CS 5142
Scripting Languages

8/04/2013

Objects in VBA
Properties, Call-backs

Administrative Issues

- Waiting for course to be “in the system”
  - Should happen soon
- Put your name, netid, major on the sign-up sheet today
- Don’t turn in your homework yet
  - We will use CMS if we can before Friday
  - Otherwise, email to the TA

Outline

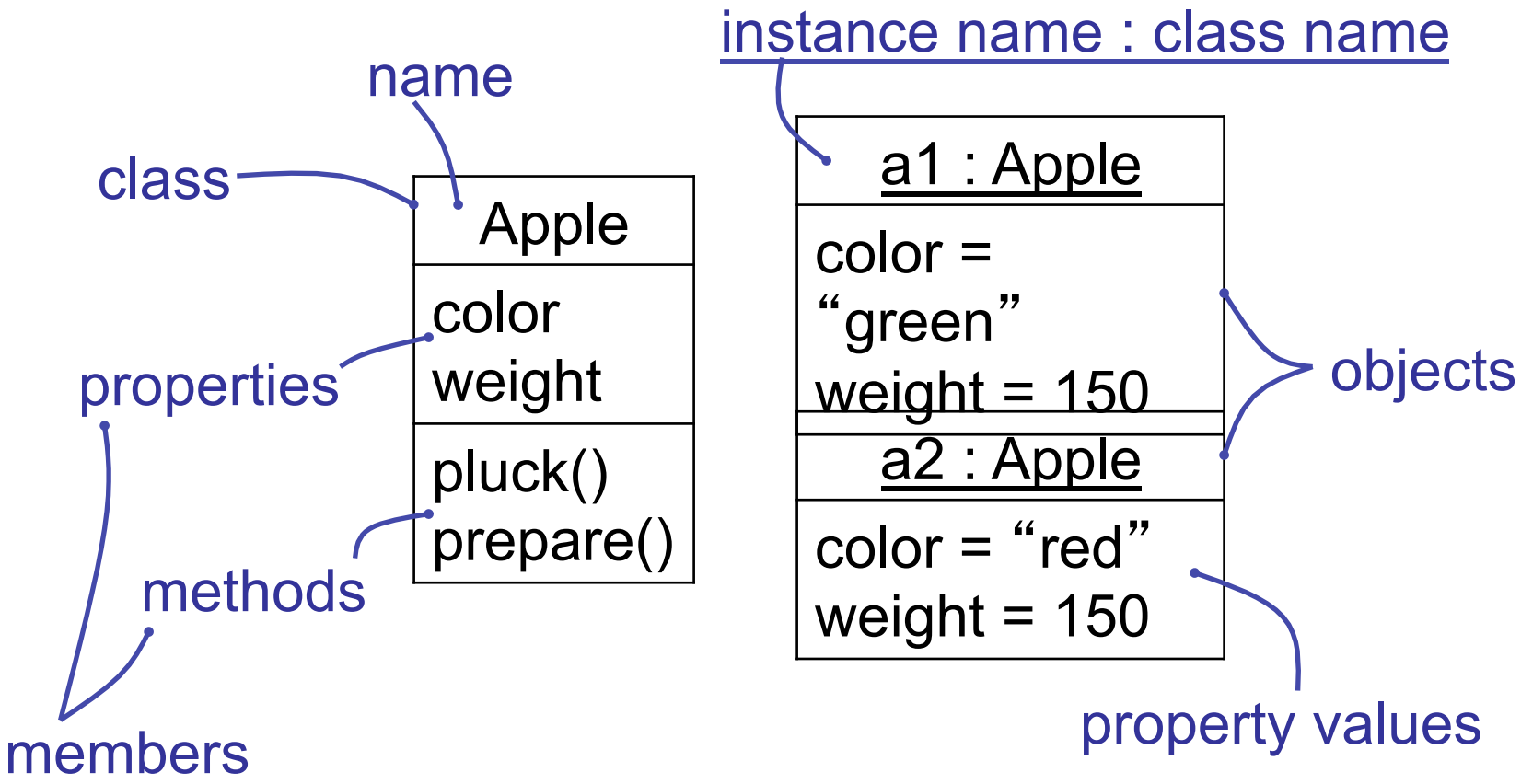
- Using objects in VBA
- Application-embedded scripting

Using Objects

```
Dim a1 As Apple          ' declare variables a1 and a2
Dim a2 As Apple          ' of class Apple
-------------------------------------------------------
Set a1 = New Apple       ' allocate new objects, assign
Set a2 = New Apple       ' references (need "Set" keyword)
-------------------------------------------------------
a1.color = "green"       ' set property, differently
a2.color = "red"         ' for a1 and a2
-------------------------------------------------------
a1.prepare("slice")      ' call method, passing
a2.prepare("squeeze")    ' string parameter
```

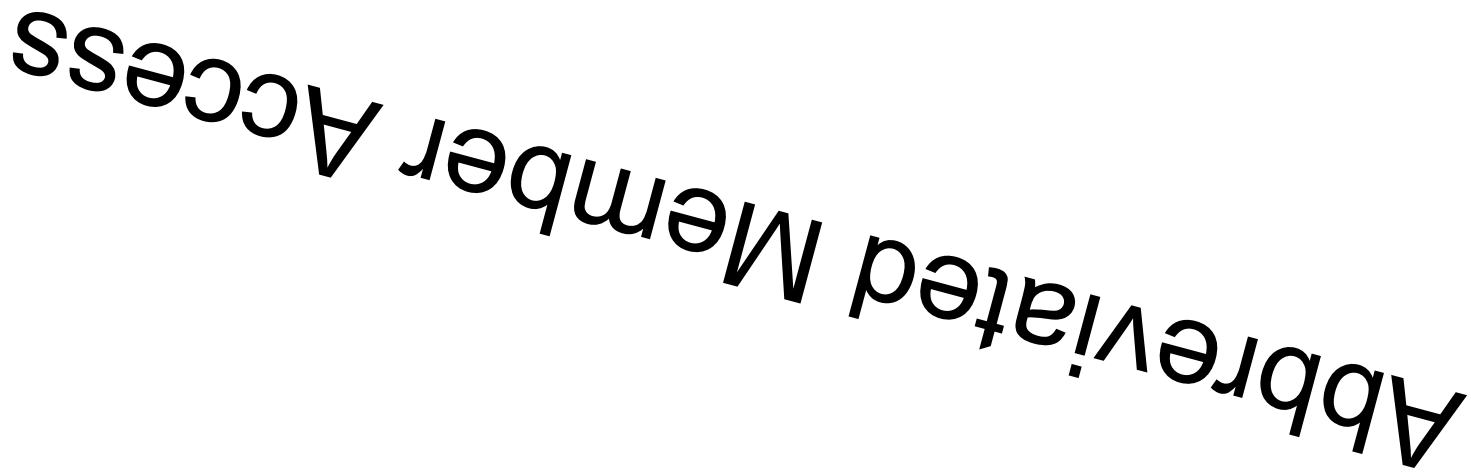
Concepts

Classes and Objects



Abbreviated Member Access

```
With ActiveWindow.Selection.SlideRange.Shapes.Title
   .Flip (msoFlipVertical) ' call method
   .Rotation = 15          ' assign to property
End With
```



Properties vs. Fields

- Both: dot notation look&feel
  - Writable: **`a1.color = "red`**”
  - Readable: **`Debug.print a1.color`**
- Properties only: active (associated behavior)
  - E.g., update graphical representation
- Properties only: may be indexed, like arrays
  - **`cake.ingredient("topping") = a1`**
- Other languages with properties:
  - E.g., PHP, Delphi, C#

Let’s Write Some Code

Common Uses of Properties

Simple (field-like)

- Visual update
- Invariant checking
  - Filter illegal values
  - Read-only
  - Copy on write
- Logging

Indexed (array-like)

- Collections
  - Resizable array
  - Hash map
- Persistence
  - File
  - Database
  - Cookie

Collections

```
Dim col As Slides
Set col = ActivePresentation.Slides
Dim i As Integer
Debug.Print "for-loop, indexed property access"
For i = 1 To col.Count
  Debug.Print col.Item(i).Name
Next i
Debug.Print "for-loop, default property access"
For i = 1 To col.Count
  Debug.Print col(i).Name
Next i
Dim s As Slide
Debug.Print "for-each loop"
For Each s In col
  Debug.Print s.Name
Next s
```

Soap-box

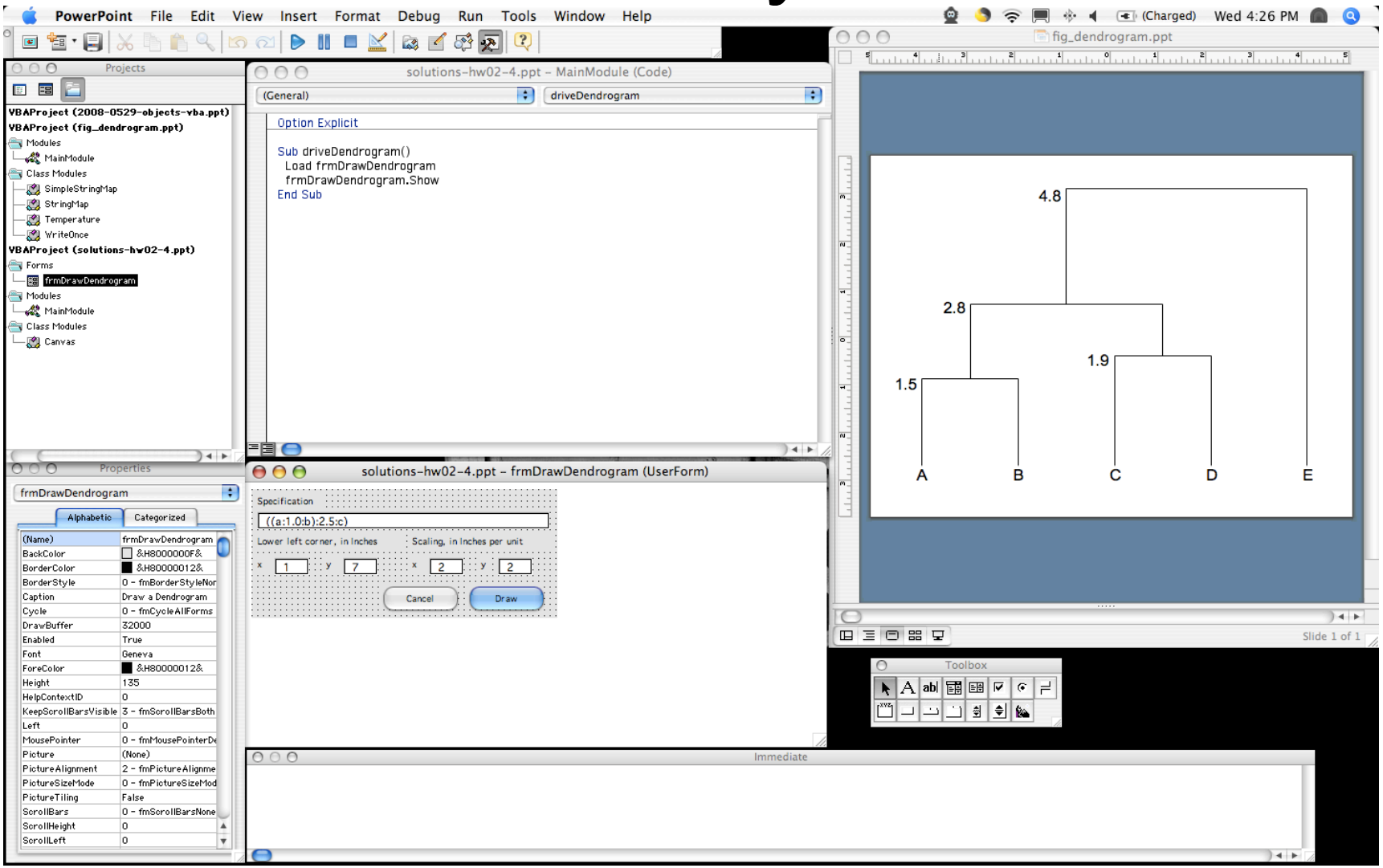
Progressive Disclosure

- We only looked at how to *use* classes, but not how to *define* classes
  - That's sufficient for most VBA tasks!
- Language design encourages this:
  - Learn small subset of language to do most important tasks
  - Learn a little more to do a little more
- *Progressive disclosure* user experience

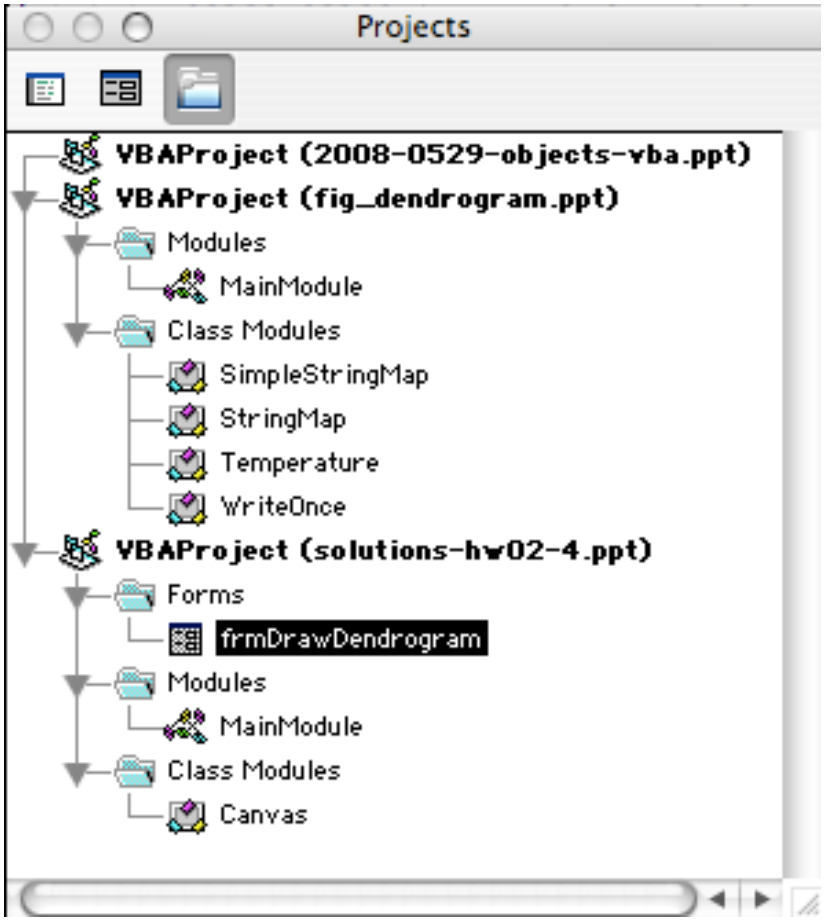
Outline

- Using objects in VBA
- Application-embedded scripting

Where Is My Code?



Structure of a VBA Application



- Projects
  - Files: .ppt, .xls, .doc
  - Normal.dot for Word
  - Personal Macro Workbook for Excel (View→Window→Unhide)
- Modules
  - Regular module
  - Class module
  - Form with code sheet
- Subroutines
  - Function or Sub (Macro)

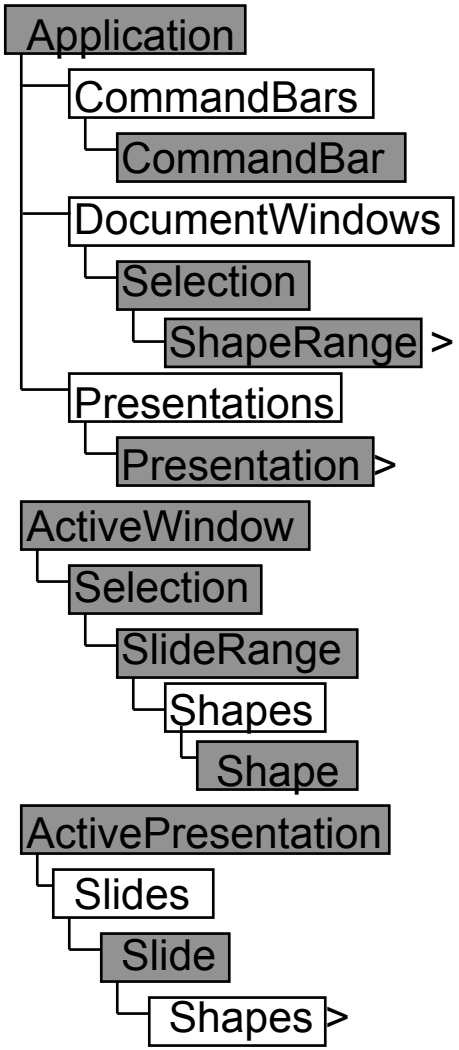
Scopes and Visibility

- Project dependencies
  - Visual Basic Editor→Tools→References
  - Then, can call across projects: **`project.module.subroutine`**
  - Cyclic references are not allowed
- Visibility
  - "Instancing" property of class module (Private or PublicNotCreatable)
  - Public/Private modifiers of declarations

Let’s Write Some More Code

Reference

Powerpoint Object Model



- The complete object model is much larger
- See Visual Basic help in editor
- Also in MSDN library:
  - →Office development
  - →Microsoft Office 2003
  - →Office 2003
  - →VBA reference
  - →Powerpoint help
  - →Object model

Object Model

- Object-oriented API for embedded scripts
- Other examples:
  - Object models for other Microsoft apps
  - DOM = document object model for XML

Administrative

Last Slide

- First homework due Friday at 6pm

- Today's lecture
  - Using objects
  - The A in VBA
- Next lecture
  - More Properties
  - Call-backs