AUTODESK UNIVERSITY 2013

Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®

Lee Ambrosius – Autodesk, Inc.

CM1560-L Visual Basic for Applications (VBA) is a programming environment that allows you to automate tasks using the Visual Basic programming language. This hands-on lab explores how to access and use the VBA integrated development environment (VBAIDE) and work with the Autodesk AutoCAD object model. You learn to work with data, manipulate AutoCAD software objects, get input from a user, store values, work with basic conditionals, and other programming concepts. This is a beginner-level class for VBA programming and is not something you want to take if you already have experience with VBA programming.

Learning Objectives

At the end of this class, you will be able to:

- Create and load a VBA project file from the VBA integrated development environment
- Identify the classes, properties, and methods that are available in the AutoCAD object library
- Create and modify objects in the current drawing
- Collect input from the user via the command line or UserForm

About the Speaker

Lee is a Principal Learning Content Developer on the AutoCAD® team at Autodesk and has been an AutoCAD user for over 15 years in the fields of architecture and facilities management. He has been teaching AutoCAD users for over a decade at both the corporate and college level. He is best known for his expertise in programming and customizing AutoCAD-based products, and has 15+ years of experience programming with AutoLISP®, VBA, Microsoft® .NET, and ObjectARX®. Lee has written articles for AUGI® publications and white papers for Autodesk on customization. He is the author of several books on AutoCAD and has been an active technical editor for the AutoCAD Bible series.

Twitter: *http://twitter.com/leeambrosius*

Email: *lee.ambrosius@autodesk.com*

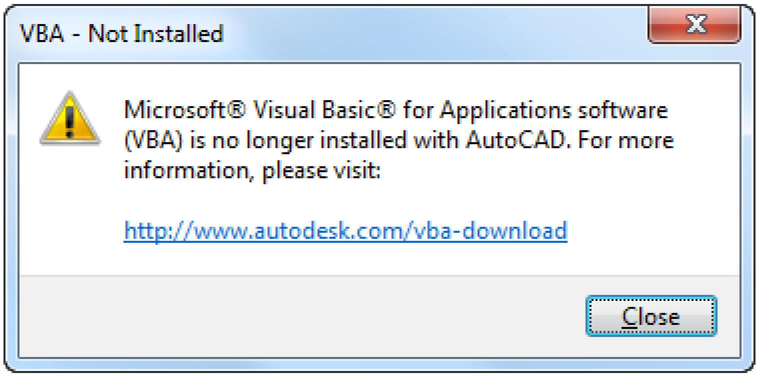
Blog: *http://hyperpics.blogs.com*

Contents

1 Getting Started

Everything you need to get started using VBA in AutoCAD is pretty much provided for you by Autodesk, but in more recent releases you do need to perform one additional step. The VBA development and runtime environment is not available as part of the AutoCAD installation download or on the physical media, but must be download separately from *autodesk.com* and then installed.

You can tell if the VBA environment has been installed by trying to execute one of the VBA related commands in AutoCAD. If the environment has not been installed, the following message box is displayed:



Microsoft® Visual Basic® for Applications software (VBA) is no longer installed with AutoCAD. For more information, please visit:

http://www.autodesk.com/vba-download

Clicking the link in the message box, http://www.autodesk.com/vba-download, opens your default web browser so you can download and install the VBA environment. Once your web browser opens, download the appropriate installer for your AutoCAD release and version.

After the VBA environment has been installed, you can perform the following tasks:

- Load a VBA project
- Create a new VBA project
- Add new and modify existing components of a VBA project
- Add and change library references
- Create and execute macros
- Debug macros and UserForm procedures

VBA Projects

A VBA project contains the code modules and UserForms that define the logic of your custom program. VBA projects are stored in files with the *.dvb* extension. Each VBA project contains at least one component named ThisDrawing, and must reference two libraries.

The ThisDrawing component is a specially named *class module* that represents the active drawing in the AutoCAD environment. Normally, a class module is used to define a new programming object, including its properties and methods.

In addition to the ThisDrawing class module, a VBA project can also contain one or more of the following components:

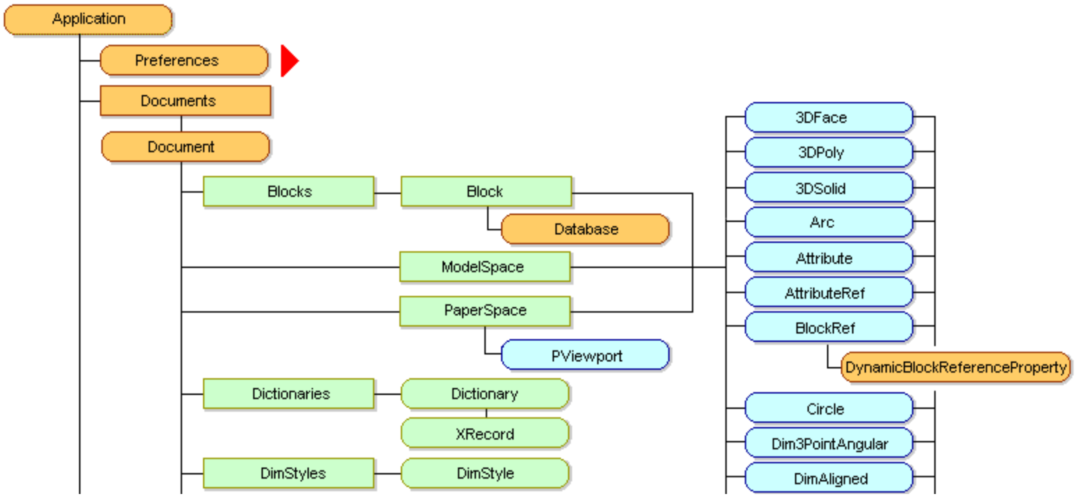
- **Standard code module** – Code modules that are used to store procedures, data types, and variables that can be accessed from anywhere in your project.
- **Class module** – Code modules that define user-defined objects. Class modules are not common to most VBA projects, with the exception of ThisDrawing, but they can be helpful when creating reusable libraries that might be used in multiple projects to avoid procedure naming conflicts and to hide variables and functions from your entire project.
- **UserForm** – A dialog box interface object within a project. Contains controls that the user can interact with to provide values to your custom program. UserForms can be displayed as modal or modeless.

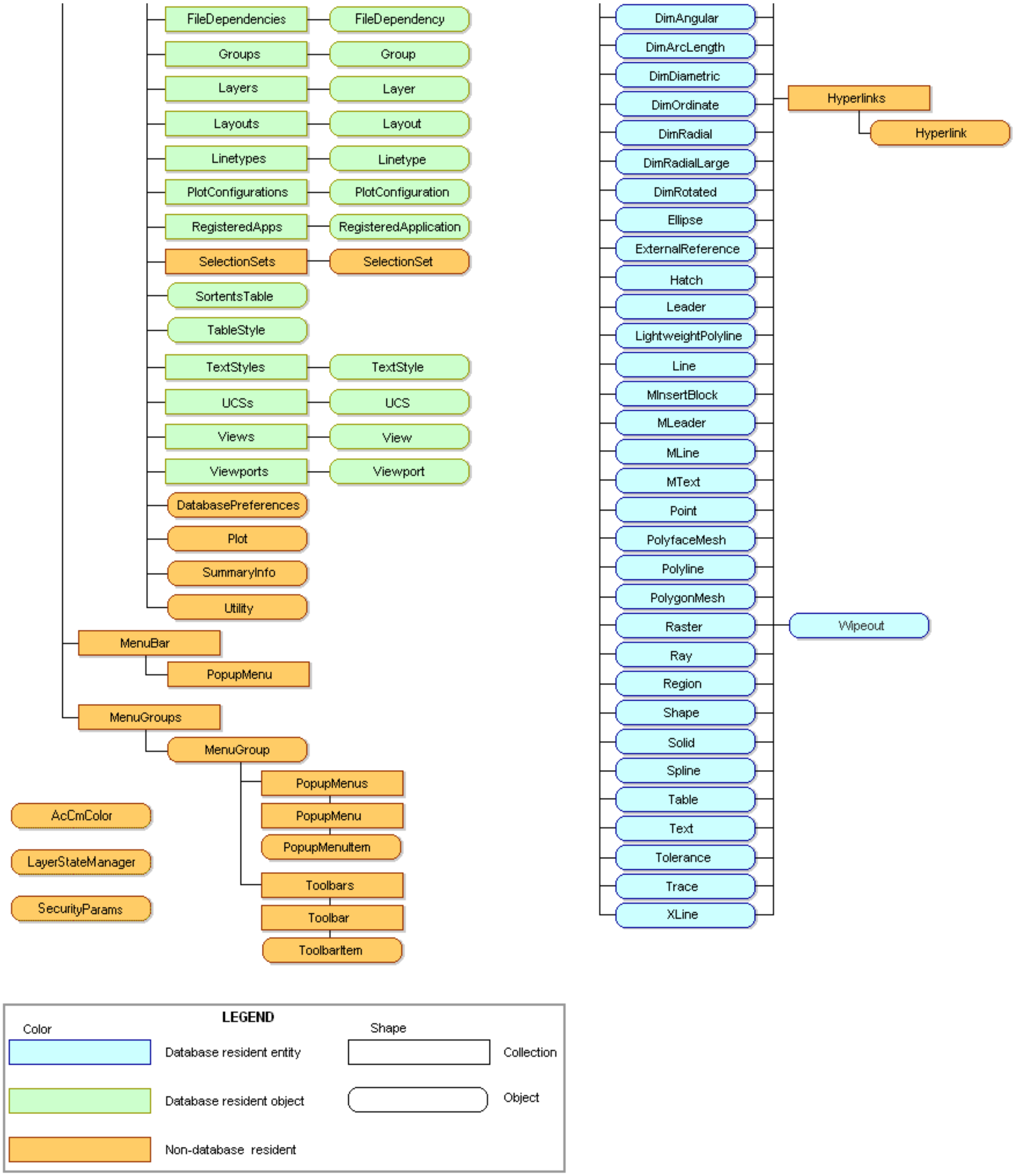
The two libraries that the VBA project must reference are:

- **Visual Basic For Applications** – VBA design time environment
- **AutoCAD <*release*> Type Library** – The object library that defines the interfaces that allow you to work with the AutoCAD application, drawings, and objects within a drawing.

AutoCAD Type Library

The AutoCAD Type Library, also referred to as an object library, contains the interfaces that you can interact with to automate tasks in AutoCAD. Interfaces are commonly also referred to as classes. To help you navigate the classes of and their relationship in the AutoCAD Type Library, you can utilize the AutoCAD Object Model which is a hierarchical structure. The Object Browser of the VBA IDE allows you to quickly see all of the available classes of the referenced libraries in your project, and their properties and methods. You can even directly access a help topic associated with each of the items shown in the Object Browser by selecting the item and pressing F1.





AutoCAD Commands

After the VBA environment is installed, the following commands are available for use:

- **VBAIDE** – Launches the AutoCAD Visual Basic for Applications Integrated Development Environment

- **VBALOAD** – Loads a previously saved VBA project (DVB) file
- **VBAMAN** – Displays the VBA Manager.
- **VBANEW** – Creates a new VBA project (DVB) file and loads it into the current VBA environment.
- **VBAPREF** – Displays the Options dialog box for the VBA environment.
- **VBARUN** – Displays the Macros dialog box which allows you to execute or debug a macro in one of the loaded VBA projects.
- **VBASTMT** – Executes a VBA statement at the AutoCAD Command prompt.
- **VBAUNLOAD** – Unloads a VBA project.
- **-VBALOAD** – Loads a previously saved VBA project file at the AutoCAD Command prompt.
- **-VBARUN** – Executes a macro in one of the loaded VBA projects at the AutoCAD Command prompt.

AutoLISP Functions

In addition to VBA related commands, there are a few AutoLISP functions that are also available upon installing the VBA environment:

- **vl-vbaload** – Loads a previously saved VBA project (DVB) file.
- **vl-vbarun** – Executes a macro in one of the loaded VBA projects.

Both of these functions require you to first make a call to the ***vl-load-com*** function.

VBA Related Terminology

Before you get started working in the VBA environment, you should be familiar with these programming terms:

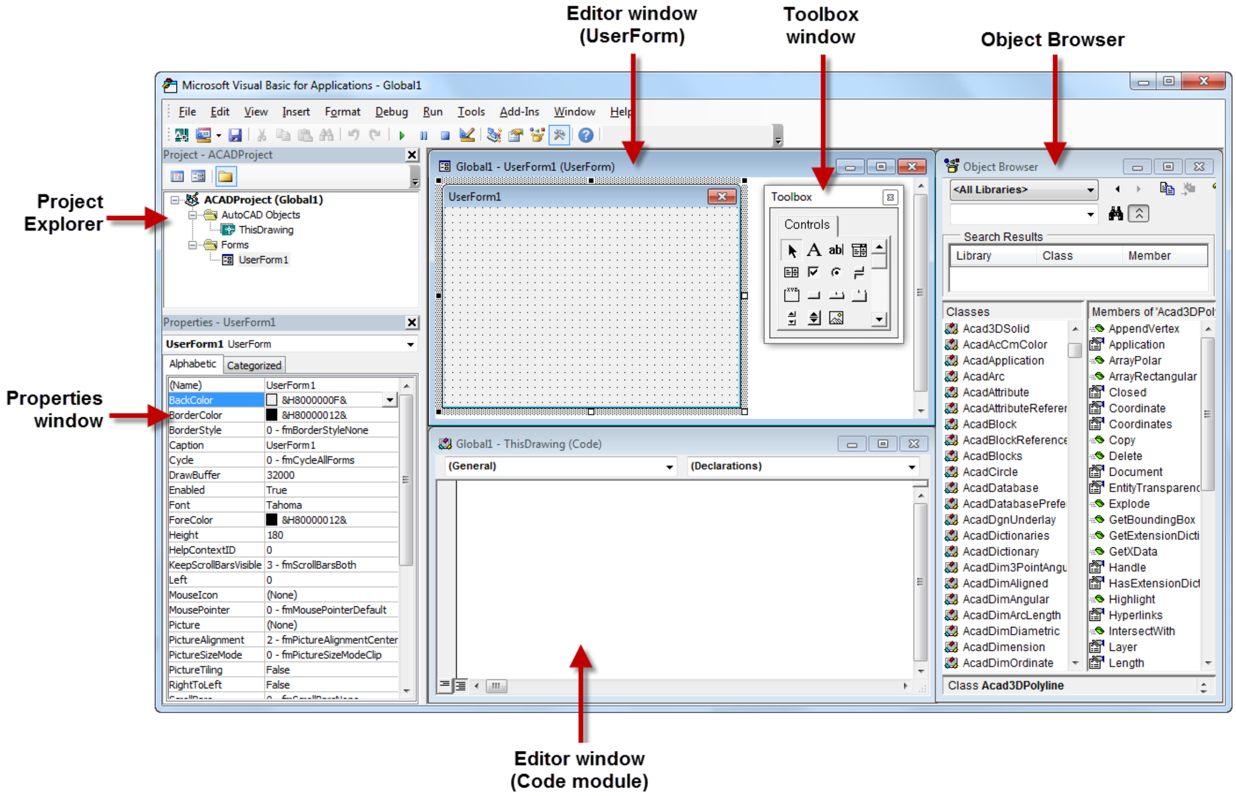
- **Class** – Definition of an object; template that is used to create an instance of an object. Defines the properties and methods that describe and manipulate an object.
- **Object** – An instance of a class that exposes its properties and methods.
- **Property** – Named attribute that describes an object at runtime. Can also be changed at design time for UserForms and controls using the Properties window of the VBA IDE.
- **Method** – Function that performs an action on a specific object.
- **Debug** – Process used to interactively step through source code and evaluate the results in realtime.
- **Statement** – Line of code that is executed when you run a macro or interact with a control on a UserForm.
- **Comment** – Statement that provides information within a code module that is not executed. Comments start with an apostrophe, and anything to the right is ignored during execution.
- **Runtime** – When a procedure within a project is being executed; UserForms and controls can only be modified through code.

- **Design-time** – When the objects within a project can be created and modified using the Visual Basic for Applications IDE.

2 Working with VBA Projects

VBA projects must be loaded into the VBA development environment before they can be edited or used. Editing a VBA project is handled in the Visual Basic for Applications Integrated Development Environment, or commonly known as the VBA IDE. The VBA IDE is displayed by clicking Manage tab ➢ Applications panel ➢ Visual Basic Editor on the ribbon, or entering **vbaide** at the AutoCAD Command prompt.

The VBA IDE contains a number of tools that are used to add and modify the components of a loaded VBA project. The following illustration calls out the main components of the VBA IDE you will want to become familiar with:



(Name)	UserForm1
BackColor	&H8000000F&
BorderColor	■ &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	UserForm1
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	180
HelpContextID	0
KeepScrollBarsVisible	3 - fmScrollBarsBoth
Left	0
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
Picture	(None)
PictureAlignment	2 - fmPictureAlignmentCenter
PictureSizeMode	0 - fmPictureSizeModeClip
PictureTiling	False
RightToLeft	False

- **Project Explorer** – Provides access to the components of all loaded VBA projects. These components include standard and class modules, and UserForms.
- **Properties window** – Provides the ability to view and modify the properties of a selected component from the Project Explorer or an object selected in the form editor window when editing a UserForm.
- **Editor window** – Provides access to the code of a component in a loaded VBA project or the form editor for a UserForm.

- **Toolbox window** – Provides access to the controls that can be placed on a UserForm.
- **Object Browser** – Provides a view into the libraries that are currently referenced to a VBA project.

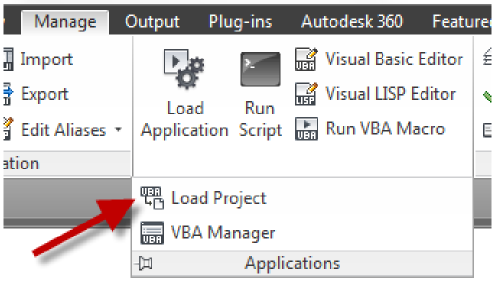
3 Exercises

This section contains all the exercises for this lab and for when you get back to the office.

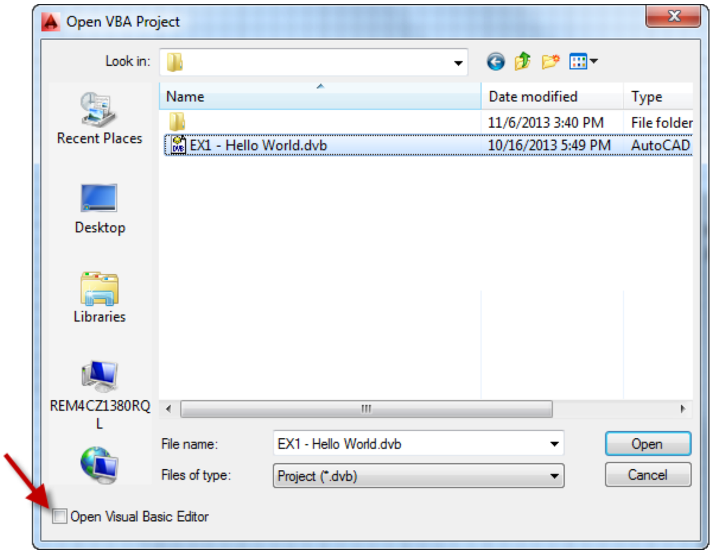
E1 Loading a VBA Project and Executing a Macro

This exercise demonstrates how to load a VBA project file named *EX1 – Hello World.dvb* and execute a macro named *HelloWorld*.

1. In AutoCAD, on the ribbon, click the Manage tab.
2. On the Manage tab, click the title bar of the Applications panel and then click Load Project.



3. In the Open VBA Project dialog box, lower-left corner, clear the Open Visual Basic Editor check box.
4. Browse to *C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®* and select the *EX1 – Hello World.dvb* file. Click Open.



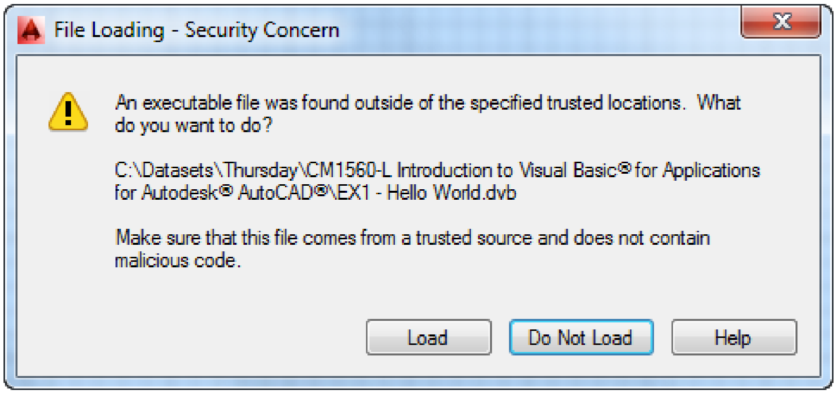
Name	Date modified	Type
	11/6/2013 3:40 PM	File folder
EX1 - Hello World.dvb	10/16/2013 5:49 PM	AutoCAD

File name: EX1 - Hello World.dvb

Files of type: Project (*.dvb)

□ Open Visual Basic Editor

5. In the File Loading – Security Concern dialog box, click Load.



File Loading - Security Concern

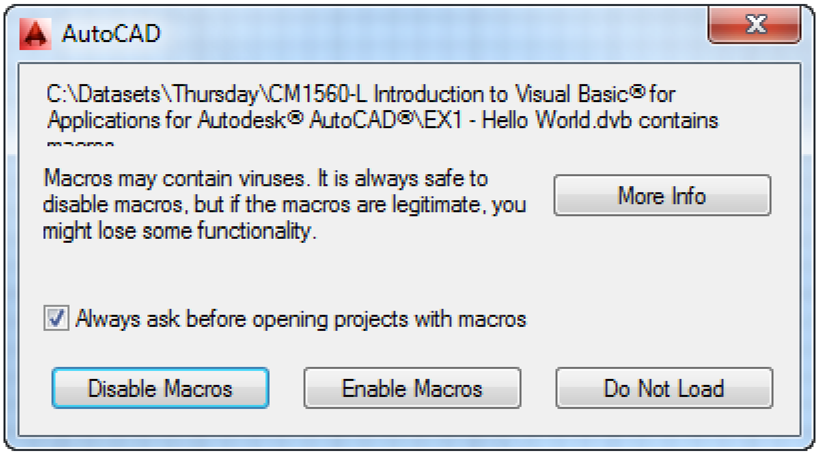
An executable file was found outside of the specified trusted locations. What do you want to do?

C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®\EX1 - Hello World.dvb

Make sure that this file comes from a trusted source and does not contain malicious code.

WARNING! – Do not click the Close or Do Not Load buttons unless you do not want to load the file. Doing so, will require you to close and restart AutoCAD to load the file.

6. In the AutoCAD message box, click Enable Macros.



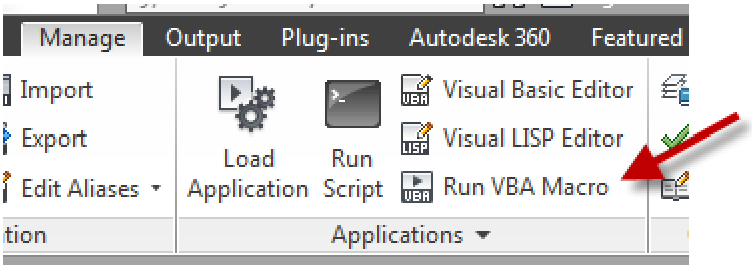
AutoCAD

C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®\EX1 - Hello World.dvb contains macros

Macros may contain viruses. It is always safe to disable macros, but if the macros are legitimate, you might lose some functionality.

☑ Always ask before opening projects with macros

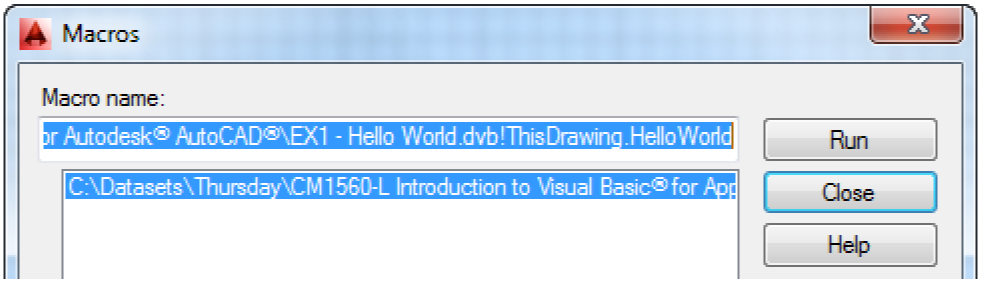
7. On the ribbon, click Manage tab ➢ Applications panel ➢ Run VBA Macro.



8. In the Macros dialog box, select the macro **ThisDrawing.HelloWorld** in the *EX1 – Hello World.dvb* project file if it is not already selected.

You will need to scroll the Macros list to see the full name of the macro since the folder path is so long.

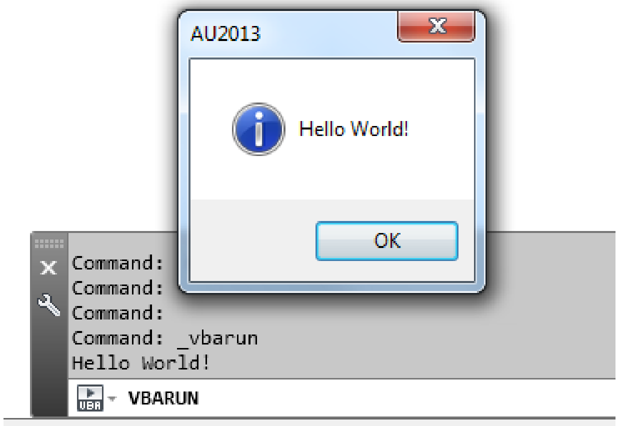
9. Click Run.


Macros

Macro name:

```
or Autodesk® AutoCAD®\EX1 - Hello World.dvb!ThisDrawing.HelloWorld
C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for App
```

The Macros dialog box closes and a message box titled AU2013 is displayed. Along with the message box, the message Hello World! is displayed in the Command Line window.


AU2013

Hello World!

```
Command:
Command:
Command:
Command: _vbarun
Hello World!
```

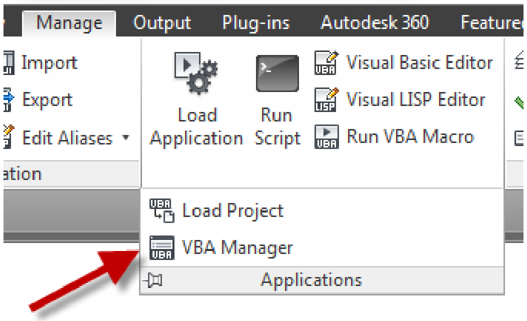
10. Click OK to dismiss the message box.

E2 Working with the Visual Basic Editor and the ActiveX Documentation

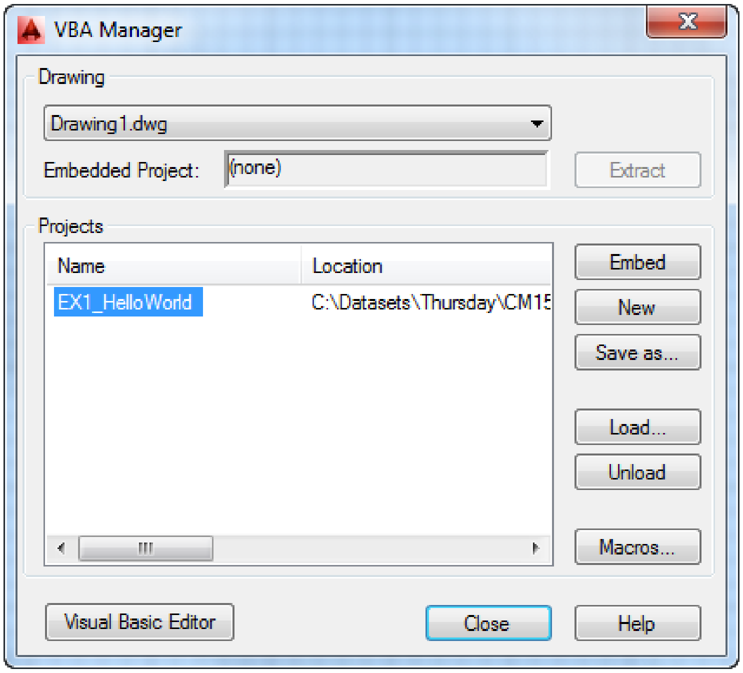
This exercise demonstrates some of the basics of the VBA environment and using the ActiveX Documentation.

1. In AutoCAD, on the ribbon, click Manage tab.

2. On the Manage tab, click the title bar of the Applications panel and then click VBA Manager.

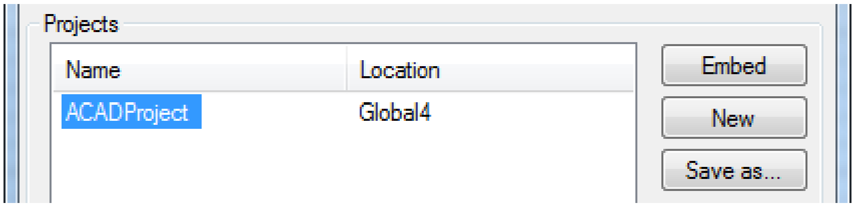


3. In the VBA Manager, Projects list, select EX1_HelloWorld and click Unload to remove the project from the VBA environment.



4. Click New to create a new VBA project.

5. In the Projects list, select ACADProject and click Save As.

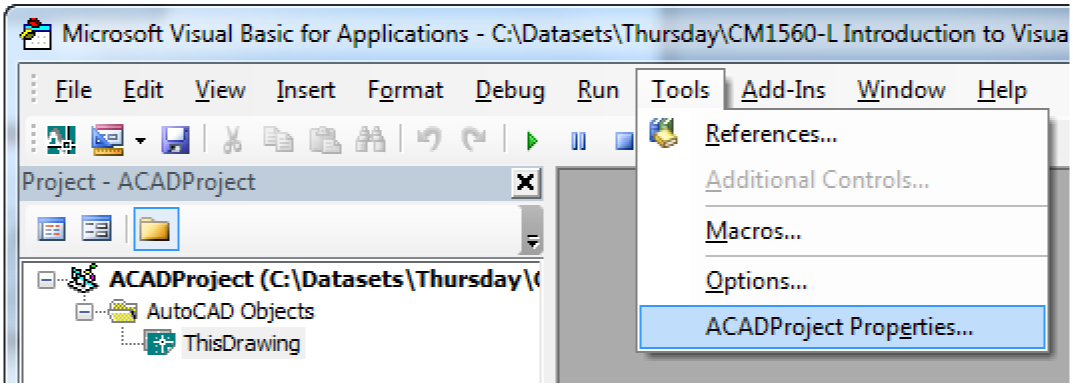


6. In the Save As dialog box, browse to the *C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®* folder.

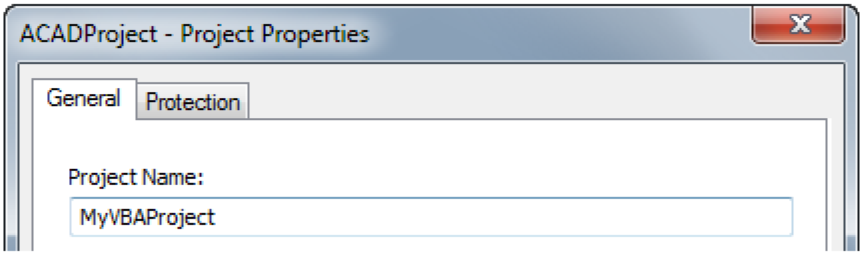
7. In the File Name text box, clear the current text and type **MyVBAProject**. Click Save.

8. In the VBA Manager, lower-left corner, click Visual Basic Editor.

9. On the menu bar, click Tools ➢ ACADProject Properties.

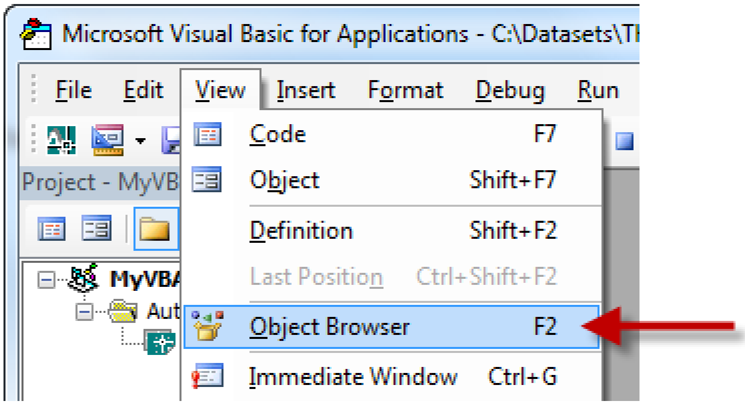


10. In the ACADProject – Project Properties dialog box, clear the value in the Project Name text box and type **MyVBAProject**. Click OK.

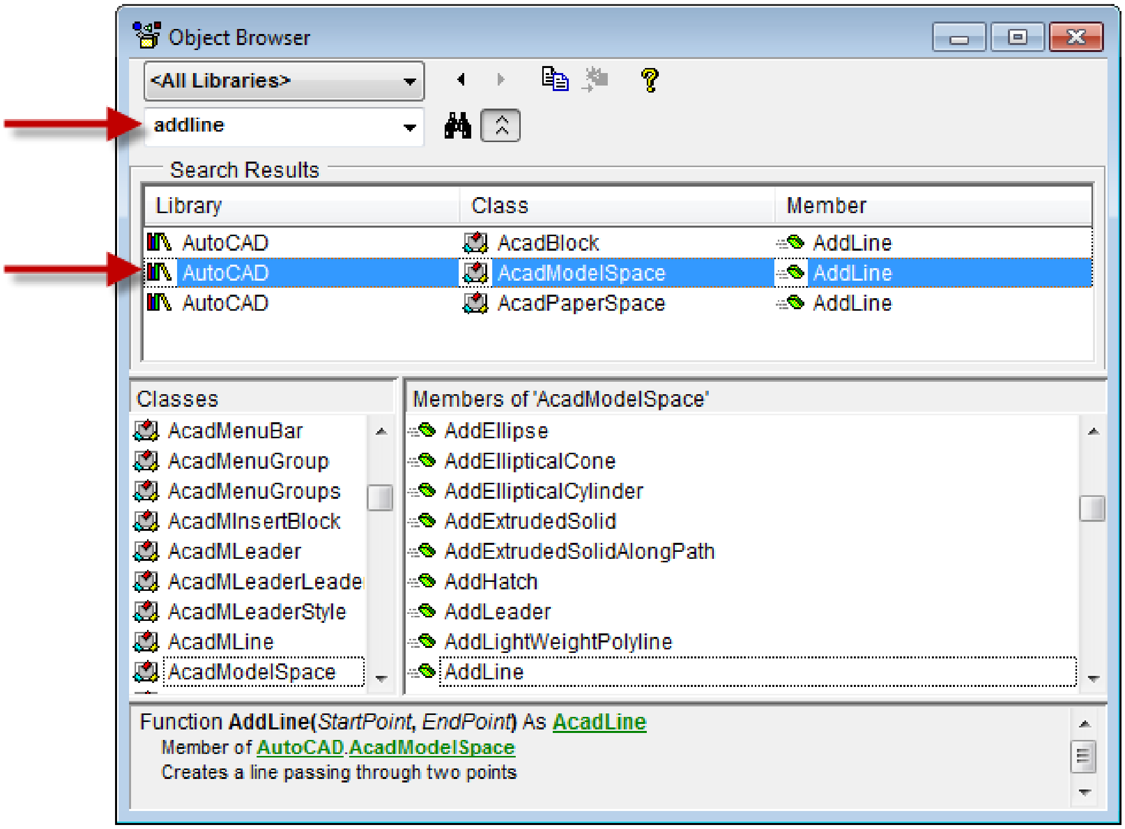


Project Name: MyVBAProject

11. On the menu bar, click View ➢ Object Browser.



12. In the Object Browser, Search box, type **addline** and press Enter.

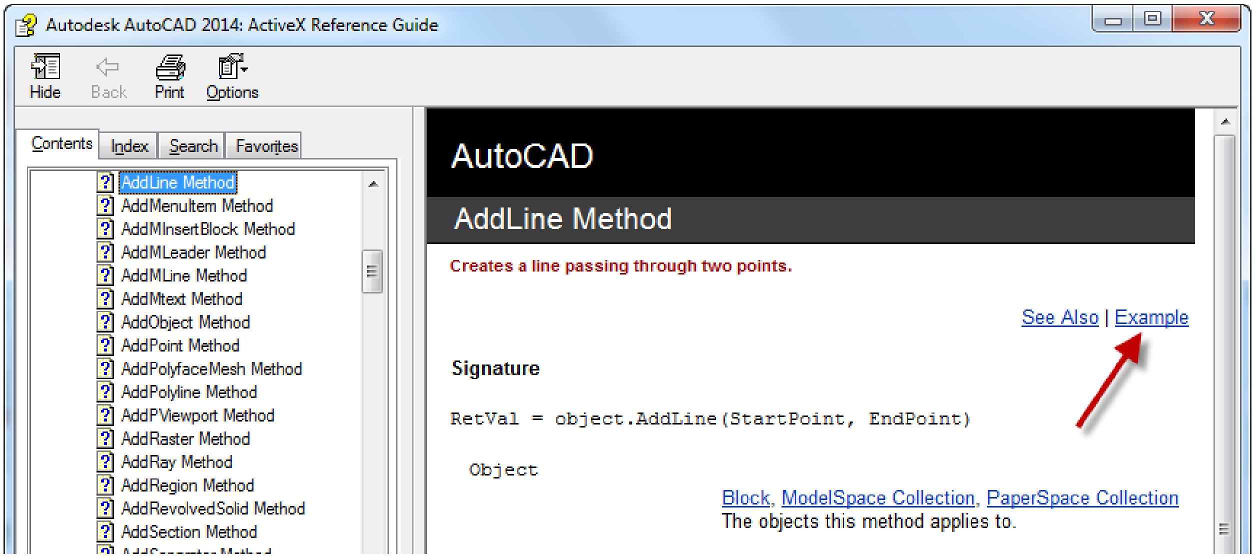


Library	Class	Member
AutoCAD	AcadBlock	AddLine
AutoCAD	AcadModelSpace	AddLine
AutoCAD	AcadPaperSpace	AddLine

Function **AddLine(***StartPoint*, *EndPoint***)** As AcadLine
Member of AutoCAD.AcadModelSpace
Creates a line passing through two points

13. In the Search Results section, select one of the listed results and press F1.

14. On the AddLine Method topic, click the Example link near the upper-right corner.



AutoCAD

AddLine Method

Creates a line passing through two points.

See Also | Example

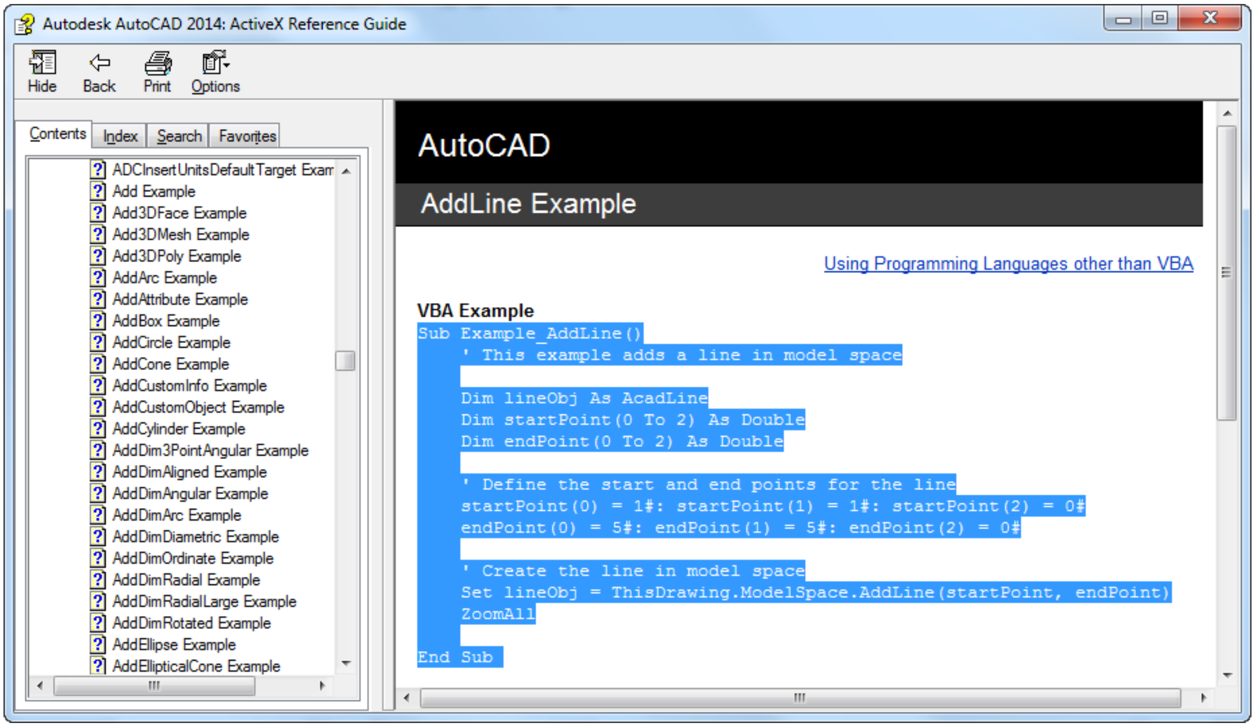
Signature

```
RetVal = object.AddLine(StartPoint, EndPoint)
```

Object

Block, ModelSpace Collection, PaperSpace Collection
The objects this method applies to.

15. Highlight the example code under the section titled VBA Example.



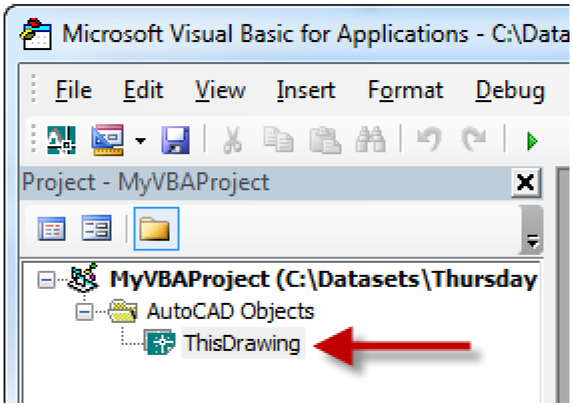
VBA Example

```
Sub Example_AddLine()
    ' This example adds a line in model space

    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double

    ' Define the start and end points for the line
    startPoint(0) = 1#: startPoint(1) = 1#: startPoint(2) = 0#
    endPoint(0) = 5#: endPoint(1) = 5#: endPoint(2) = 0#

    ' Create the line in model space
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    ZoomAll

End Sub
```

16. Right-click over the highlighted code and click Copy.

17. Switch back to the VBA Editor using the Windows taskbar.

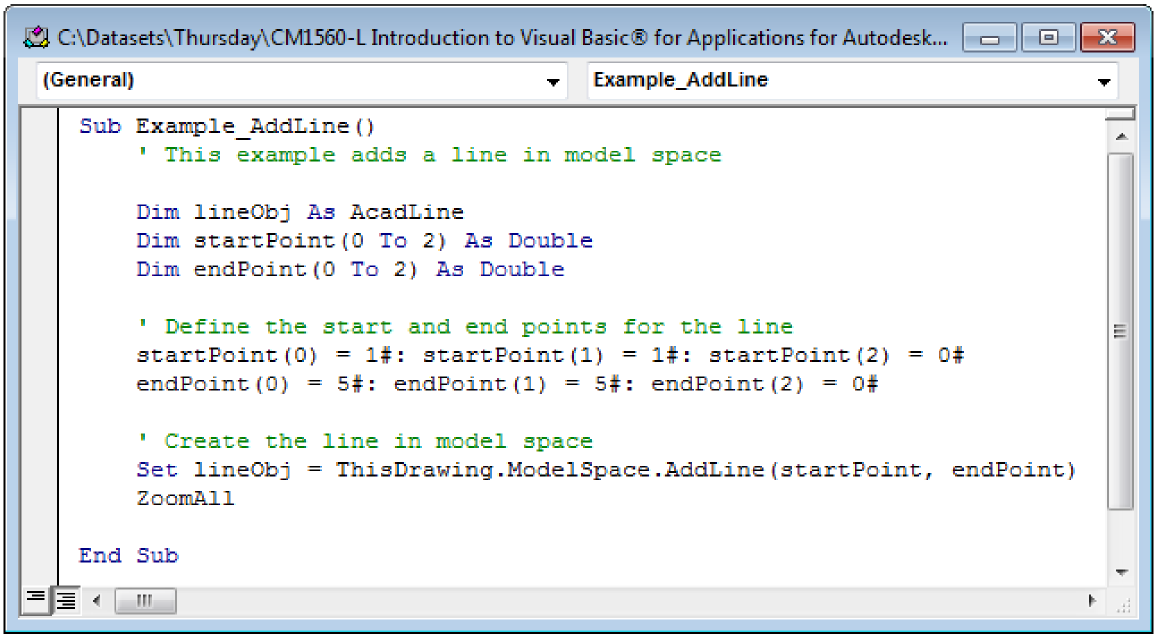
18. In the Project Explorer, on the left side, double-click the ThisDrawing class module to open it in a code editor window.


Microsoft Visual Basic for Applications - C:\Data
File
Edit
View
Insert
Format
Debug
Project - MyVBAProject
MyVBAProject (C:\Datasets\Thursday
AutoCAD Objects
ThisDrawing

19. Click anywhere in the code editor window that appears.

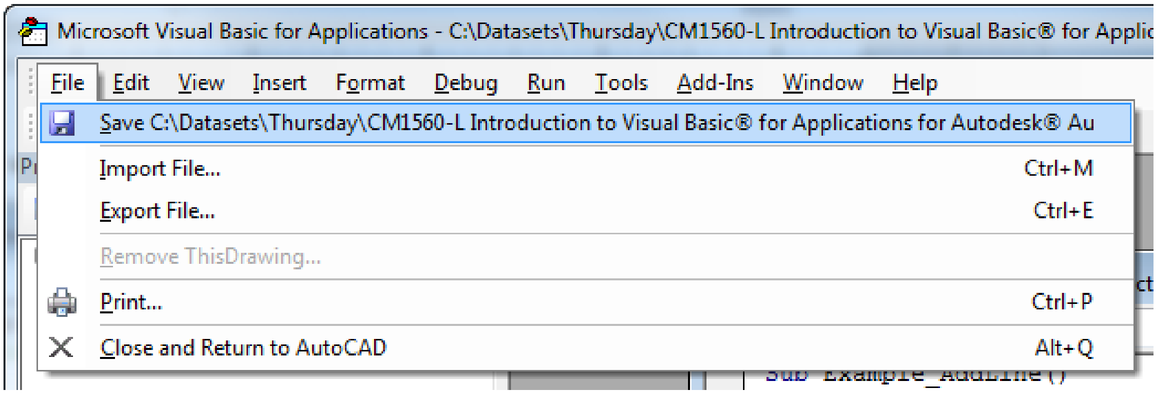
20. In the code editor window, right-click and click Paste.

    The code editor window should now look like the following image.

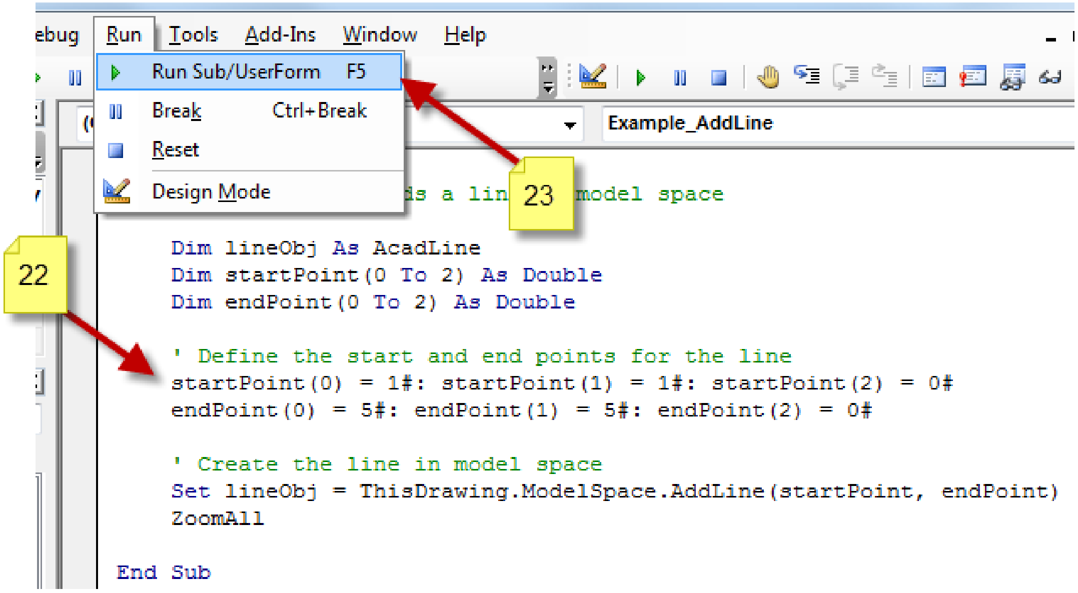

C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk...
(General)
Example_AddLine

```
Sub Example_AddLine()
    ' This example adds a line in model space

    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double

    ' Define the start and end points for the line
    startPoint(0) = 1#: startPoint(1) = 1#: startPoint(2) = 0#
    endPoint(0) = 5#: endPoint(1) = 5#: endPoint(2) = 0#

    ' Create the line in model space
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    ZoomAll

End Sub
```

21. On the menu bar, click File ➢ Save.


Microsoft Visual Basic for Applications - C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applic
File
Edit
View
Insert
Format
Debug
Run
Tools
Add-Ins
Window
Help
Save C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk® Au
Import File... Ctrl+M
Export File... Ctrl+E
Remove ThisDrawing...
Print... Ctrl+P
Close and Return to AutoCAD Alt+Q

22. In the code editor window, click between the statements `Sub Example_AddLine()` and `End Sub`.



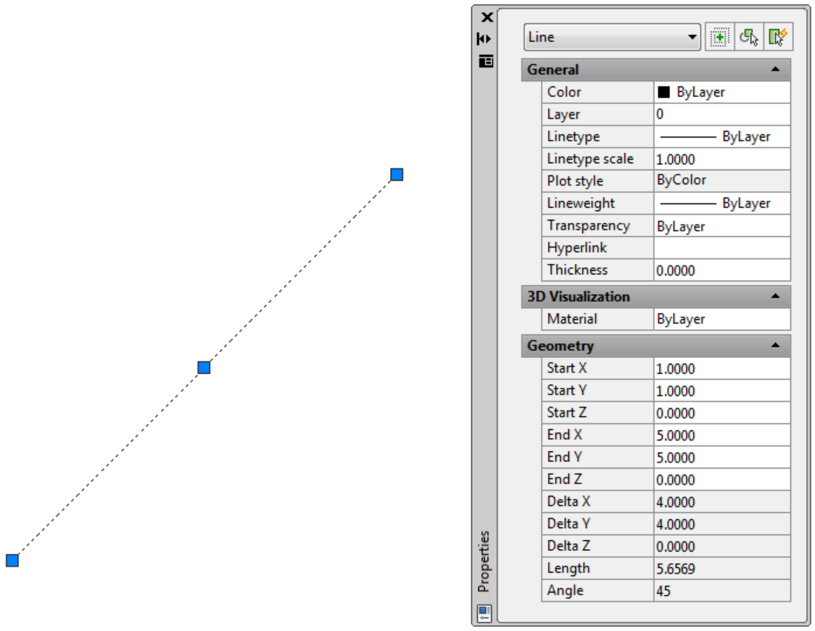
```
ds a lin     model space

    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double

    ' Define the start and end points for the line
    startPoint(0) = 1#: startPoint(1) = 1#: startPoint(2) = 0#
    endPoint(0) = 5#: endPoint(1) = 5#: endPoint(2) = 0#

    ' Create the line in model space
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    ZoomAll

End Sub
```

23. On the menu bar, click Run ➢ Run Sub/UserForm.

24. Switch back to the AutoCAD application window.

25. Select the line in the drawing area. Right-click and click Properties.

    On the Properties palette, you should notice that the new line is drawn from 1,1,0 to 5,5,0.

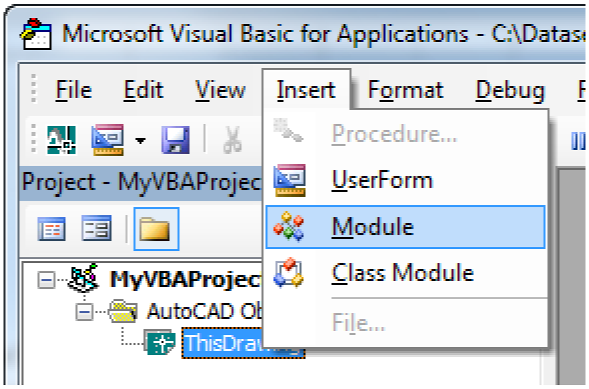


General	
Color	■ ByLayer
Layer	0
Linetype	——— ByLayer
Linetype scale	1.0000
Plot style	ByColor
Lineweight	——— ByLayer
Transparency	ByLayer
Hyperlink	
Thickness	0.0000
3D Visualization	
Material	ByLayer
Geometry	
Start X	1.0000
Start Y	1.0000
Start Z	0.0000
End X	5.0000
End Y	5.0000
End Z	0.0000
Delta X	4.0000
Delta Y	4.0000
Delta Z	0.0000
Length	5.6569
Angle	45

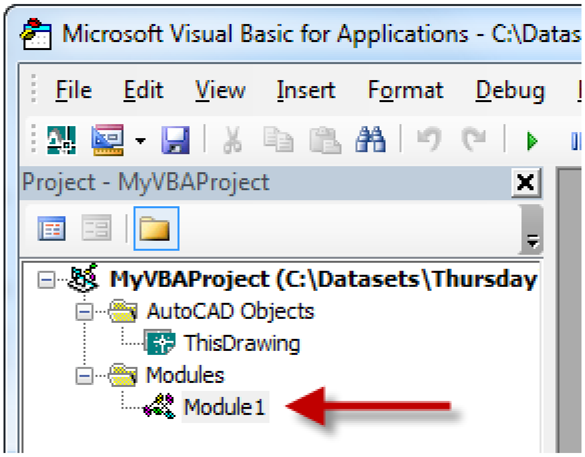
E3 Create New Procedures

This exercise demonstrates how to add a new Public procedure named `DrawCircle` and a Public procedure named `MakeLayer`.

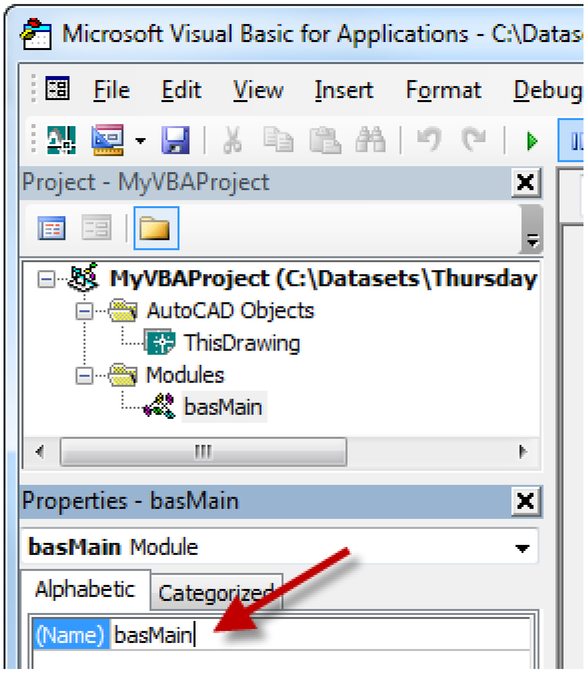
1. In the VBA IDE, on the menu bar, click Insert ➢ Module.



2. In the Project Explorer, double-click the Module1 code module to make sure the code editor window is opened and has focus.



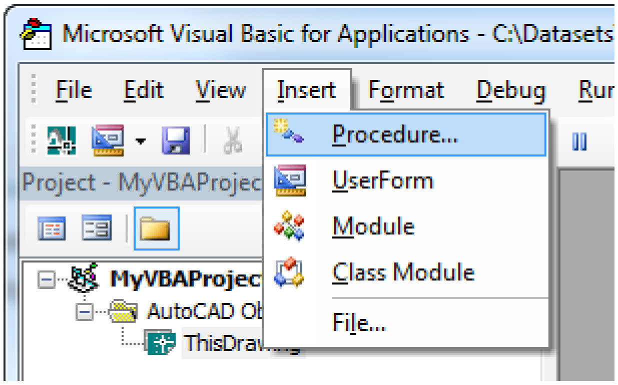
3. In the Properties window, click in the (Name) field and clear the current value. Type **basMain** and press Enter.


Microsoft Visual Basic for Applications - C:\Datas
File Edit View Insert Format Debug
Project - MyVBAProject
MyVBAProject (C:\Datasets\Thursday
AutoCAD Objects
ThisDrawing
Modules
basMain
Properties - basMain
basMain Module
Alphabetic
Categorized

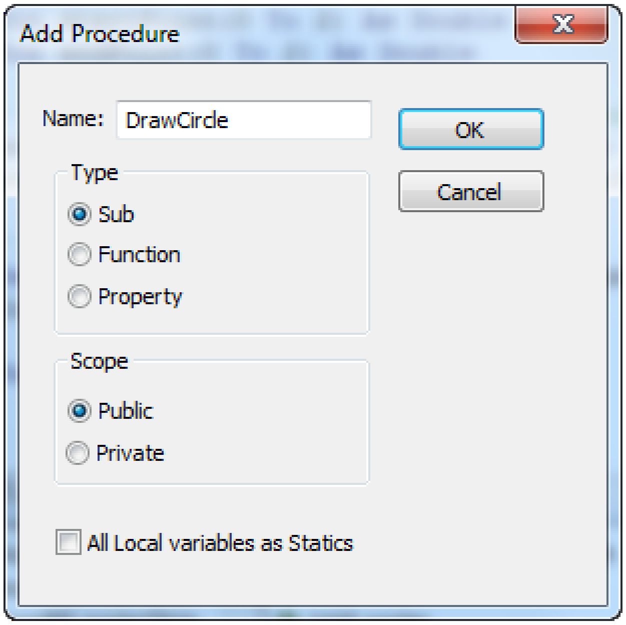
(Name) basMain

4. In the code editor window, click anywhere so it has focus.

5. On the menu bar, click Insert ➢ Procedure.



6. In the Add Procedure dialog box, Name text box, type **DrawCircle**.



Name: DrawCircle

Type

● Sub
○ Function
○ Property

Scope

● Public
○ Private

□ All Local variables as Statics

7. In the Type section, click Sub.
8. In the Scope section, click Public.
9. Click OK to create the procedure.

   The following code should now be in the code editor window:

```
Public Sub DrawCircle()

End Sub
```

10. On the menu bar, click Insert ➢ Procedure.
11. In the Add Procedure dialog box, Name text box, type **MakeLayer**.
12. Set the type of the procedure to Function and its scope to Public. Click OK.

    The following code should now be in the code editor window:

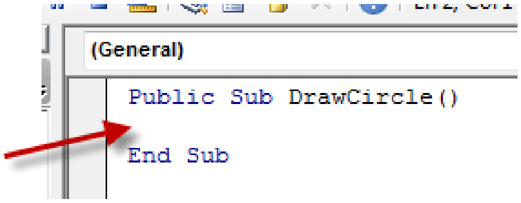
```
Public Function MakeLayer()

End Function
```

13. On the menu bar, click File ➢ Save.

E4 Adding a New Circle

This exercise explains how to add statements to the `DrawCircle` procedure that add a new circle to model space.

1. In the VBA IDE, Project Explorer, double-click basMain to open it in the code editor window.
2. In the code editor window, click between the two statements that define the `DrawCircle` procedure.



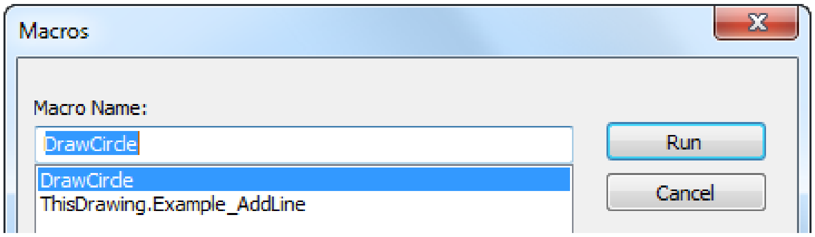
```
Public Sub DrawCircle()

End Sub
```

3. Type the following:

```
Dim centerPt(0 To 2) As Double
Dim radius As Double

centerPt(0) = 2: centerPt(1) = 3: centerPt(2) = 0
radius = 3

ThisDrawing.ModelSpace.AddCircle centerPt, radius
```

4. On the menu bar, click File ➢ Save.
5. Click in between the two statements that define the `DrawCircle` procedure.
6. On the menu bar, click Run ➢ Run Sub/UserForm.

   It might appear that nothing happened, but that is okay. If the Macros dialog box is displayed, the cursor was not inside the `DrawCircle` procedure. Select the `DrawCircle` macro and click Run.



Macro Name: DrawCircle

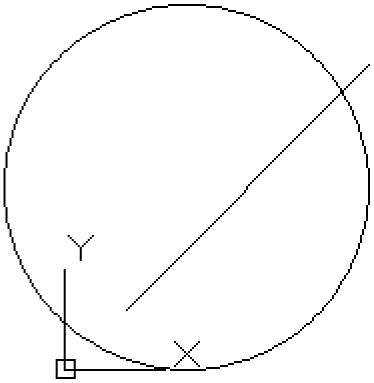
DrawCircle
ThisDrawing.Example_AddLine

7. Switch to the AutoCAD application window using the Windows task bar.

   You can also click View ➢ AutoCAD on the VBA IDE menu bar.

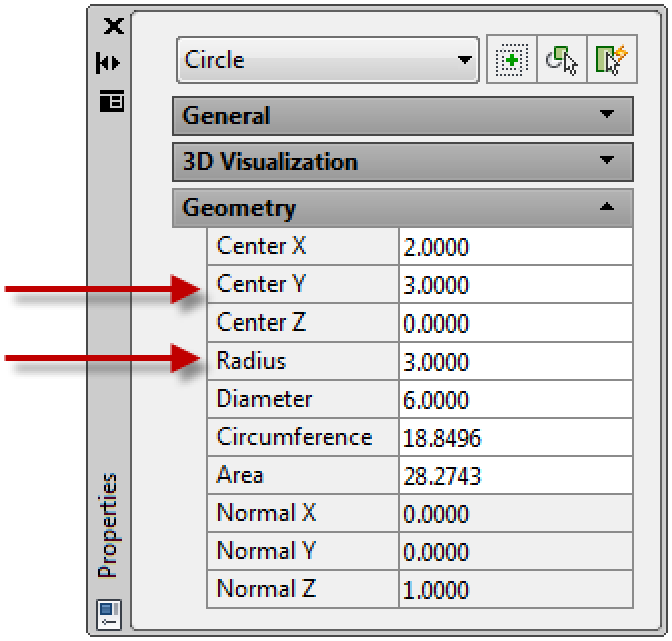
8. Zoom to the extents of the drawing.

   You should now see the new circle, and the line that was created when the Example_AddLine was executed.



9. Select the circle in the drawing area. Right-click and click Properties.
10. On the Properties palette, review the properties of the new circle.

    You should see that the circle has a radius of 3 and a center point of 2,3,0.



Center X	2.0000
Center Y	3.0000
Center Z	0.0000
Radius	3.0000
Diameter	6.0000
Circumference	18.8496
Area	28.2743
Normal X	0.0000
Normal Y	0.0000
Normal Z	1.0000

E5 Using the MakeLayer Procedure with the DrawCircle Procedure

This exercise explains how to modify the `DrawCircle` procedure so the new circle is placed on a layer created by the `MakeLayer` procedure. You will also use the `GetPoint` and `GetDistance` methods to request input at the Command prompt to specify the radius and centerpoint of the circle.

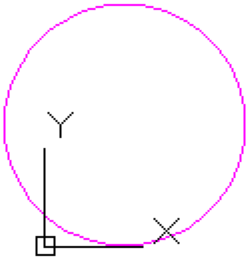
1. Right-click the Windows Start button, and click Open Windows Explorer or File Explorer based on the Windows version installed.
2. In Windows Explorer or File Explorer, browse to *C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®* and double-click the *EX5 - Procedures.txt* file.
3. In Notepad, highlight all the code and right-click. Click Copy.
4. Switch back to the VBA Editor by selecting it from the Windows taskbar.
5. In the VBA IDE, Project Explorer, double-click basMain to open it in the code editor window.
6. In the code editor window, select all of the code in the window and right-click. Click Paste.

   The procedures in the code editor window should now look like:

```
Public Sub DrawCircle()
  Dim newCircle As AcadCircle
  Dim centerPt As Variant
  Dim radius As Double

  AppActivate ThisDrawing.Application.Caption

  Dim objUtil As AcadUtility
  Set objUtil = ThisDrawing.Utility

  centerPt = objUtil.GetPoint(, vbLf + "Enter center point: ")

  radius = objUtil.GetDistance(centerPt, vbLf + "Enter radius: ")

  Set newCircle = ThisDrawing.ModelSpace. _
                       AddCircle(centerPt, radius)

  Dim objLayer As AcadLayer
  Set objLayer = MakeLayer("Objects-Circs", acMagenta)
  newCircle.Layer = objLayer.Name
  objUtil.Prompt vbLf + "Circle added to " + newCircle.Layer
End Sub

Public Function MakeLayer(Name As String, _
                            ACI As Integer) As AcadLayer
  Dim newLayer As AcadLayer
  Dim color As New AcadAcCmColor

  color.ColorIndex = ACI

  Set newLayer = ThisDrawing.Layers.Add(Name)
  newLayer.TrueColor = color

  Set color = Nothing
  Set MakeLayer = newLayer
End Function
```

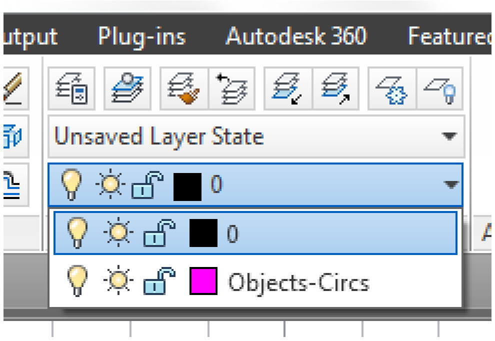
7. On the menu bar, click File ➢ Save.
8. Click in between the two statements that define the `DrawCircle` procedure.
9. On the menu bar, click Run ➢ Run Sub/UserForm.
10. In AutoCAD, at the *`Enter center point:`* prompt, specify a point in the drawing area. You can type a value or use the mouse to pick a point.
11. At the *`Enter radius:`* prompt, specify a point in the drawing area or enter a value.

    The circle object is now drawn based on the center point and radius values you specified, and is placed on the Objects-Circs layer.



12. On the ribbon, click Home tab ➢ Layers panel ➢ Layer drop-down list.

You see the Objects-Circs layer has been added to the drawing.



E6 Selecting Objects

This exercise demonstrates how to select objects in a drawing and step through each of the selected objects.

1. In the VBA IDE, Project Explorer, double-click basMain to open it in the code editor.
2. On the menu bar, click Insert ➢ Procedure.
3. In the Add Procedure dialog box, Name text box, type **SelectAndModifyObjects**.
4. Set the type of the procedure to Sub and its scope to Public. Click OK.
5. In Windows Explorer or File Explorer, browse to *C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®* and double-click the *EX6 - SelectAndModifyObjects.txt* file.
6. In Notepad, highlight all the code and right-click. Click Copy.
7. Switch back to the VBA Editor by selecting it from the Windows taskbar.
8. Click in between the two statements that define the `SelectAndModifyObjects` procedure and right-click. Click Paste.

   The SelectAndModifyObjects procedure should now look like:

```
Public Sub SelectAndModifyObjects()
  Dim sset As AcadSelectionSet

  Dim ssets As AcadSelectionSets
  Set ssets = ThisDrawing.SelectionSets

  On Error Resume Next

  Set sset = ssets.Item("AU2013_SS")
  If Err.Number <> 0 Then
    Set sset = ssets.Add("AU2013_SS")
  Else
    sset.Clear
  End If

  AppActivate ThisDrawing.Application.Caption

  sset.SelectOnScreen

  Dim ent As AcadEntity
  Dim line As AcadLine
  Dim circ As AcadCircle

  For Each ent In sset
    If ent.ObjectName = "AcDbLine" Then
       Set line = ent

       line.EndPoint = _
         ThisDrawing.Utility.PolarPoint(line.EndPoint, _
                                        line.Angle, _
                                        line.Length * 0.25)
    ElseIf ent.ObjectName = "AcDbCircle" Then
       Set circ = ent

       circ.radius = circ.radius * 1.25
    End If

    ent.Update
  Next ent
End Sub
```

9. On the menu bar, click File ➢ Save.

10. Click in between the two statements that define the `SelectAndModifyObjects` procedure.

11. On the menu bar, click Run ➢ Run Sub/UserForm.

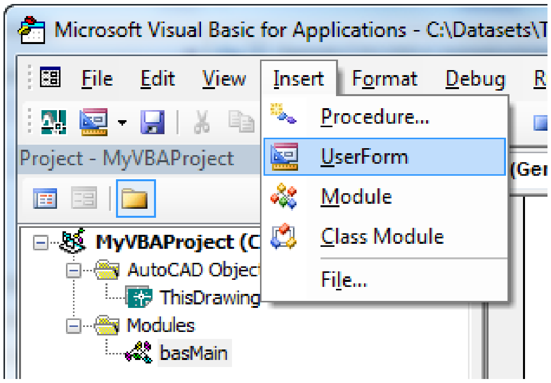
12. At the *`Select objects:`* prompt, select some objects in the drawing area.

    If a line (AcDbLine) is selected, its length is changed by a factor of 25%; while if a circle (AcDbCircle) is selected, its radius is increased by 25%.

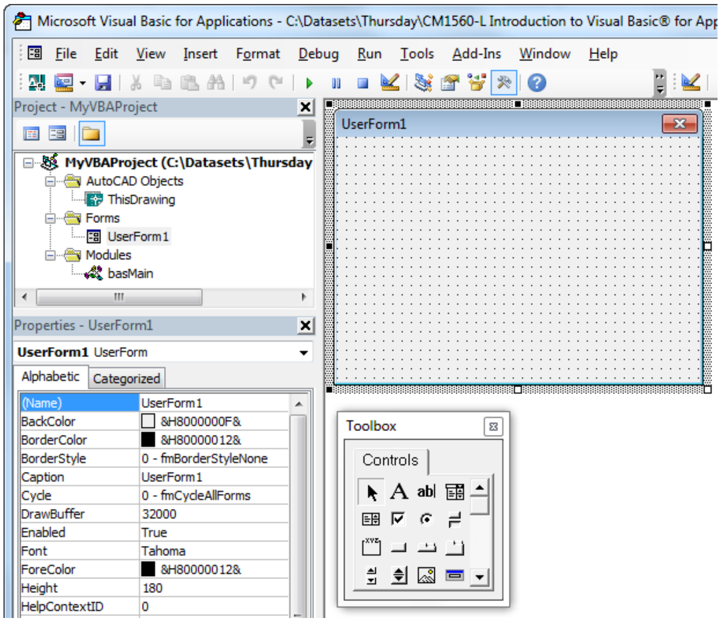
E7 Create a UserForm

This exercise explains how to create a UserForm that you can use to start a procedure and get input from a user.

1. In the VBA IDE, on the menu bar, click Insert ➢ UserForm.



2. In the Project Explorer, double-click UserForm1 to make sure the form editor window is opened and has focus.

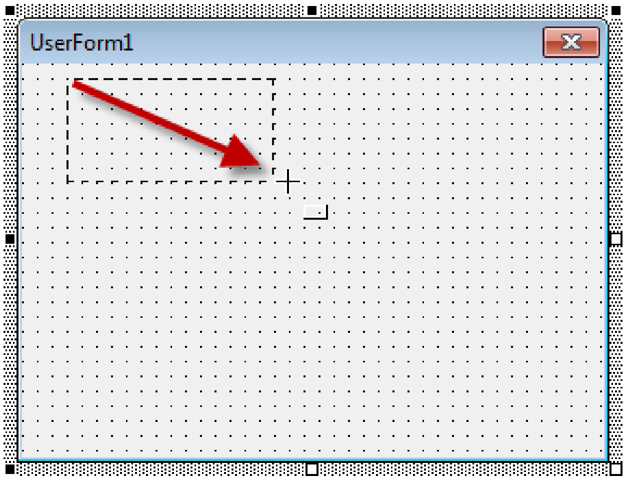


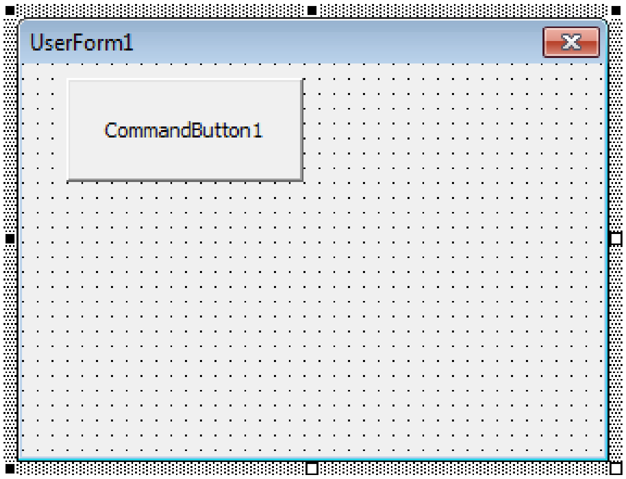
(Name)	UserForm1
BackColor	□ &H8000000F&
BorderColor	■ &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	UserForm1
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	180
HelpContextID	0

3. On the Toolbox window, click the CommandButton control.

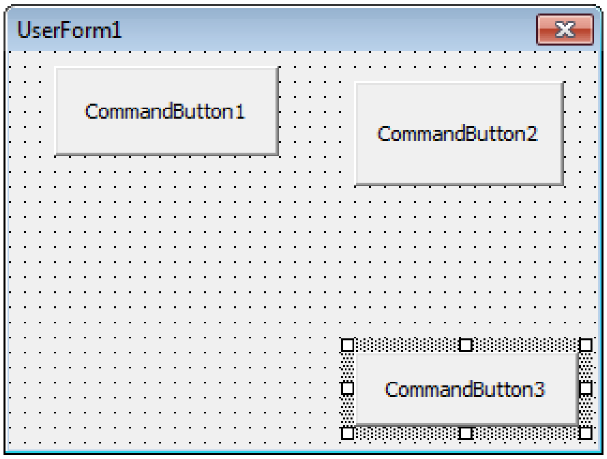
   If the Toolbox window is not displayed, on the menu bar, click View ➢ Toolbox.

4. Click and drag in the form editor window to create the control. Release the mouse button when finished.

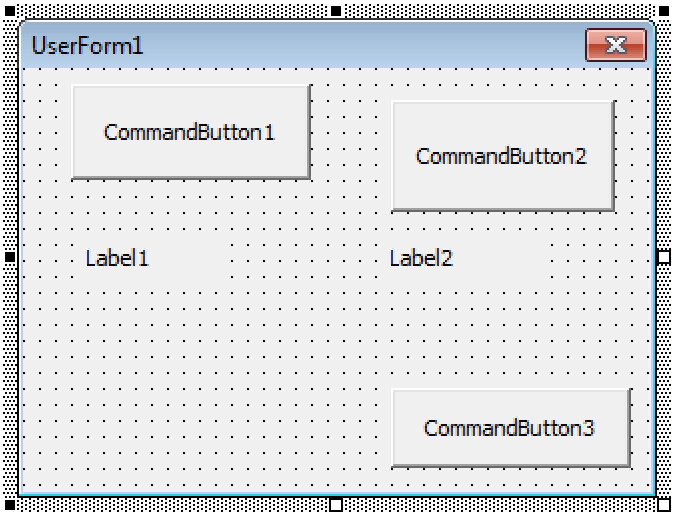




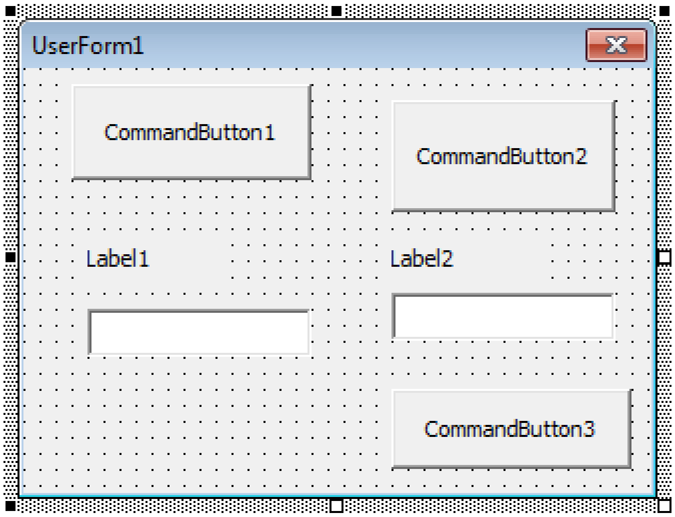
5. Add two more CommandButtons to the UserForm.



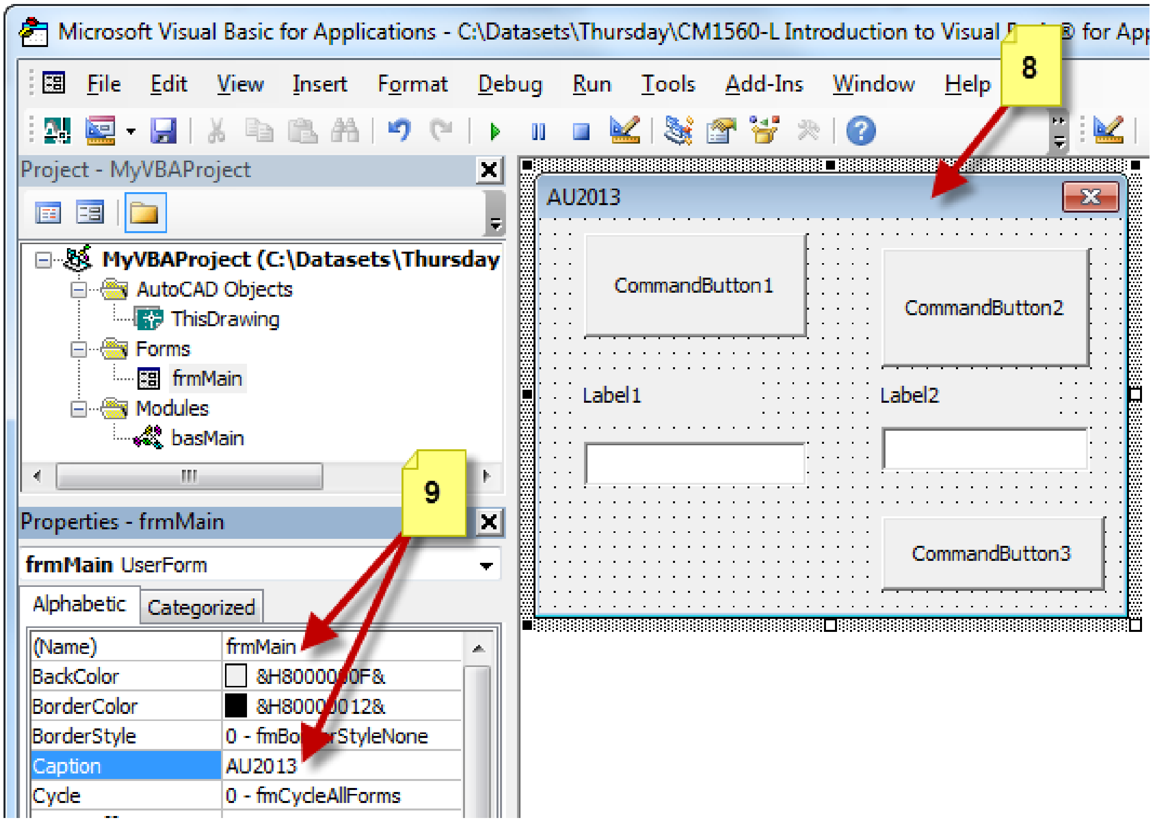
6. On the Toolbox window, click the Label control A and add two labels; one below each of the two CommandButton controls that you added along the top of the UserForm.



7. On the Toolbox window, click the TextBox control ab| and add two text boxes; one below each of the two Label controls that you added to the UserForm.



8. Click the title bar of the UserForm.


Microsoft Visual Basic for Applications - C:\Datasets\Thursday\CM1560-L Introduction to Visual B... for App
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
8
Project - MyVBAProject
MyVBAProject (C:\Datasets\Thursday
AutoCAD Objects
ThisDrawing
Forms
frmMain
Modules
basMain
9
Properties - frmMain
frmMain UserForm
Alphabetic
Categorized
AU2013
CommandButton1
CommandButton2
Label1
Label2
CommandButton3

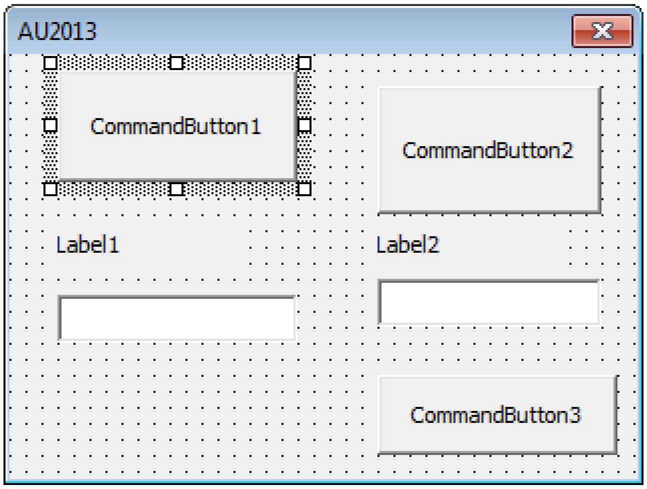
(Name)	frmMain
BackColor	&H8000000F&
BorderColor	&H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	AU2013
Cycle	0 - fmCycleAllForms

9. In the Properties Window, click in the
   a. (Name) field and change the value to **frmMain**
   b. Caption field and change the value to **AU2013**

10. Click in the form editor window.

11. On the Toolbox window, click the Select Objects tool.

12. In the form editor window, click the first CommandButton you added; labeled CommandButton1 by default.



13. In the Properties Window, click in the
    a. (Name) field and change the value to **cmdDrawCircle**
    b. Caption field and change the value to **Draw Circle**

14. Change the properties of the other controls you added to:

<u>CommandButton2</u>
(Name): cmdSelectObjects
Caption: Modify Objects

<u>Label1</u>
(Name): lblLayerName
Caption: Name:

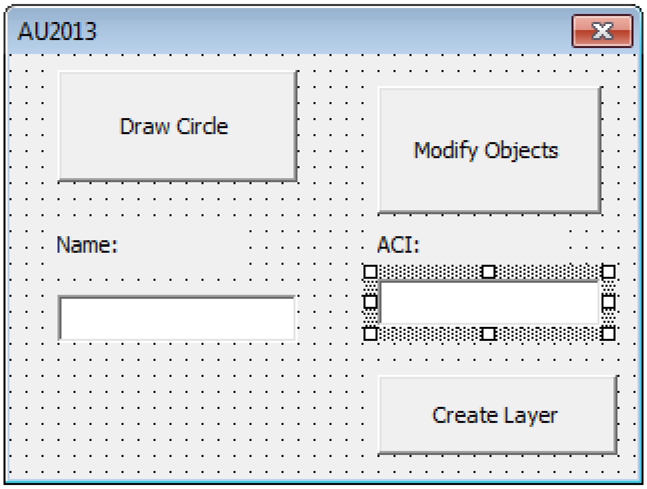
<u>TextBox1</u>
(Name): txtLayerName

<u>CommandButton3</u>
(Name): cmdCreateLayer
Caption: Create Layer

<u>TextBox2</u>
(Name): lblLayerACI
Caption: ACI:

<u>TextBox2</u>
(Name): txtLayerACI

The UserForm should now look like the following:



15. In the Project Explorer, double-click basMain to open it in the code editor window.
16. On the menu bar, click Insert ➢ Procedure.
17. In the Add Procedure dialog box, Name text box, type **ShowMain**.
18. Set the type of the procedure to Sub and its scope to Public. Click OK.
19. Add the following statements in bold to the new `ShowMain` procedure:

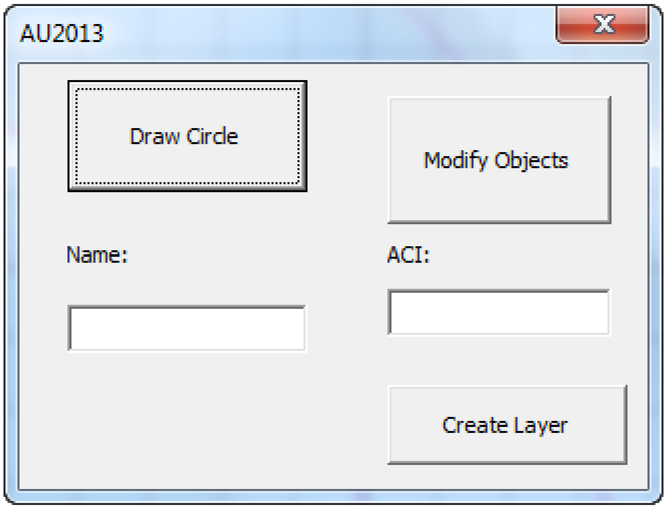
```
Public Sub ShowMain()
  Dim form As New frmMain
  form.show

  Set form = Nothing
End Sub
```

20. On the menu bar, click File ➢ Save.
21. Click in between the two statements that define the `ShowMain` procedure.
22. On the menu bar, click Run ➢ Run Sub/UserForm.

The UserForm is displayed in the AutoCAD application window.

23. On the UserForm, interact with the controls you added and then click the Close button when you are done.

    None of the buttons work with the exception of the Close button.

E8 Adding Events to the Controls on the UserForm

This exercise explains how to add procedures that are executed when the user interacts with the controls on the UserForm.

1. In the VBA IDE, Project Explorer, double-click frmMain to make sure the form editor window is opened and has focus.

2. Double-click the Draw Circle command button.

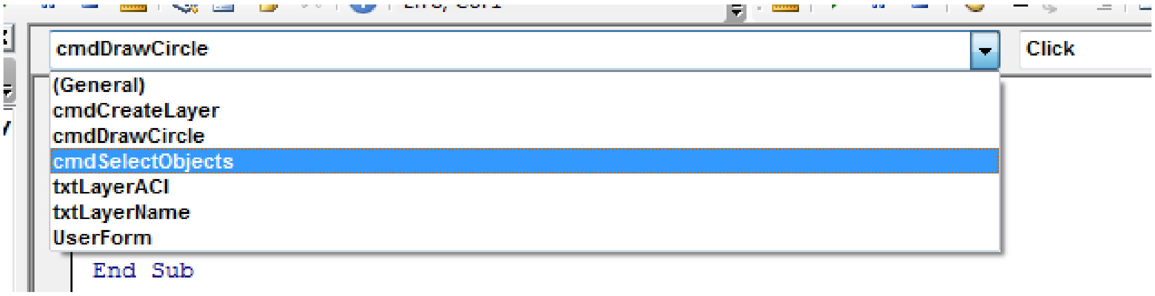
   The code editor window for the UserForm opens and the following procedure is added.

```
Private Sub cmdDrawCircle_Click()

End Sub
```

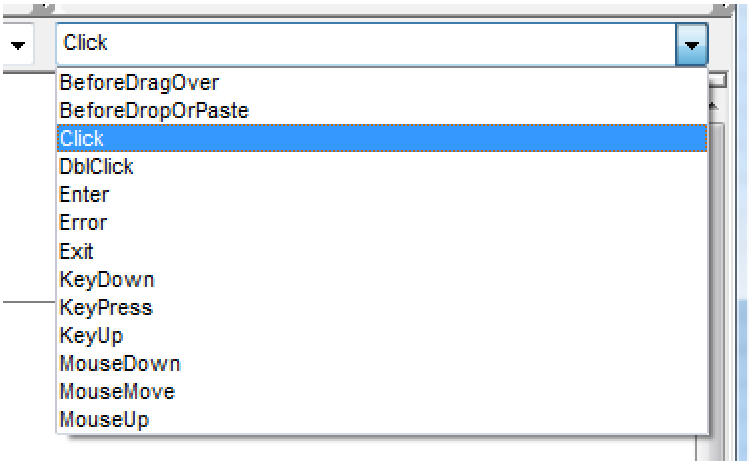
3. Add the following statements to the new `cmdDrawCircle_Click` procedure:

```
Private Sub cmdDrawCircle_Click()
  Me.Hide
  DrawCircle
  Me.show
End Sub
```

4. In the code editor window, along the top, click the Objects drop-down list and select cmdSelectObjects.



5. In the Procedure drop-down list, select Click if it is not by default.



The Click event for the cmdSelectObjects CommandButton is added to the code editor window and is just like double-clicking on the control in the form editor window.

```
Private Sub cmdSelectObjects_Click()

End Sub
```

6. Add the following statements to the new `cmdSelectObjects_Click` procedure:

```
Private Sub cmdSelectObjects_Click()
  Me.Hide
  SelectAndModifyObjects
  Me.show
End Sub
```

7. In the Project Explorer, double-click frmMain to make sure the form editor window is opened and has focus.

8. Double-click the Create Layer command button.

9. Add the following statement to the new `cmdCreateLayer_Click` procedure:

```
Private Sub cmdCreateLayer_Click()
  MakeLayer txtLayerName.Text, CInt(txtLayerACI.Text)
End Sub
```

10. On the menu bar, click File ➢ Save.

11. In the Project Explorer, double-click basMain to open it in the code editor window.

12. Click between the two statements that define the `ShowMain` procedure.

13. On the menu bar, click Run ➢ Run Sub/UserForm.

14. On the AU2013 dialog box, click Draw Circle and follow the prompts that are displayed.

15. Click Modify Objects and select some of the circles and lines in your drawing.

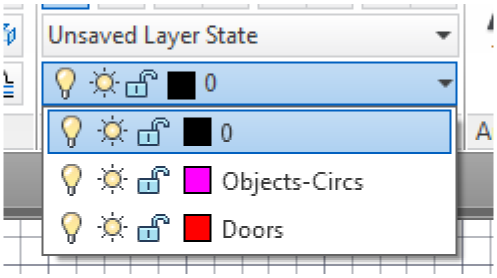
16. In the first text box, below the Name: label, type **Doors**.

17. In the second text box, below the ACI: label, type **1**.

18. Click Create Layer and then click Close.

19. Switch to the AutoCAD application window. On the ribbon, click Home tab ➢ Layers panel ➢ Layer drop-down list.

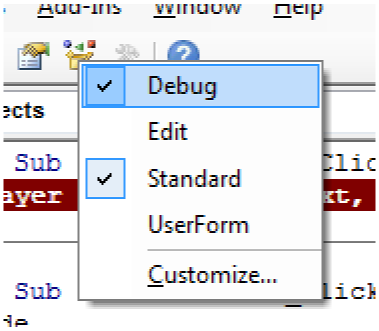
    You see the Doors layer has been added to the drawing.



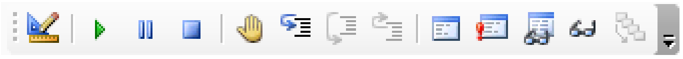
E9 Stepping through Code and Adding Watches

This exercise explains how to use Breakpoints and Watches while a procedure is executing.

1. In the VBA IDE, right-click over a toolbar and click Debug if it is not checked.

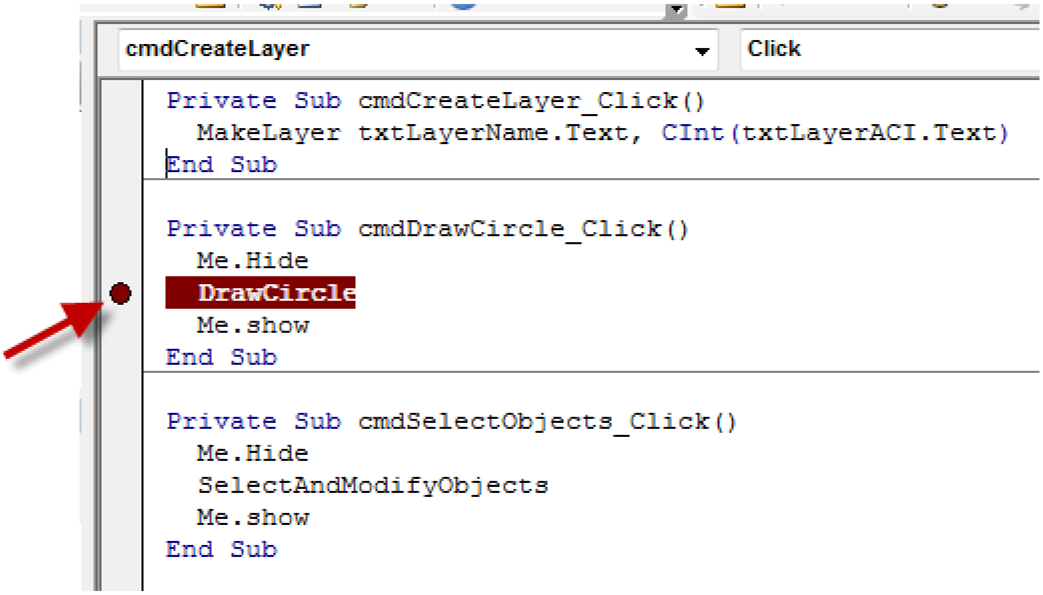


The following image shows the Debug toolbar.



2. In the Project Explorer, right-click frmMain and click View Code.

3. In the code editor window, click in the Indicator Margin bar as shown in the following image to add a breakpoint.



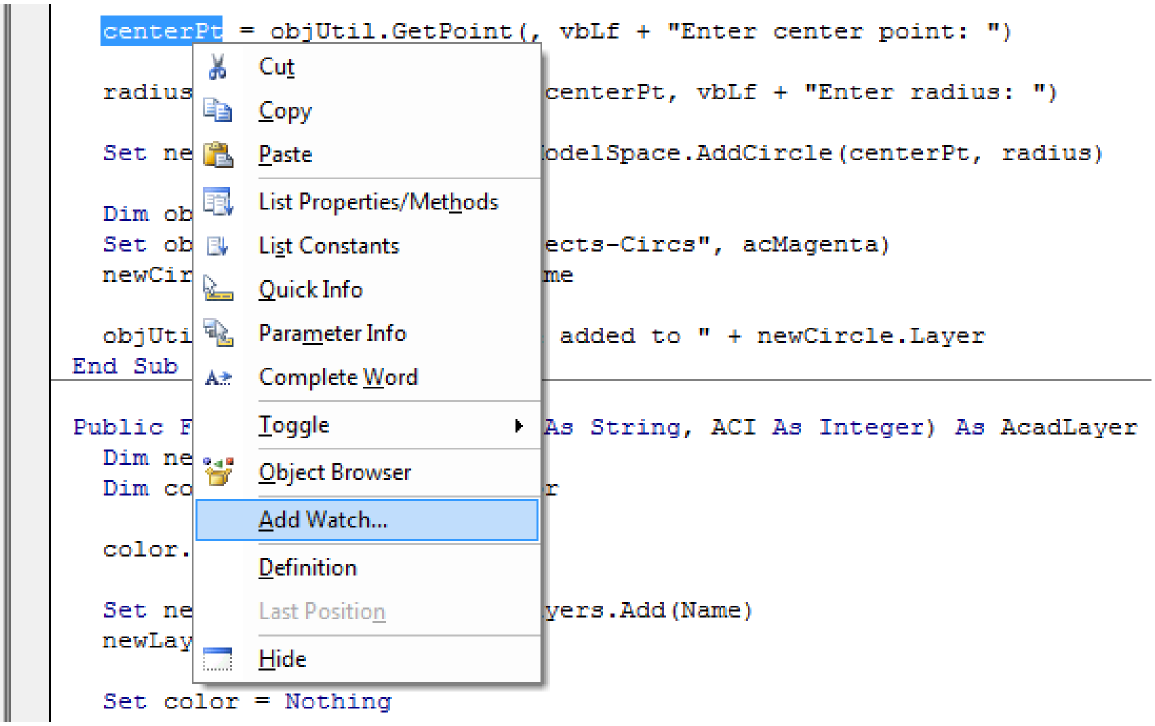
cmdCreateLayer
Click

```
Private Sub cmdCreateLayer_Click()
  MakeLayer txtLayerName.Text, CInt(txtLayerACI.Text)
End Sub

Private Sub cmdDrawCircle_Click()
  Me.Hide
  DrawCircle
  Me.show
End Sub

Private Sub cmdSelectObjects_Click()
  Me.Hide
  SelectAndModifyObjects
  Me.show
End Sub
```

4. In the Project Explorer, double-click basMain to open it in the code editor window.

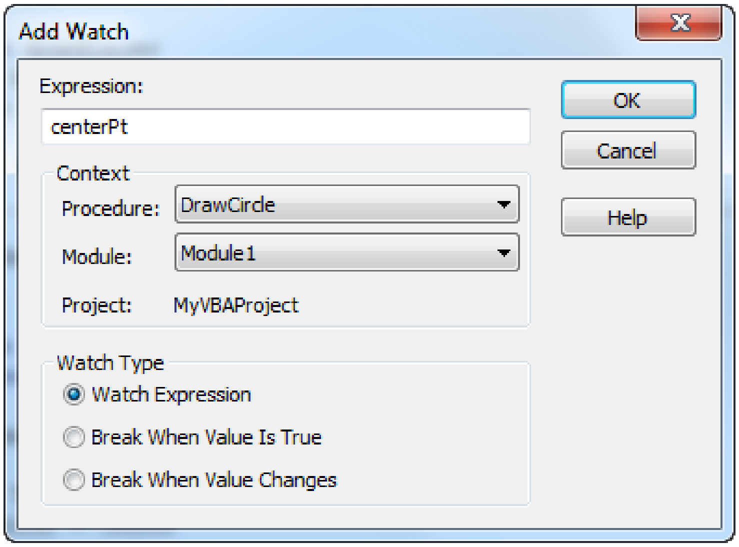
5. Scroll to the `DrawCircle` procedure.

6. Highlight the *centerPt* variable name in the `DrawCircle` procedure. Right-click and click Add Watch.



```
    centerPt = objUtil.GetPoint(, vbLf + "Enter center point: ")

    radius                      centerPt, vbLf + "Enter radius: ")

    Set ne                     odelSpace.AddCircle(centerPt, radius)

    Dim ob
    Set ob                      ects-Circs", acMagenta)
    newCir                      me

    objUti                        added to " + newCircle.Layer
End Sub

Public F                        As String, ACI As Integer) As AcadLayer
    Dim ne
    Dim co                      r

    color.

    Set ne                      yers.Add(Name)
    newLay

    Set color = Nothing
```

7. In the Add Watch dialog box, in the Watch Type section, click Watch Expression. Click OK.



Add Watch

Expression: centerPt

Context

Procedure: DrawCircle

Module: Module1

Project: MyVBAProject

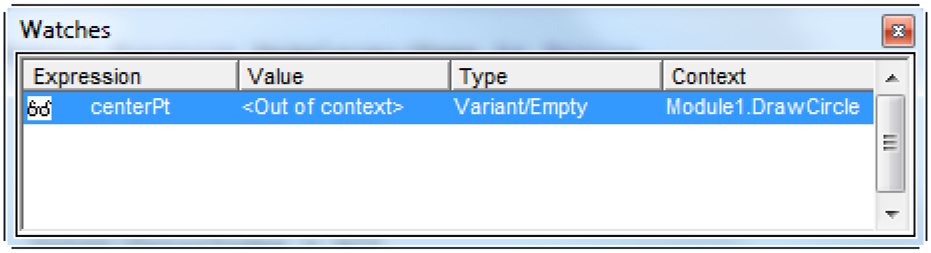
Watch Type

● Watch Expression

○ Break When Value Is True

○ Break When Value Changes

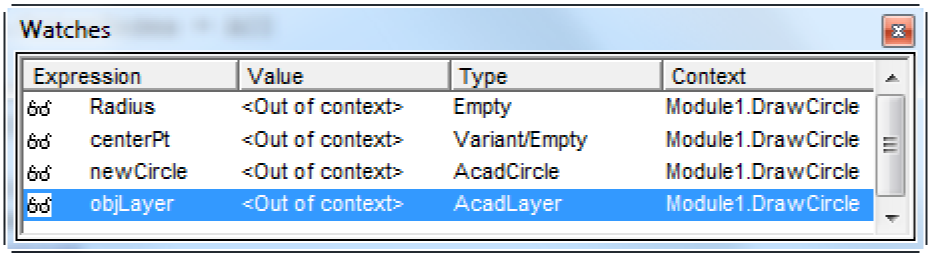
The Watches window should open and show the *centerPt* variable you are now watching.



Watches

Expression	Value	Type	Context
centerPt	<Out of context>	Variant/Empty	Module1.DrawCircle

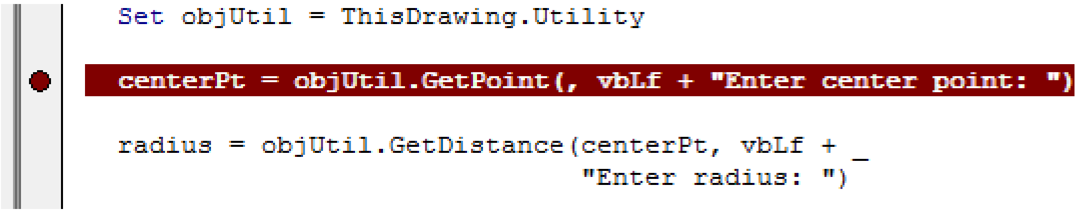
8. Add watches for the *radius*, *newCircle*, and *objLayer* variables in the `DrawCircle` procedure.



Watches

Expression	Value	Type	Context
Radius	<Out of context>	Empty	Module1.DrawCircle
centerPt	<Out of context>	Variant/Empty	Module1.DrawCircle
newCircle	<Out of context>	AcadCircle	Module1.DrawCircle
objLayer	<Out of context>	AcadLayer	Module1.DrawCircle

9. Add a breakpoint to the following statement in the `DrawCircle` procedure.

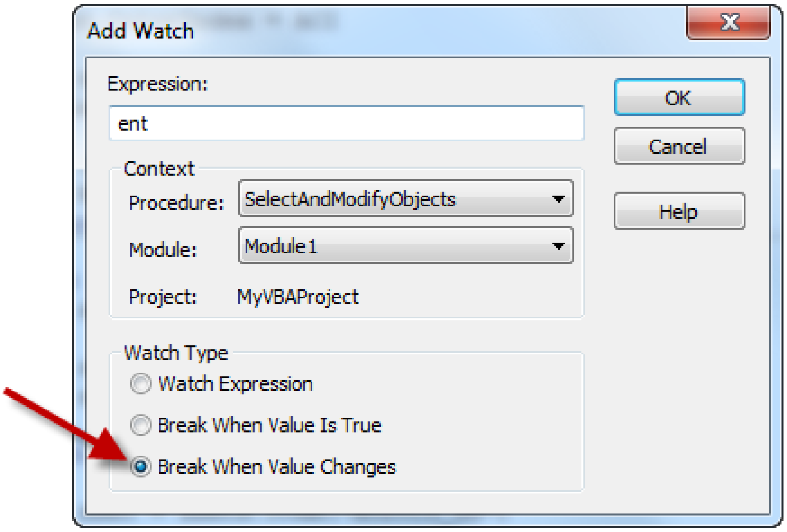


```
Set objUtil = ThisDrawing.Utility

centerPt = objUtil.GetPoint(, vbLf + "Enter center point: ")

radius = objUtil.GetDistance(centerPt, vbLf + _
                             "Enter radius: ")
```

10. Scroll to the `SelectAndModifyObjects` procedure and highlight the variable name *ent*.

11. Right-click and click Add Watch.

12. In the Add Watch dialog box, in the Watch Type section, click Break When Value Changes. Click OK.



Expression: ent

Context
Procedure: SelectAndModifyObjects
Module: Module1
Project: MyVBAProject

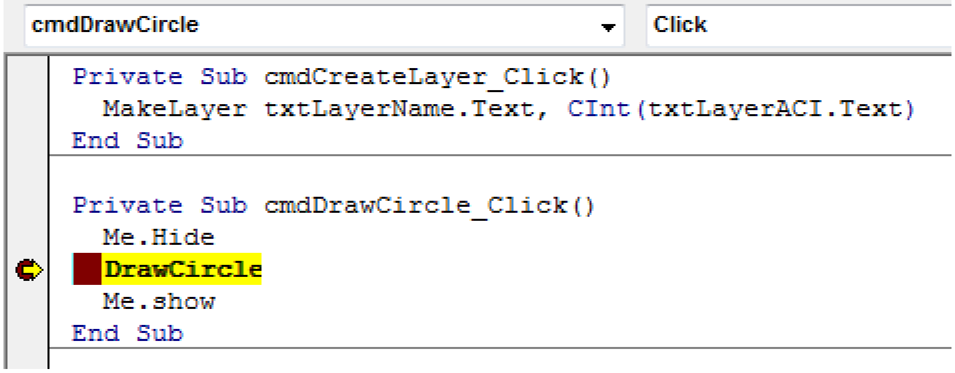
Watch Type
○ Watch Expression
○ Break When Value Is True
● Break When Value Changes

13. Scroll to and click between the two statements that define the `ShowMain` procedure.

14. On the menu bar, click Run ➢ Run Sub/UserForm.

15. In the AU2013 dialog box, click Draw Circle.

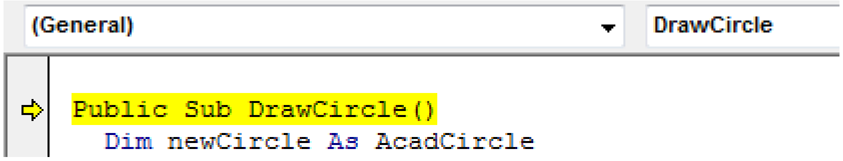
The code editor window opens and the execution of the button's Click event is suspended.



```
Private Sub cmdCreateLayer_Click()
  MakeLayer txtLayerName.Text, CInt(txtLayerACI.Text)
End Sub

Private Sub cmdDrawCircle_Click()
  Me.Hide
  DrawCircle
  Me.show
End Sub
```

16. On the Debug toolbar, click Step Into.

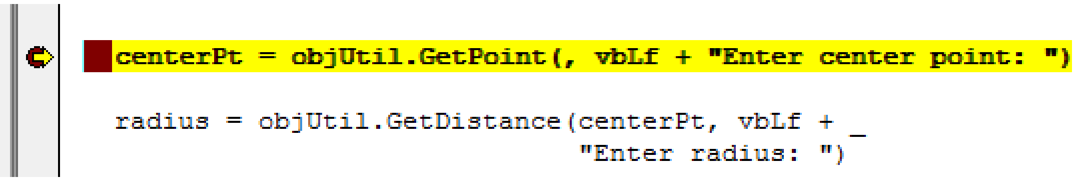
Execution advances to the `DrawCircle` procedure.



```
Public Sub DrawCircle()
  Dim newCircle As AcadCircle
```

17. On the Debug toolbar, click Continue.

    Execution runs as normal until an error or the next breakpoint is encountered.



```
centerPt = objUtil.GetPoint(, vbLf + "Enter center point: ")

radius = objUtil.GetDistance(centerPt, vbLf + _
                             "Enter radius: ")
```

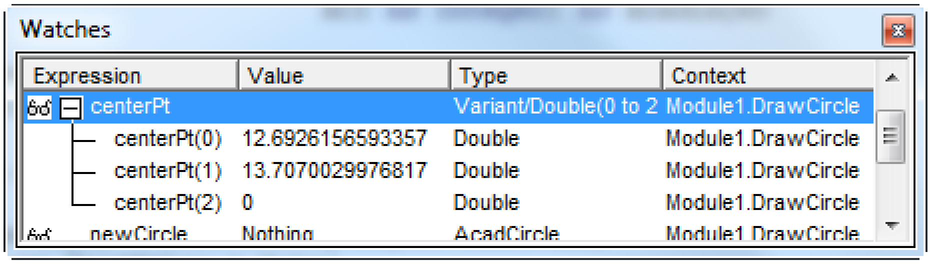
18. On the Debug toolbar, click Step Into.

19. Switch to AutoCAD and specify a center point for the circle.

    Execution returns to the code editor window.

20. In the Watches window, click the plus sign next to the *centerPt* variable.

    The values of the *centerPt* are displayed.

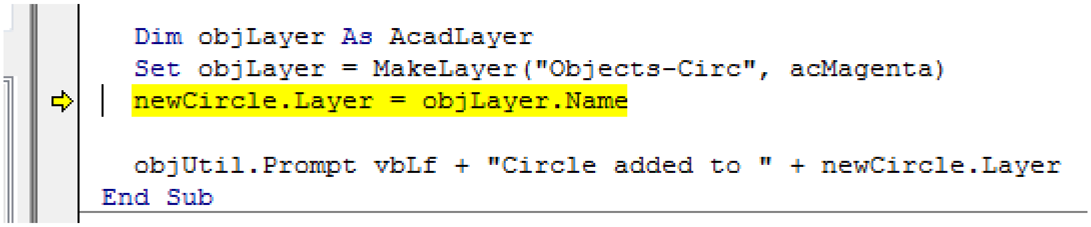


Expression	Value	Type	Context
centerPt		Variant/Double(0 to 2	Module1.DrawCircle
centerPt(0)	12.6926156593357	Double	Module1.DrawCircle
centerPt(1)	13.7070029976817	Double	Module1.DrawCircle
centerPt(2)	0	Double	Module1.DrawCircle
newCircle	Nothing	AcadCircle	Module1.DrawCircle

21. In the code editor window, right-click in front of the `newCircle.Layer = objLayer.Name` statement. Click Run to Cursor.

22. Switch to AutoCAD and specify a radius for the circle.

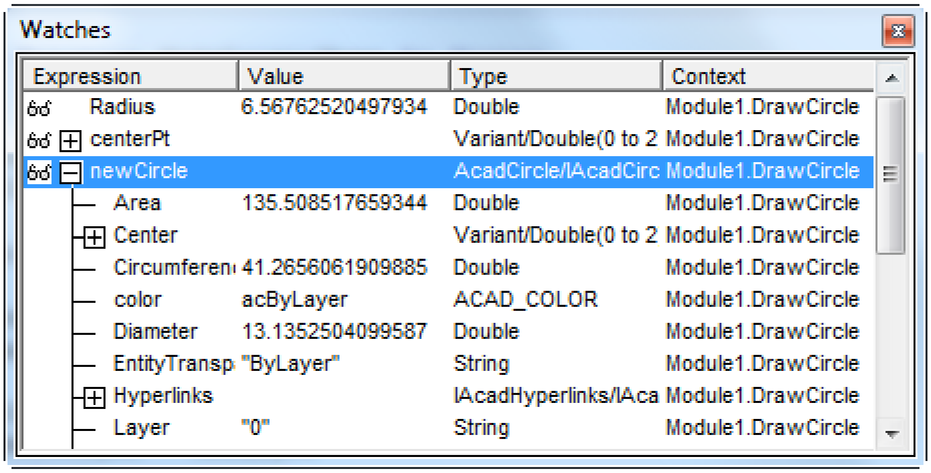
    Execution continues to the `newCircle.Layer = objLayer.Name` statement.



```
  Dim objLayer As AcadLayer
  Set objLayer = MakeLayer("Objects-Circ", acMagenta)
  newCircle.Layer = objLayer.Name

  objUtil.Prompt vbLf + "Circle added to " + newCircle.Layer
End Sub
```

23. In the Watches window, click the plus sign next to the *newCircle* variable.

    The properties of the AcadCircle object assigned to the *newCircle* variable are displayed.



Expression	Value	Type	Context
Radius	6.56762520497934	Double	Module1.DrawCircle
⊞ centerPt		Variant/Double(0 to 2	Module1.DrawCircle
⊟ newCircle		AcadCircle/IAcadCirc	Module1.DrawCircle
Area	135.508517659344	Double	Module1.DrawCircle
⊞ Center		Variant/Double(0 to 2	Module1.DrawCircle
Circumferen	41.2656061909885	Double	Module1.DrawCircle
color	acByLayer	ACAD_COLOR	Module1.DrawCircle
Diameter	13.1352504099587	Double	Module1.DrawCircle
EntityTransp	"ByLayer"	String	Module1.DrawCircle
⊞ Hyperlinks		IAcadHyperlinks/IAca	Module1.DrawCircle
Layer	"0"	String	Module1.DrawCircle

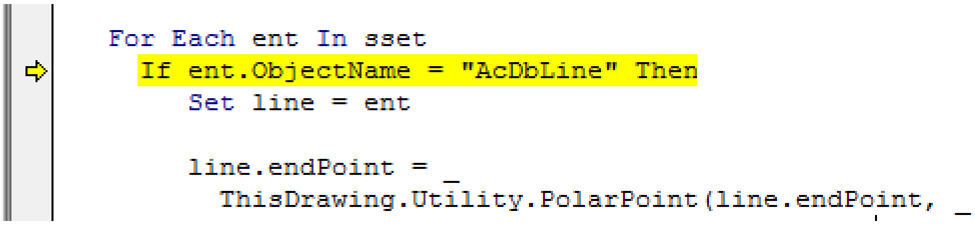
24. On the Debug toolbar, click Continue. ▶

    Execution runs as normal and the AU2013 dialog box is displayed.

25. In the AU2013 dialog box, click Modify Objects.

26. At the `Select objects:` prompt, select some circle and/or line objects in the drawing and press Enter.

    Execution continues until the value of the *ent* variable is changed as indicated by the Watch you set earlier in step 11.



```
For Each ent In sset
  If ent.ObjectName = "AcDbLine" Then
    Set line = ent

    line.endPoint = _
      ThisDrawing.Utility.PolarPoint(line.endPoint, _
```

27. In the Watches window, review the value assigned to the *ent* variable.

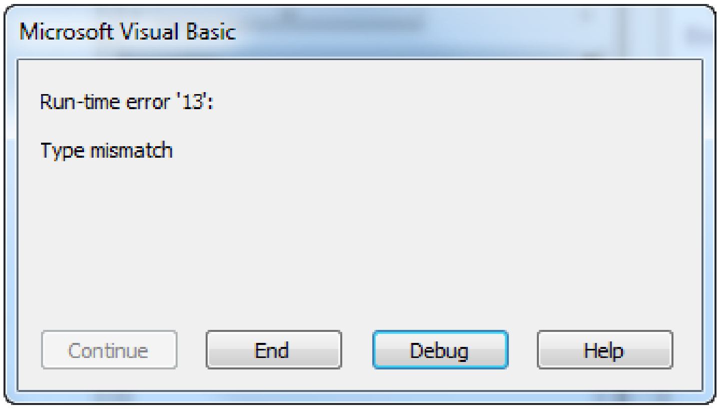
28. On the Debug toolbar, click Continue. ▶

    Execution continues until the value of the *ent* variable changes again.

29. Step through the code using the Step Into or Step Out buttons, and keep clicking Continue ▶ until you are returned to the AU2013 dialog box.

30. In the AU2013 dialog box, clear both text boxes if they contain a value, and click Create Layer.

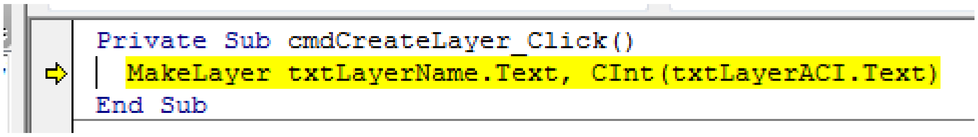
    The Microsoft Visual Basic dialog box is displayed.



```
Run-time error '13':

Type mismatch
```

31. On the Microsoft Visual Basic dialog box, click Debug.

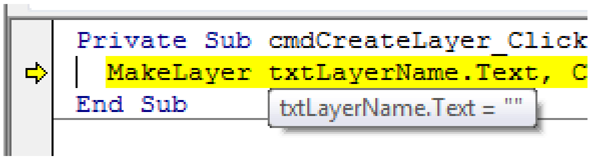
    Execution is suspended and you are sent to the code editor window for the frmMain UserForm.



```
Private Sub cmdCreateLayer_Click()
    MakeLayer txtLayerName.Text, CInt(txtLayerACI.Text)
End Sub
```

32. Position the cursor over the text `txtLayerName.Text`.

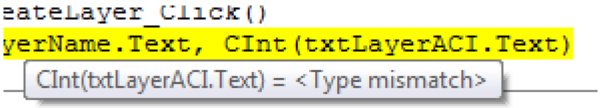
    A tooltip with the statement's current value is displayed. While a blank text string will be a problem later in the `MakeLayer` function that is not the cause of the problem.



```
Private Sub cmdCreateLayer_Click
    MakeLayer txtLayerName.Text, C
End Sub
txtLayerName.Text = ""
```

33. Position the cursor over the text `CInt`.

    A tooltip with the statement's current value is displayed. Notice the tooltip contains the text <Type mismatch> which matches the error message previously displayed. This is caused because `CInt` cannot convert an empty text string to a number value.



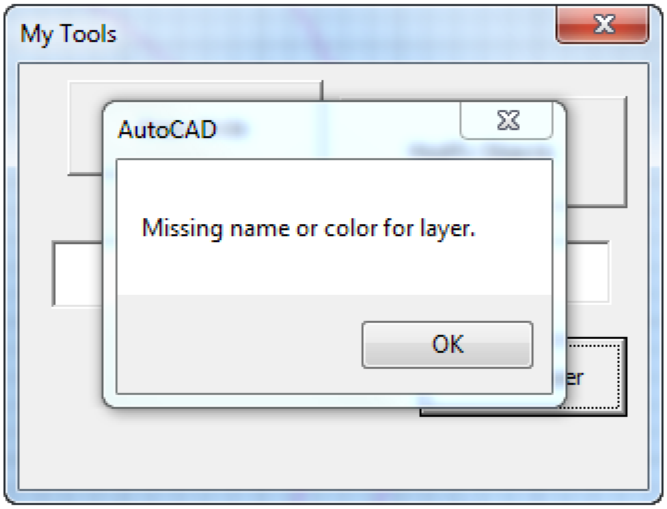
```
eateLayer_Click()
yerName.Text, CInt(txtLayerACI.Text)
CInt(txtLayerACI.Text) = <Type mismatch>
```

34. On the Debug toolbar, click Reset. ■

35. In the Project Explorer, right-click frmMain and click View Code.
36. In the code editor window, change `cmdCreateLayer_Click` procedure so it looks like the following:

```
Private Sub cmdCreateLayer_Click()
  If txtLayerName.Text <> "" And txtLayerACI.Text <> "" Then
    MakeLayer txtLayerName.Text, CInt(txtLayerACI.Text)
  Else
    MsgBox "Missing name or color for layer."
  End If
End Sub
```

37. On the menu bar, click File ➢ Save.
38. In the Project Explorer, double-click basMain to open it in the code editor window.
39. Scroll to and click between the two statements that define the `ShowMain` procedure.
40. On the menu bar, click Run ➢ Run Sub/UserForm.
41. In the AU2013 dialog box, leave both text boxes empty and click Create Layer.



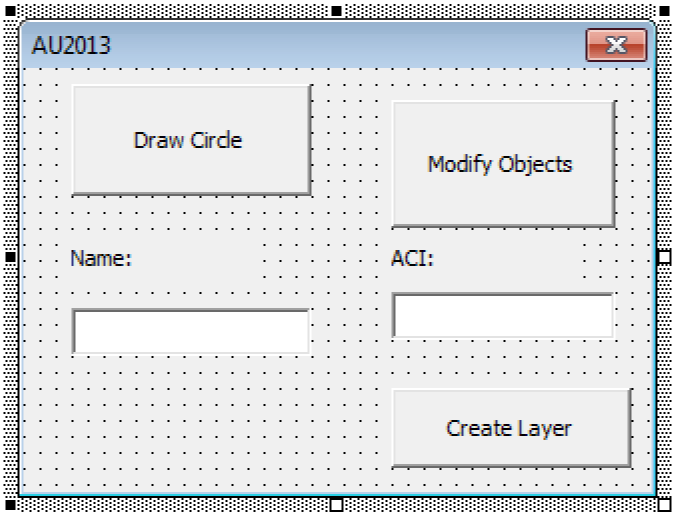
Missing name or color for layer.

42. In the message box, click OK to continue.
43. In the AU2013 dialog box, type a name and color for the layer and then click Create Layer.
44. Close the dialog box when done.
45. Switch to the AutoCAD application window. On the ribbon, click Home tab ➢ Layers panel ➢ Layer drop-down list.

E10 Formatting Controls on a UserForm

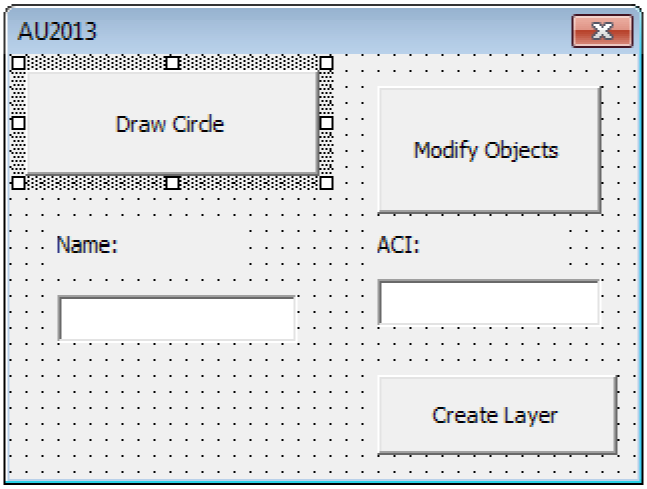
This exercise explains how to use some of the UserForm formatting tools.

1. In the VBA IDE, Project Explorer, double-click frmMain to view it in the form editor window.



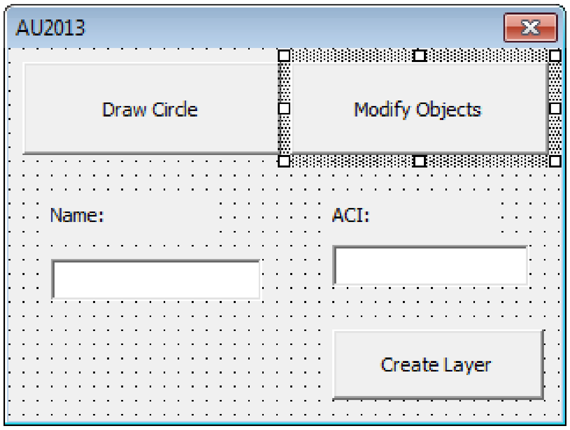
2. In the form editor window, select the Draw Circle button.

3. In the Properties window, change the following properties for the button:
   - Height = 40
   - Left = 6
   - Top = 6
   - Width = 110

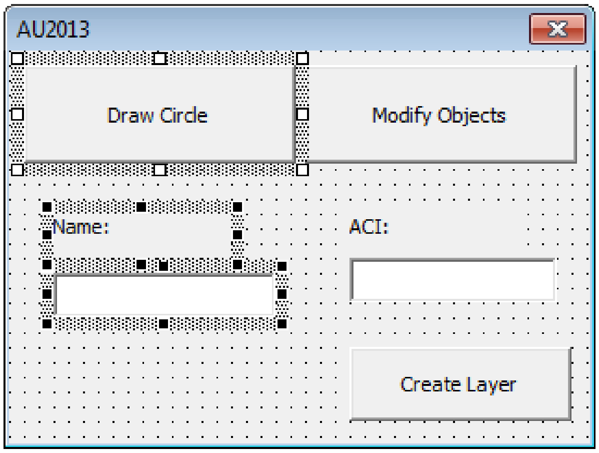


4. In the form editor window, select the Modify Objects button.

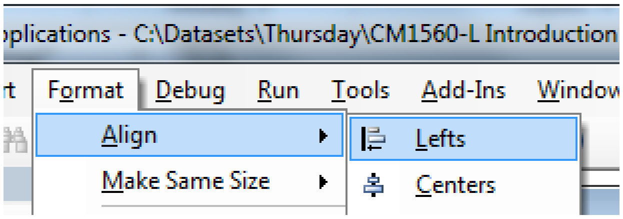
5. In the Properties window, change the following properties for the button:
   - Height = 40
   - Left = 120
   - Top = 6
   - Width = 110



6. Along the left side of the UserForm, select the text box.
7. Hold down the Ctrl key. Select the label and the Draw Circle button; in that order.

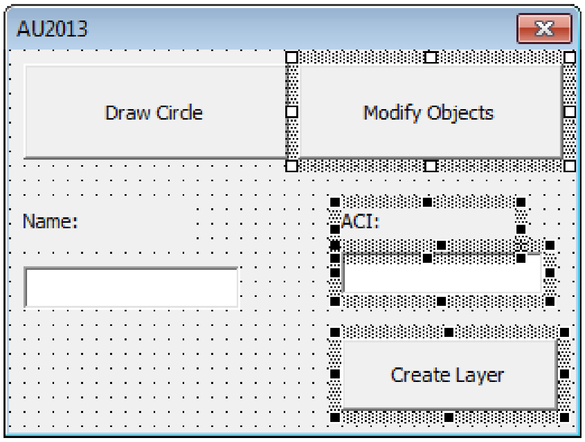


8. On the menu bar, click Format ➢ Align ➢ Lefts.



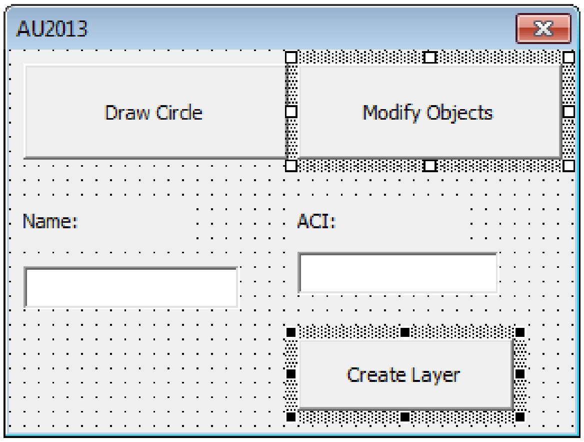
9. Along the right side of the UserForm, select the Create Layer button.

10. Hold down the Ctrl key. Select the text box, label, and the Modify Objects button; in that order.

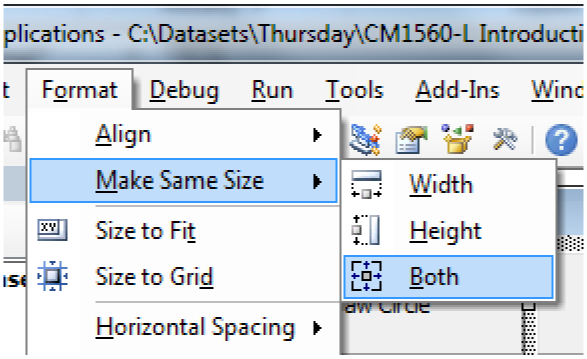


11. On the menu bar, click Format ➢ Align ➢ Lefts.

12. Click the UserForm and then select the Create Layer button. Hold down the Ctrl key and select Modify Objects button.



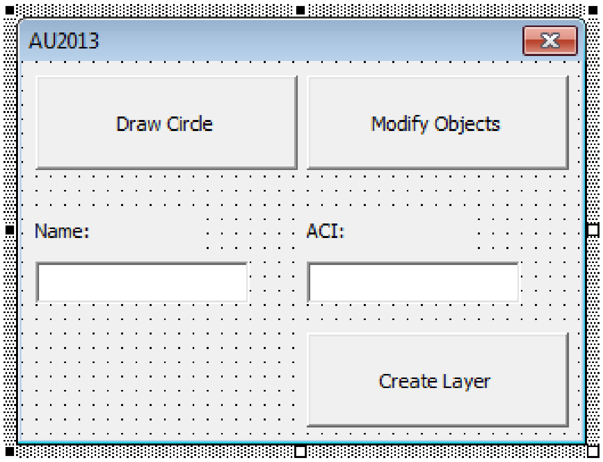
13. On the menu bar, click Format ➢ Make Same Size ➢ Both.



14. Click the UserForm, and then select the Create Layer button. Drag the Create Layer button up/down as needed.

15. Align the labels and text boxes using the tools on the Format menu.

    The end result should be a cleaner looking UserForm. Yours does not need to look like the following in the end, but gives you an idea what you might want to do.



16. In the VBA IDE, on the menu bar, click File ➢ Save.

E11 Exporting AutoCAD Information to a Spreadsheet

This exercise explains how to create a reference to Microsoft Excel and write out information to a spreadsheet. This could be a great way to do estimating or generate information from a drawing for a bid package.

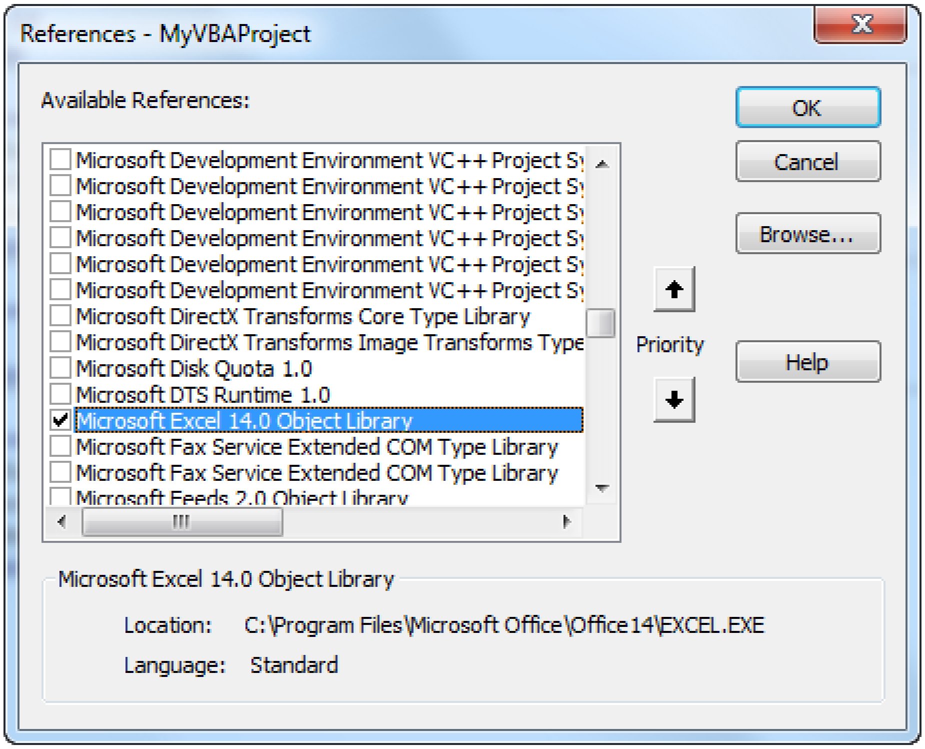
1. In the VBA IDE, Project Explorer, double-click frmMain to view it in the form editor window.

2. Add a new CommandButton to the UserForm, and then change the following properties for the button:
   - (Name) = cmdExport
   - Caption = Export

3. Double-click the button to open the code editor window.

4. On the menu bar, click Tools ➢ References.

5. In the References dialog box, scroll down and locate Microsoft Excel 14.0 Object Library. Click the library reference and then click OK.



□ Microsoft Development Environment VC++ Project Sy
□ Microsoft Development Environment VC++ Project Sy
□ Microsoft Development Environment VC++ Project Sy
□ Microsoft Development Environment VC++ Project Sy
□ Microsoft Development Environment VC++ Project Sy
□ Microsoft Development Environment VC++ Project Sy
□ Microsoft DirectX Transforms Core Type Library
□ Microsoft DirectX Transforms Image Transforms Type
□ Microsoft Disk Quota 1.0
□ Microsoft DTS Runtime 1.0
☑ Microsoft Excel 14.0 Object Library
□ Microsoft Fax Service Extended COM Type Library
□ Microsoft Fax Service Extended COM Type Library
□ Microsoft Feeds 2.0 Object Library

Microsoft Excel 14.0 Object Library

Location: C:\Program Files\Microsoft Office\Office14\EXCEL.EXE

Language: Standard

6. In Windows Explorer or File Explorer, browse to *C:\Datasets\Thursday\CM1560-L Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®* and double-click the *EX11 – Export.txt* file.

7. In Notepad, highlight all the code and right-click. Click Copy.

8. Switch back to the VBA IDE. Click between the two statements that define the `cmdExport_Click` procedure. Right-click and click Paste to add the copied code.

```
Private Sub cmdExport_Click()
  ' Create a reference to a new Excel application
  Dim execl As New Excel.Application
  execl.Visible = True

  ' Create a new workbook
  Dim workbk As Excel.workbook
  Set workbk = execl.workbooks.Add

  ' Get the first worksheet
  Dim worksht As Excel.worksheet
  Set worksht = workbk.worksheets(1)

  ' Add Intro Text to cell A1
  Dim workshtRange As Excel.Range
  Set workshtRange = worksht.Range("A1")
  workshtRange.Value = "AU2013 - Introduction to Visual Basic®
for Applications for Autodesk® AutoCAD®"

  ' Add column headers
  worksht.Range("A2").Value = "Handle"
  worksht.Range("B2").Value = "Radius"

  ' Step through each of the objects in the current space
  ' and extract only the circles
  Dim ent As AcadEntity
  Dim circ As AcadCircle

  Dim row As Integer
  row = 3

  For Each ent In ThisDrawing.ModelSpace
    If ent.ObjectName = "AcDbCircle" Then
      Set circ = ent

      worksht.Range("A" & CInt(row)).Value = "'" & circ.Handle
      worksht.Range("B" & CInt(row)).Value = CStr(circ.radius)

      row = row + 1
    End If
  Next ent
End Sub
```

9. On the menu bar, click File ➢ Save.

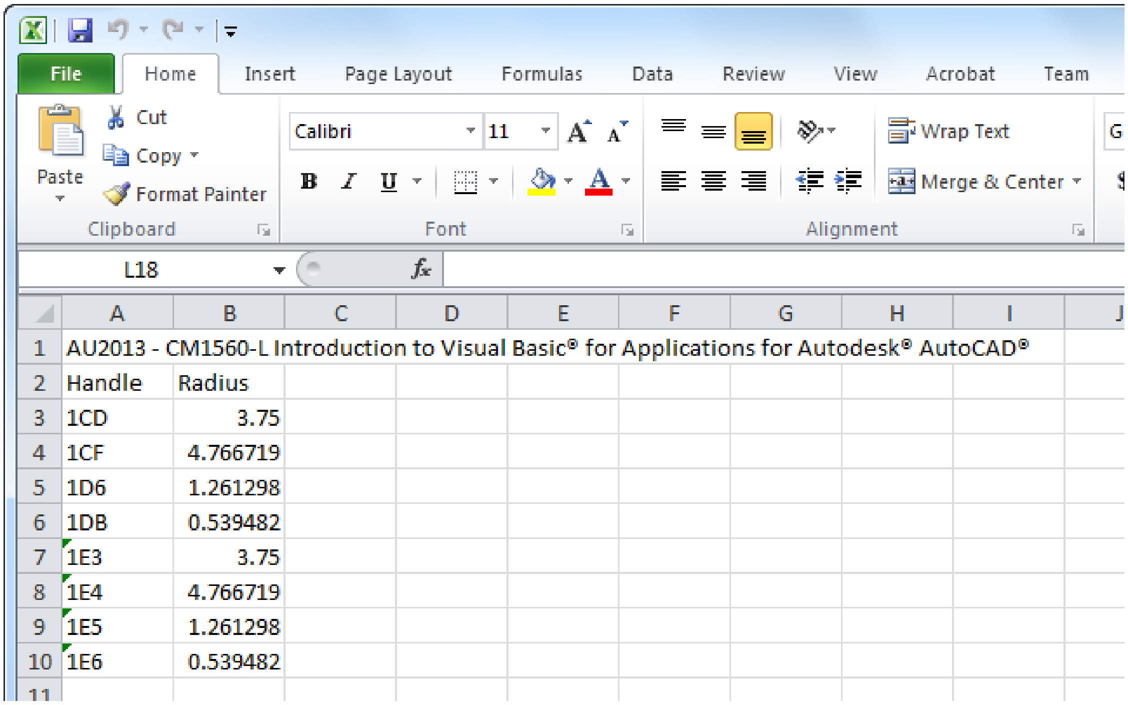
10. In the Project Explorer, double-click basMain to open it in the code editor window.

11. Scroll to and click between the two statements that define the `ShowMain` procedure.

12. On the menu bar, click Run ➢ Run Sub/UserForm.

13. In the AU2013 dialog box, click Export.

    The following image provides an idea what the output should look like in Microsoft Excel.



	A	B	C	D	E	F	G	H	I	J
1	AU2013 - CM1560-L Introduction to Visual Basic® for Applications for Autodesk® AutoCAD®									
2	Handle	Radius								
3	1CD	3.75								
4	1CF	4.766719								
5	1D6	1.261298								
6	1DB	0.539482								
7	1E3	3.75								
8	1E4	4.766719								
9	1E5	1.261298								
10	1E6	0.539482								
11										

14. Close Microsoft Excel and discard the changes made.

15. Close the AU2013 dialog box.