

# **VB Scripting for CATIA V5: Expanded Edition**

Workshop #2

Nick Weisenberger

<http://www.scripting4v5.com>

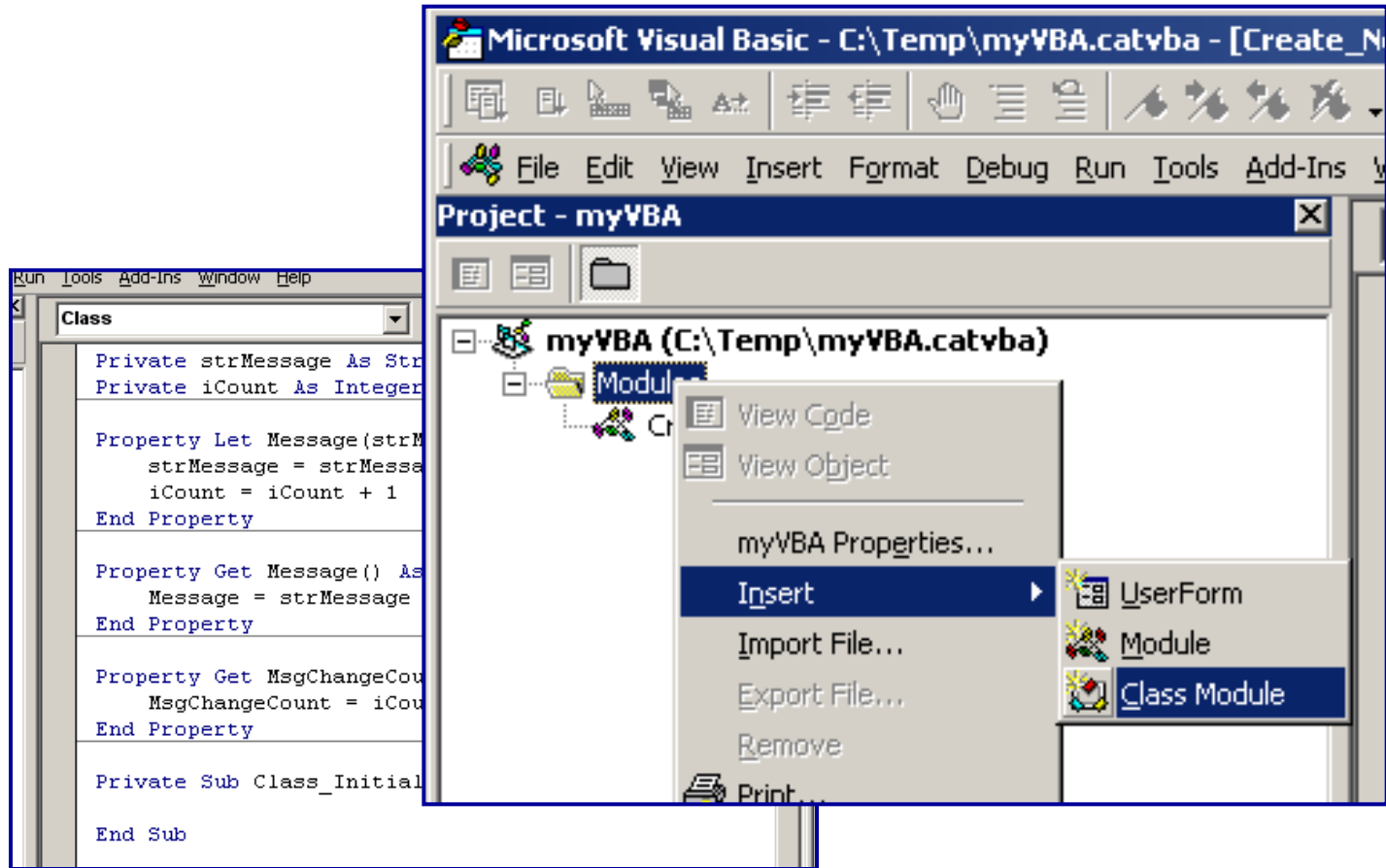
Copyright 2012 by Nick Weisenberger

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing by the author. The only exception is by a reviewer, who may quote short excerpts in a review.

This free tutorial is an excerpt from VB Scripting for CATIA V5: Expanded Edition.  
Please visit [www.scripting4v5.com](http://www.scripting4v5.com) to purchase the full version.

# WORKSHOP 2

## Creating Your Own VBA Modules and Classes Tutorial



## ■ Description

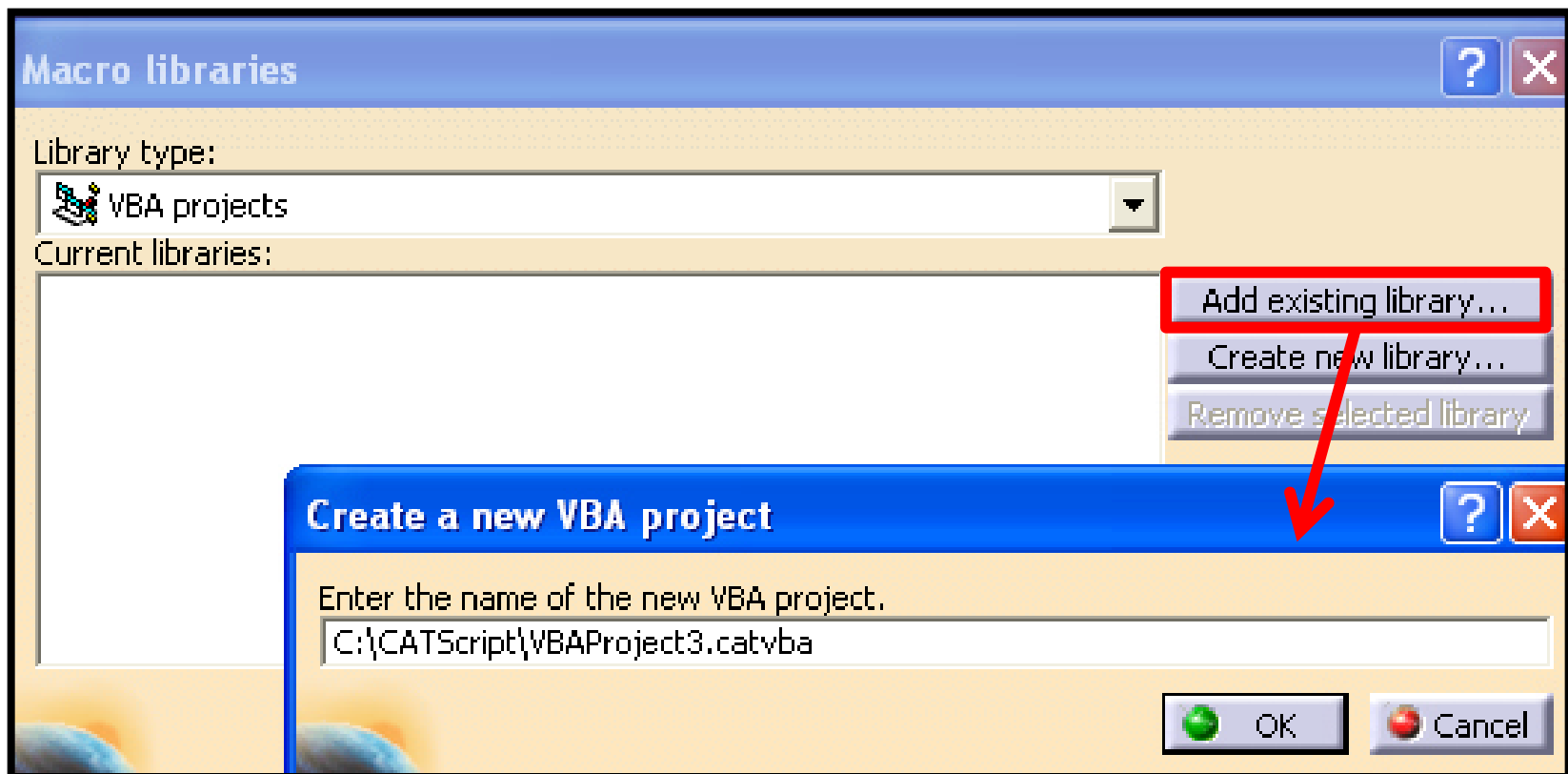
Although most of the CATIA VBA programming that you will do will involve the use of classes that are defined by the CATIA programming API, you will likely find it useful to define your own classes. In this workshop you will create a custom class in order to demonstrate the fundamentals of VBA object design.

## ■ Outline

1. Create a new class module in a new VBA library named “Messenger”.
2. Give the Messenger class a simple “Public” property and use the class in a CATIA macro.
3. Enhance the Messenger class so that the “getting” and “setting” of its properties are controlled by “Get” and “Let” methods.
4. Give the Messenger class a method called “Capitalize” that serves to manipulate the string information that is stored in its properties.

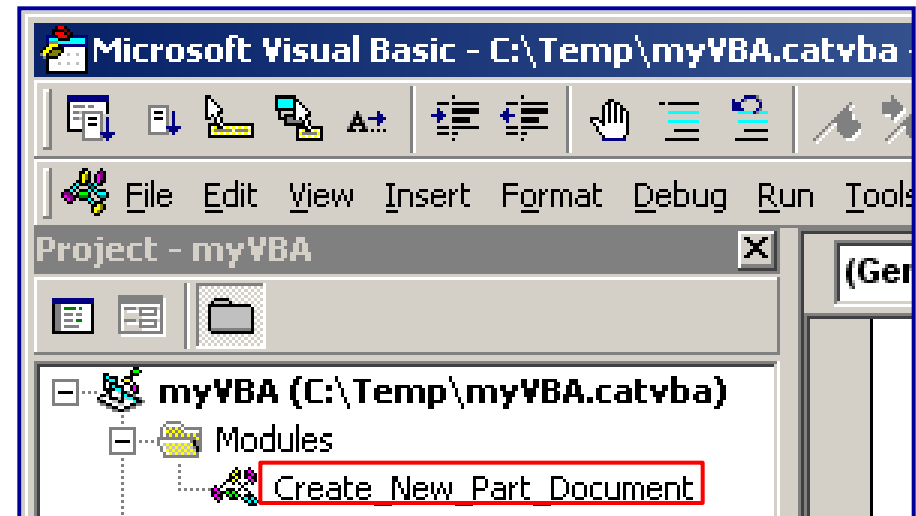
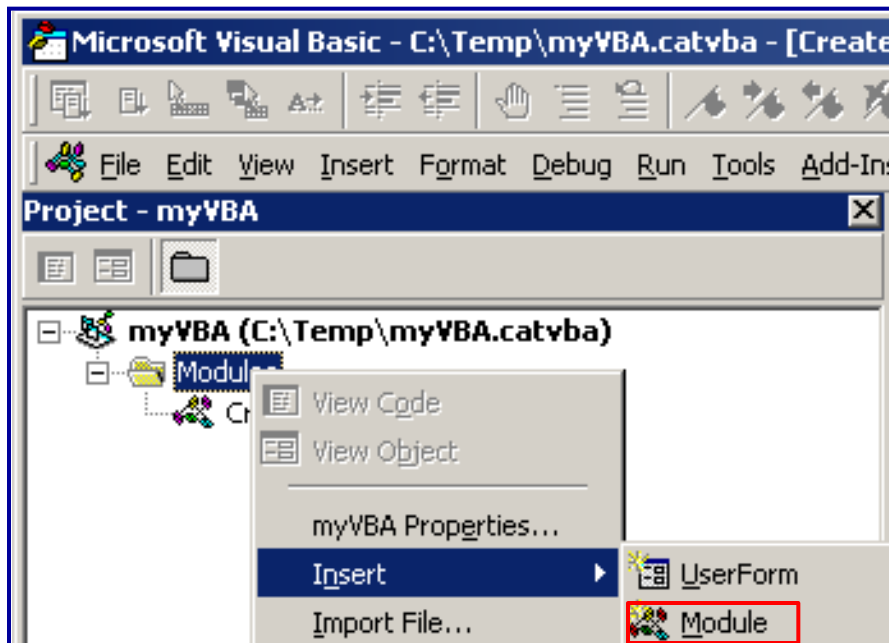
# Step 1

- Open the VBA editor by hitting Alt + F11. Create a new macro library and VBA project, called "myVBA".
- Double click on your newly created library (which will appear in Current Libraries).



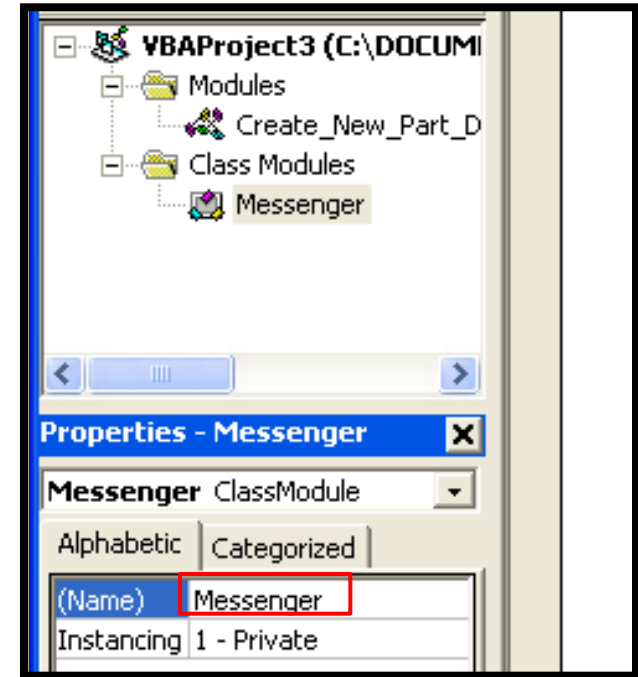
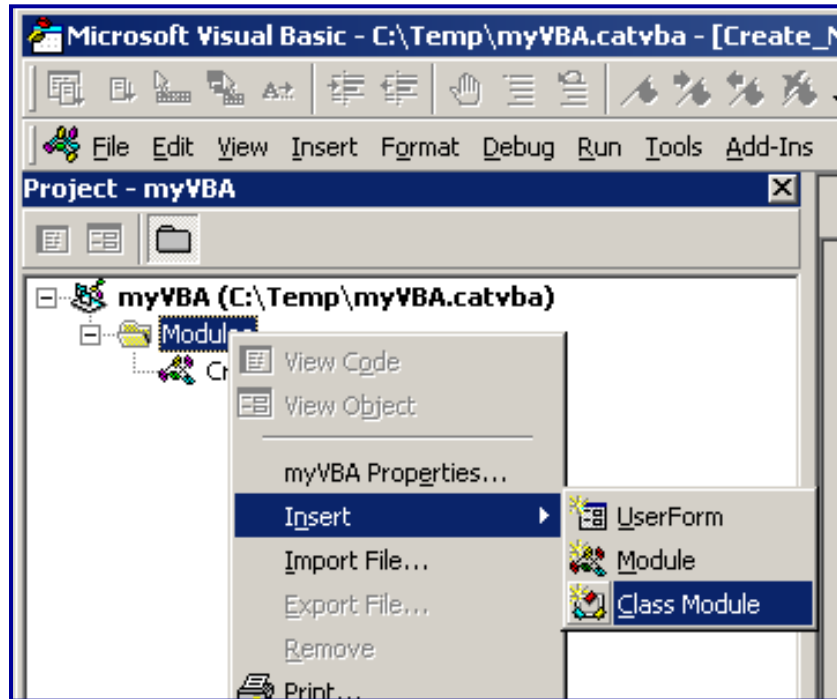
## Step 2

- Ensure the project and properties boxes are visible by going to the top menu bar and clicking View >View Project Explorer and View Properties Window
- Right click on VBAProject (myVBA in this example) > Insert> Module.
- Use the (Name) field in the properties box to rename it  
"Create\_New\_Part\_Document"

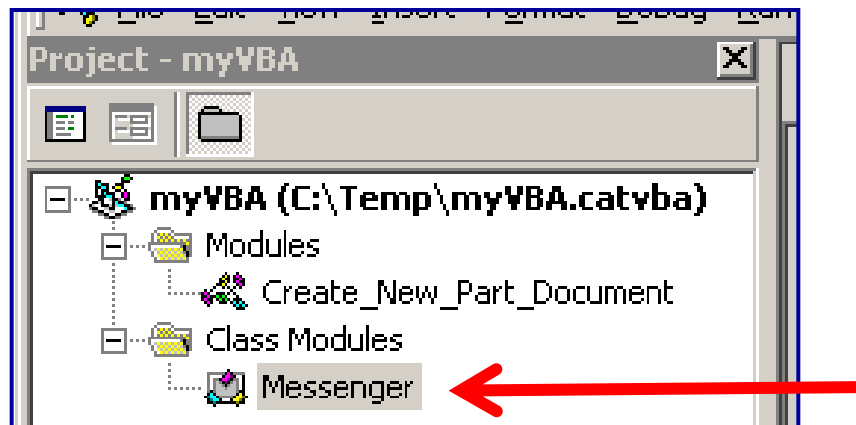


## Step 3

Right click on the “Modules” folder of the VBA project and select Insert > Class Module.



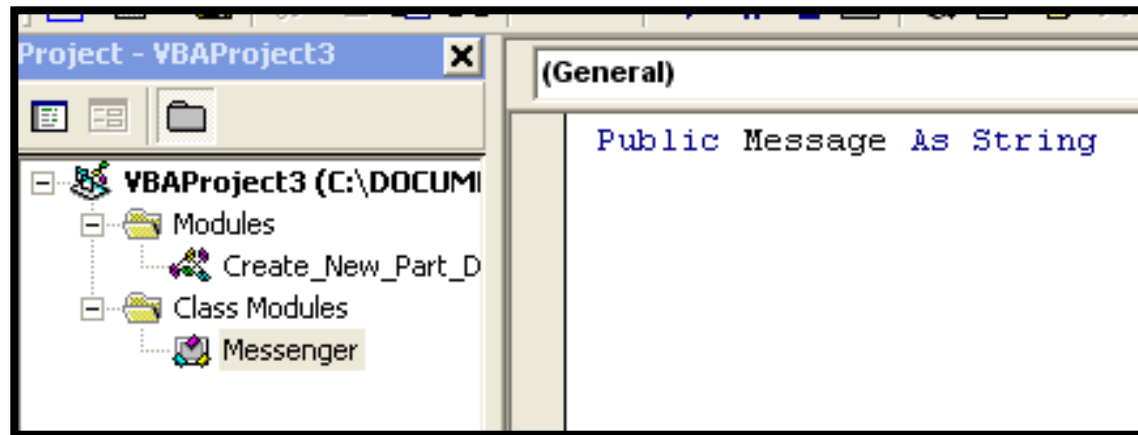
Rename the resulting class module as “Messenger” by clicking on Class1 then renaming the “Name” field in the properties box.



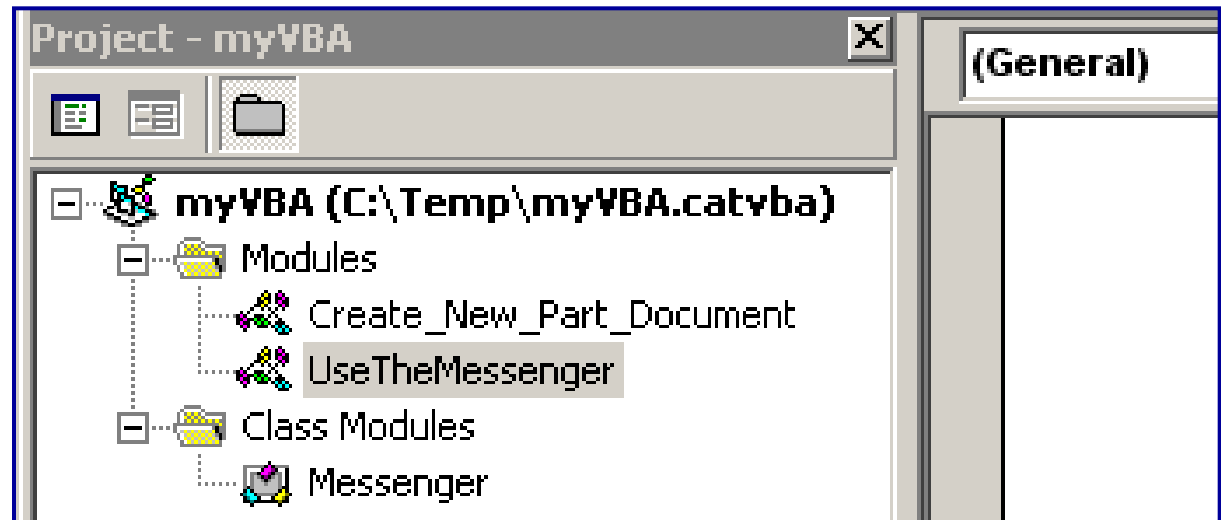
## Step 4

- In the code window enter the line:

***Public Message As String***



- Create another module on the VBA project. This needs to be a regular module, not a class module. Name it "UseTheMessenger".





## Step 5

Enter the code seen to the right into the “UseTheMessenger” module. Note the following:

- A Messenger object is dimmed. Because you created the class module named “Messenger” this class is immediately available in your VBA project.
- The “New” keyword orders the creation of a Messenger object. The variable “oMssgr” is set to this new object
- The property “Message” gets set to a specific value here.
- The Message property then gets used as it is passed to the message box.

Run the UseTheMessenger module by clicking the play button icon. A message box displaying “Hello” should result in your CATIA Window.

```
Sub CATMain()
```

```
Dim oMssgr As Messenger  
Set oMssgr = New Messenger  
oMssgr.Message = "Hello"  
MsgBox oMssgr.Message
```

```
End Sub
```



## Step 6

---

- Return to the code window for the class module “Messenger” and make the changes shown below. These changes have the effect of “hiding” the strMessage variable, but then create a read-only property named “Message” whose value is stored in the strMessage variable.
- Now go back to the “UseTheMessenger” module and make the changes shown below and to the right.

```
Private strMessage As String
```

```
Property Let Message(MessageIN As String)
```

```
    strMessage = MessageIN
```

```
End Property
```

```
Sub CATMain()
```

```
    Dim oMssgr As Messenger
```

```
    Set oMssgr = New Messenger
```

```
    oMssgr.Message = "Hello"
```

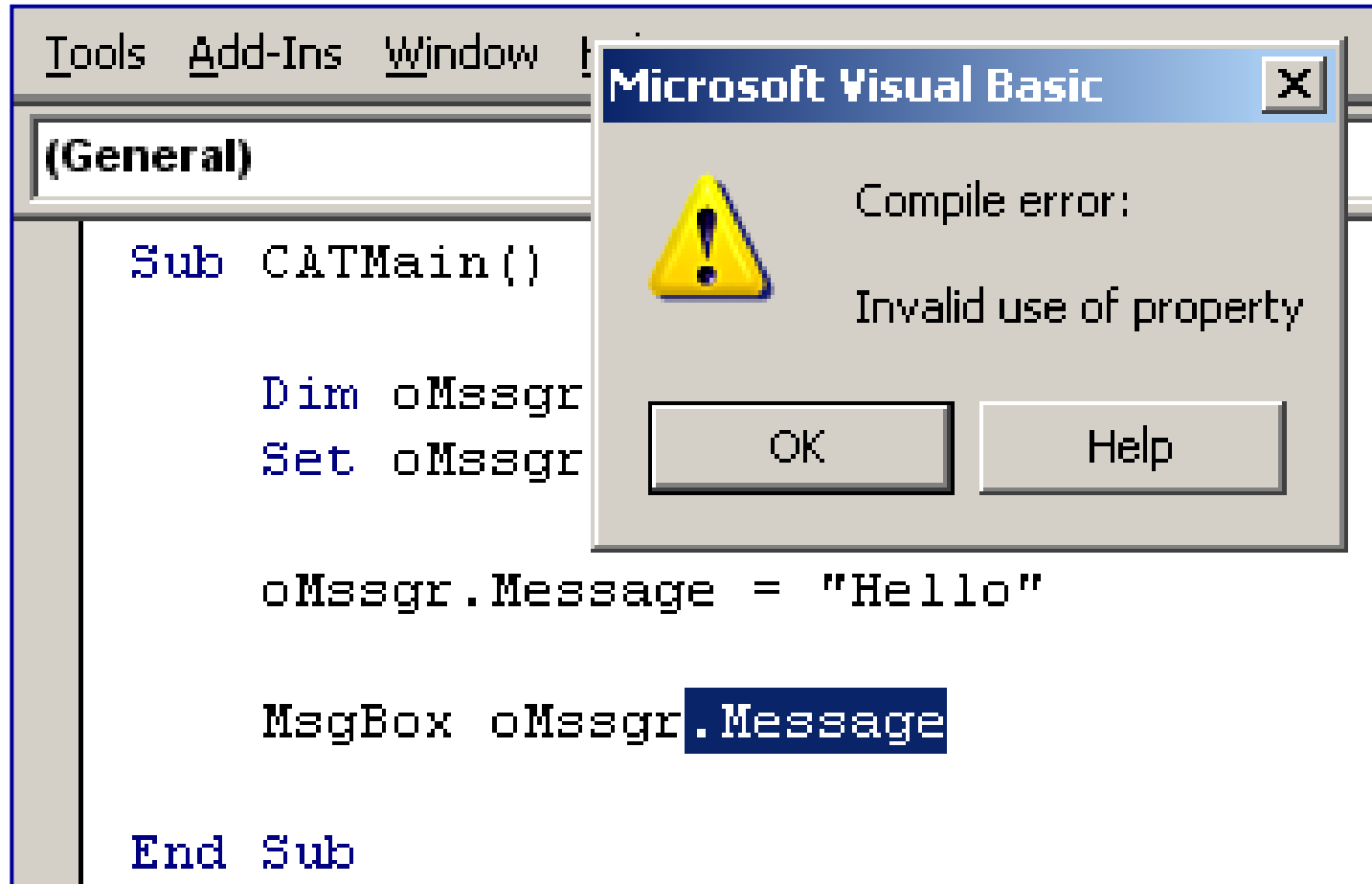
```
    MsgBox oMssgr.Message
```

```
End Sub
```

## Step 7

---

Attempt to run “UseTheMessenger” and note that although the Message property can be “set”, it fails when the property is “gotten”. This is because no “Property Get” method has been defined (although, “Property Let” has been defined).



## Step 8

---

- Return to the code window for the class module “Messenger” and add a “Property Get” method as shown below.
- Run the UseTheMessenger module again. It should work and the message “Hello” should be displayed.

Note: the advantage of strictly defining these “Let” and “Get” methods is that it gives the programmer control over whether a variable is read-only or read-write. Also, the code that is in the “Let” and “Get” methods can contain more complex operations and logic.

```
Private strMessage As String
```

```
Property Let Message(strMessageIN As String)
```

```
    strMessage = strMessageIN
```

```
End Property
```

```
Property Get Message() As String
```

```
    Message = strMessage
```

```
End Property
```

## Step 9

---

Assume it would be desirable to keep count of how many times the Message property's value is changed. To do this, create a private integer variable "iCount", change the "Property Let" method of the Message property, and add a "Property Get" method for a new property named "MsgChangeCount".

```
Private strMessage As String
```

```
Private iCount As Integer
```

```
Property Let Message(strMessageIn As String)
```

```
    strMessage = strMessageIn
```

```
    iCount = iCount + 1
```

```
End Property
```

```
Property Get Message() As String
```

```
    Message = strMessage
```

```
End Property
```

```
Property Get MsgChangeCount() As Integer
```

```
    MsgChangeCount = iCount
```

```
End Property
```

## Step 10

---

- Change the “UseTheMessenger” module as shown below. Run it and a message should be displayed saying “Message changed 1 times”.
- Change the “UseTheMessenger” module as shown to the lower right and run it again. A message should be displayed saying “Message changed 2 times”.

```
Sub CATMain()  
  
Dim oMssgr As Messenger  
Set oMssgr = New Messenger  
  
oMssgr.Message = "Hello"  
  
MsgBox "Message changed " &  
oMssgr.MsgChangeCount & " times."  
  
End Sub
```



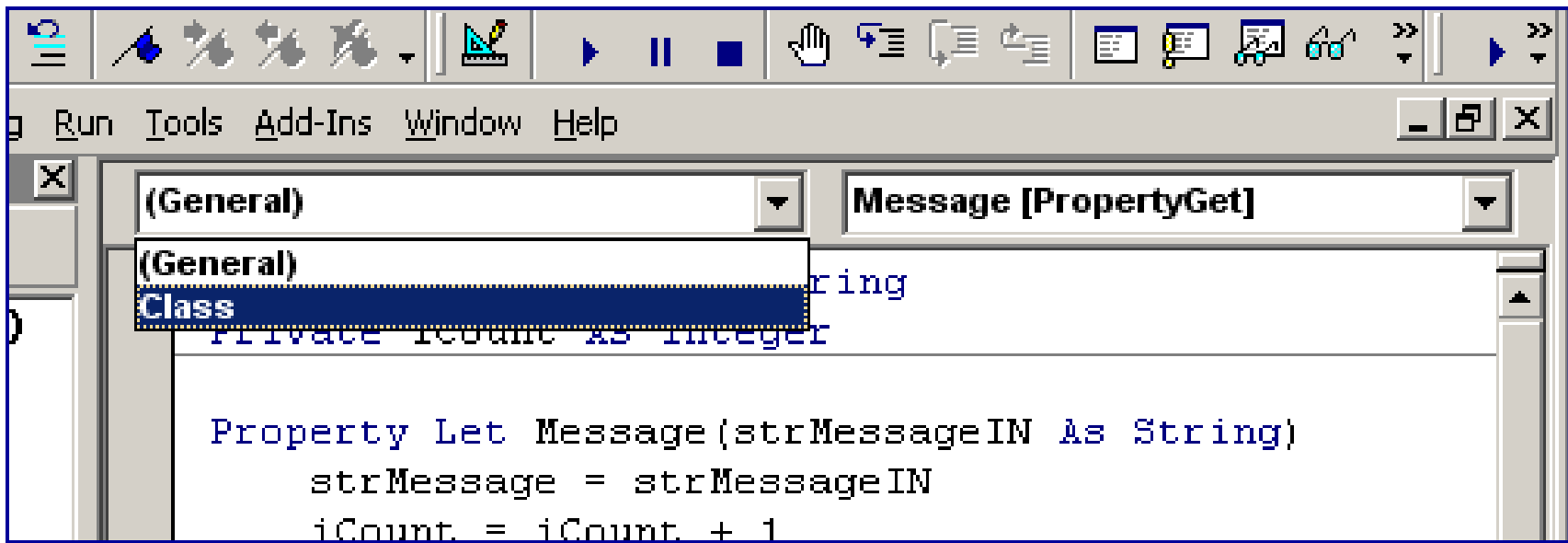
```
Sub CATMain()  
  
Dim oMssgr As Messenger  
Set oMssgr = New Messenger  
  
oMssgr.Message = "Hello"  
oMssgr.Message = "Hello again"  
  
MsgBox "Message changed " &  
oMssgr.MsgChangeCount & " times."  
  
End Sub
```



## Step 11

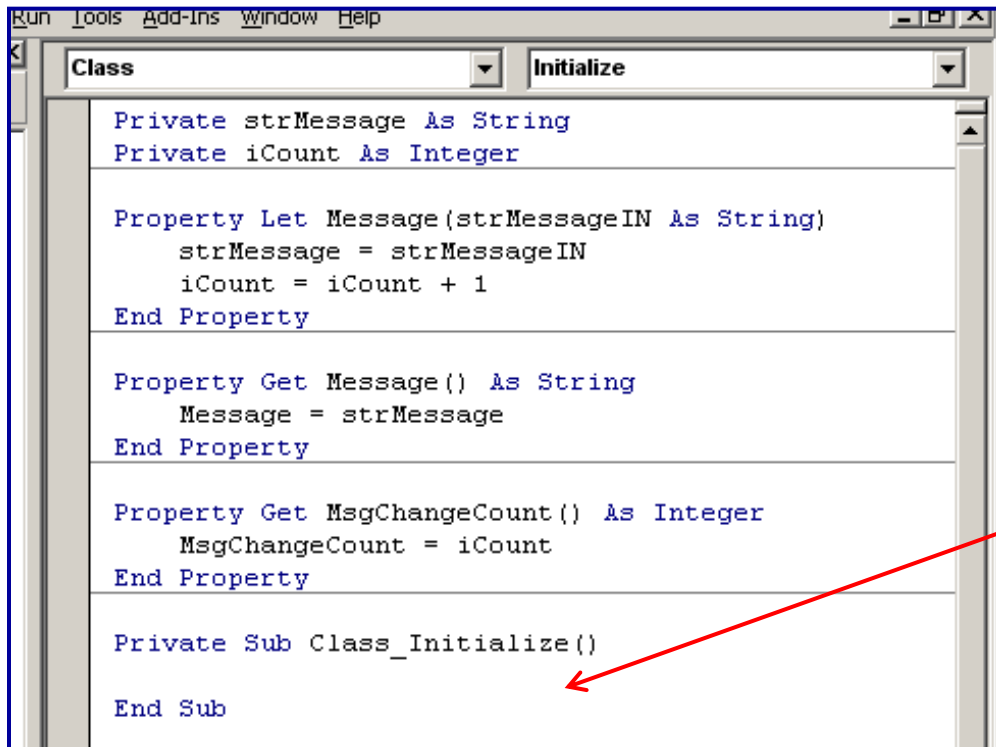
---

- Although the class works presently, there is one area where the class code could be more explicit. The “iCount” variable is incremented by one every time the Message property is changed, but it isn’t clear what value iCount starts at. Testing the code has shown that it does start at zero, however it’s best to be explicit.
- In the code window for the Messenger class module, click the left drop-down menu and choose “Class”.



## Step 12

The result of the previous step should be that the text for the subroutine “Class\_Initialize” appears in the code window. Enter the code shown below into this sub. This has the effect of setting the value of the iCount to zero when a Messenger object is created with the “New” command.



The screenshot shows a code editor window with a menu bar (Run, Tools, Add-Ins, Window, Help) and a title bar (Class, Initialize). The code is as follows:

```
Private strMessage As String
Private iCount As Integer

Property Let Message(strMessageIN As String)
    strMessage = strMessageIN
    iCount = iCount + 1
End Property

Property Get Message() As String
    Message = strMessage
End Property

Property Get MsgChangeCount() As Integer
    MsgChangeCount = iCount
End Property

Private Sub Class_Initialize()

End Sub
```

**Private Sub Class\_Initialize()**

**iCount = 0**

**End Sub**



## Step 13

---

- Properties have been defined for the messenger class. Now create a method for this class. This method will capitalize the message that is stored in the Message property. Enter the code seen below in the Messenger class module.
- The function “UCase” is a standard VBA function that takes a String as an argument and returns the same string in all capital letters.

```
Private strMessage As String
Private iCount As Integer

Property Let Message(strMessageIN As String)
    strMessage = strMessageIN
    iCount = iCount + 1
End Property

Property Get Message() As String
    Message = strMessage
End Property

Property Get MsgChangeCount() As Integer
    MsgChangeCount = iCount
End Property

Public Sub Capitalize()
    strMessage = UCase(strMessage)
End Sub
```

## Step 14

---

- Change the “UseTheMessenger” code so that it calls this new capitalize method as shown.
- Run the code. Note that the capitalize method has the intended effect of changing the message box text from “Hello” to “HELLO”.

```
Sub CATMain()
```

```
Dim oMssgr As Messenger
```

```
Set oMssgr = New Messenger
```

```
oMssgr.Message = "Hello"
```

```
oMssgr.Capitalize
```

```
MsgBox oMssgr.Message
```

```
End Sub
```



## Workshop 2 – Conclusion

---

This concludes Workshop 2. In this workshop you have learned to:

- Create a VBA class
- Define a read-only property for the class (only provide a “Property Let” method)
- Define a read-write property for the class (provide both “Property Let” and “Property Get” methods)
- Use the “Class\_Initialize” subroutine to assign initial values to class variables
- Define a method for a class that operates on private variables of that class

## Tools, Links, and Resources

In addition to the CATIA macro resources that have proved invaluable to me, I rely on a number of other products and services in my everyday life. Below is a list of all the tools I use, a description of each, and where I employ them at so you can see for yourself how they work. If you're thinking about starting your own blog, website, newsletter, or business I **highly recommend** using these services:

### Microsoft Excel Resources

[Microsoft Excel 2012 In Depth by Bill Jelen](#) From Mr Excel himself.

[Excel Video Training Course - Beginner & Advanced](#) – Teach yourself Excel with this course.

[Scheduler123](#) : Affordable Excel based production system

[Excel Spreadsheets Help](#): Great blog with free downloadable Excel spreadsheet templates for all of your needs.

### Web Site Hosting

[Namecheap](#): Very affordable site to quickly and easily register your perfect domain name. I registered [www.scripting4v5.com](http://www.scripting4v5.com) here.

[Hostgator](#): Website host with easy to use, 1-click automatic WordPress installation, and excellent customer service. I highly recommended this for your first web site.

## Themes, Plugins, and Tools

[Socrates](#): This is the WordPress theme and framework that scripting4v5 runs on, with built in navigation and easy customization.

[WP FileLock](#): Protect your download pages and documents. Great for niche profit websites or ebook sellers.

[ShareCash](#): File hosting for downloads – make \$0.30 to \$0.60 per download!

## SEO, Marketing, and Advertising

- [Traffic Travis](#): One of the best SEO optimization tools around, Traffic Travis easily lets you analyze your web site's page rank, back links, keyword research, and more.
- [Aweber](#): The internet's most powerful email opt-in service and email marketing/broadcasting tool. This is what I use to build my email newsletters for my web sites.
- [Kontera](#): Make money by placing relevant in text ad links on your sites. I use this on the majority of my blogs.

## Miscellaneous Resources and Goodies

- [JustCloud](#): Backup and protect your computer files online in the cloud and access them from anywhere at anytime. Awesome service for when I want to switch between working on the desktop to the laptop, or get a file when visiting the in-laws.
- [Bigstock](#) – Site where I get really amazing pictures and images to use on ApartmentHunterHQ.com and my other blogs.
- [oDesk](#): Outsource website for freelancers, CAD designers, programmers, and others. Some of the articles on my web sites were written by Virtual Assistants hired on oDesk, freely me up to do more important tasks. oDesk makes it very easy to find the best person for the job available.

Some of these premium services do come with a price (just being honest here), but seriously I wouldn't risk mentioning them here if I didn't know they work great and will save a lot of people time and money.

*Disclosure: Please note that some of the proceeding links are affiliate links and I will earn a commission if you purchase through those links (at no extra cost to you). These are all things that I have experience with that I am recommending because they are helpful and are companies that I trust, not because of the commissions that I may earn from you using these products. Please do not spend money unless you feel it's for something you really need and will help you reach your goals –they helped me reach mine!*