapes.AddLine(
            BeginX:=300, BeginY:=200 + i * 100 / n,
            EndX :=400, EndY :=300 - i * 100 / n)
        l.Line.ForeColor.RGB = RGB(i * 255 / n, i * 255 / n, 0)
    Next i
End Sub

Private Sub cmdPaint_Click()
    paintLemonStar CInt(txtNumberOfLines.Text)
    Hide
End Sub

Private Sub cmdCancel_Click()
    End
End Sub

callbacks with mangled names

Auxiliary subroutine

retrieve user input
Code Sheets

- Right-click → View code
  - Just like other modules
  - Contains event handlers as well as regular subroutines

- Double-click on control
  - Goes to handler, creating it if necessary
  - Caution: handler name not changed automatically with control name!

- Dynamic dialog: assign properties at runtime
  - E.g., `Visible=True` reveals hidden controls
  - `UserForm_Initialize` sets default, e.g. for list box
### Call-backs

#### VBA code

- `frmLemonStar.Show()`

#### Interpreter

- `cmdPaint_Click()`
- `return from handler`
- `return from "Show"`
- `wait for user input`
- `edit text box`
- `click button`

#### User
Common Controls and Events

• Controls:
  Label, TextBox, ComboBox, ListBox, CheckBox, OptionButton, ToggleButton, Frame, CommandButton, TabStrip, MultiPage, ScrollBar, SpinButton, Image

• Events:
  Change, Click, DblClick, Enter, Exit, Initialize, KeyPress, SpinDown, and many more

• In editor for user form code sheet:
  • Left drop-down list: control objects on this form
  • Right drop-down list: events on that object
### Callback Mechanisms

<table>
<thead>
<tr>
<th>Language</th>
<th>Method Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBA form</td>
<td>Subroutine in form with mangled name</td>
</tr>
<tr>
<td>VBA class</td>
<td><code>WithEvent</code> / <code>RaiseEvent</code> statements</td>
</tr>
<tr>
<td>Java</td>
<td>Pass object on which to call method</td>
</tr>
<tr>
<td>Perl, Python, JavaScript</td>
<td>Pass anonymous function (lambda)</td>
</tr>
<tr>
<td>C, C++</td>
<td>Pass function pointer</td>
</tr>
<tr>
<td>C++</td>
<td>Pass object on which to call “()” operator</td>
</tr>
<tr>
<td>SmallTalk</td>
<td>Pass code block</td>
</tr>
<tr>
<td>PHP</td>
<td>Pass name of function as string</td>
</tr>
</tbody>
</table>
Events on Classes

• In event source class $id_{src}$:
  - Public Event $id_{event}(arg*)$
  - RaiseEvent $id_{event}(expr*)$

• In event sink module:
  - WithEvents $id_{handler} As id_{src}$
  - Sub $id_{handler}._id_{event}(arg*) ... End Sub
Reusing Dialog Boxes

• Don’t write “open file” dialog from scratch!

• `Application.Dialogs(id).Show`
  – E.g., `id = xlDialogOptionsGeneral`
  – To find others: Help→Visual Basic→Search “built-in dialog argument”

• `Display` instead of `Show` prevents handlers
  – Return value: -2=Close, -1=OK, 0=Cancel, >0 other command buttons
  – Retrieve user input from controls with `.Value`
How to Learn a Language

I. Use peers & gurus
II. Install tools
III. Read tutorial
IV. Find language & library reference
V. Read example programs
VI. Write example programs:
   I/O, types, control flow, libraries
VII. Understand error messages
VIII. Practice
### Common VBA Mistakes

<table>
<thead>
<tr>
<th>Error</th>
<th>Message</th>
<th>Common cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime error ‘91’</td>
<td>Object variable or With block variable not set</td>
<td>Missing “Set” before object assignment</td>
</tr>
<tr>
<td>Compile error</td>
<td>Invalid character</td>
<td>Missing space before “_” at end of line</td>
</tr>
<tr>
<td>Compile error</td>
<td>Expected: =</td>
<td>Parentheses around arguments, missing “Call”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>And many more</td>
</tr>
</tbody>
</table>
Suggestions for VBA Practice

• hw02 gets you points, but you may want to do more at your own leisure

• Powerpoint
  – Center shape on slide
  – Draw object model
  – Plot a polynomial

• Excel
  – Create graph, set fonts and grid preferences
  – Generate email from name+title+address sheet
Last Slide

• First homework due today at 6pm
• Second homework on the course website

• Today’s lecture
  – Properties
  – Call-backs

• Next lecture
  – Associative arrays
  – Regular expressions
  – Basics of Perl