

Forum:[Hack Forums](#)

Thread: [My Polymorphic numbergen TuT and full \[source\]](#)

GodWall - April 26, 2011, 3:43 am

Ok HF This is it a true "Now public" polymorphic source. I wrote this in 2010 and did post an example here, Anyways I have had it to myself for a few years and now its time to make it public, but I don't want you to just copy/paste it I want you to understand it so I'm releasing it in the form of a tut. Theirs 5 parts I spent a few hours commenting them for you, and even made a downloadable 4 part version you can compile each part until you truly get it.

Basically I'm taking the original example I posted and walking you through the full conversion to a workable Gen.

Part 1. My Basic poly source

Part 2. Add random numbers

Part 3. Set to a secure random Key

Part 4. My Polymorphic Number Gen

Part 5. Final Source

and the download for those who need it

Note: I suck at TUT's (sorry if its basic)

Part 1. My Basic poly source ?

1.button

Code 'Working Example of True Polymorphism in VB.Net

'By GodWall

'Polymorphic Source

'2010

'This is just a basic example

'Parent class

Public Class Form1

Const GW1 As Double = 2

'This number would be your original value

Const GW2 As Double = 9

'this number to set poly x10 "because of below #"

'child first class

'GW1

Public Class BasePoly

Overridable Function PolyCrypt(ByVal Addpoly As Double) As Double

Return Addpoly * GW1

End Function

End Class

'child second class

'GW2

```

Public Class ModPoly
    Inherits BasePoly
    Private BaseAddpoly As Double
    Overrides Function PolyCrypt(ByVal Addpoly As Double) As Double
        BaseAddpoly = MyBase.PolyCrypt(Addpoly)
        Return GW2 * (BaseAddpoly + Addpoly) + BaseAddpoly
    End Function
End Class

```

```

Sub TestPoly()
    Dim GodWall1 As New BasePoly
    Dim GodWall2 As New ModPoly
    ShowPoly(GodWall1, 10)      'This number is original value x10
    ShowPoly(GodWall2, 10)      'This number is original value x10 + Set#
    'The set is affected by this second number above
    'During Decryption these numbers and set numbers
    'Must stay the same so you can get the original value back
    'Changing the numbers randomly be4 encryption will give you
    'Random encryptions of the value each time
    'Feel free to play around with the numbers so you get the idea
End Sub

```

```

Sub ShowPoly(ByVal GodWall As BasePoly, ByVal Addpoly As Double)
    Dim polyplus As Double
    polyplus = GodWall.PolyCrypt(Addpoly)
    MsgBox("This is True Polymorphism " & Format(polyplus, "G"))
End Sub

```

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    TestPoly()
End Sub
End Class

```

Part 2. Add random numbers ?

Add another Button and a textbox to your form

Code 'Working Example of True Polymorphism in VB.Net

'By GodWall

'Polymorphic Source

'2010

'This is just a basic example with random value

'Notice the aplication of the set key in message 2

'Parent class

Public Class Form1

'Const GW1 As Object = "" 'This number would be your original value

Const GW2 As Double = 9 'this number to set poly x10 "because of below #"

'l8ter we will randomly change the set# into a random key.

'child first class

'GW1

Public Class BasePoly

```

Overridable Function PolyCrypt(ByVal Addpoly As String) As Double
    Addpoly = Form1.TextBox1.Text 'Added this and changed Double to String
    Return Addpoly * GW1          'In both PolyCrypt Functions
End Function
End Class

```

'child second class

'GW2

Public Class ModPoly

Inherits BasePoly

Private BaseAddpoly As Double

Overrides Function PolyCrypt(ByVal Addpoly As String) As Double

'changed Double to String

BaseAddpoly = MyBase.PolyCrypt(Addpoly)

Return GW2 * (BaseAddpoly + Addpoly) + BaseAddpoly

End Function

End Class

Sub TestPoly()

Dim GodWall1 As New BasePoly

Dim GodWall2 As New ModPoly

ShowPoly(GodWall1, 10) 'This number is original value x10

ShowPoly(GodWall2, 10) 'This number is original value x10 + Set#

'The set is affected by this second number above

'During Decryption these numbers and set numbers

'Must stay the same so you can get the original value back

'Changing the numbers randomly before encryption will give you

'Random encryptions of the value each time

'Feel free to play around with the numbers so you get the idea

End Sub

Sub ShowPoly(ByVal GodWall As BasePoly, ByVal Addpoly As Double)

Dim polyplus As Double

polyplus = GodWall.PolyCrypt(Addpoly)

MsgBox("This is True Polymorphism " & Format(polyplus, "G"))

End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Handles Button1.Click

TextBox1.Text = GetPoly(True)

TestPoly()

End Sub

Shared Function GetPoly(ByVal gw As Integer) As String

'Added this function, will make Metamorphic later

Dim Value(9) As String

Dim Rndm As New Random()

Dim x As String = ""

Dim i As Integer

Value(1) = 100

Value(2) = 200

Value(3) = 300

Value(4) = 400

Value(5) = 500

```

Value(6) = 600
Value(7) = 700
Value(8) = 800
Value(9) = 900
For i = 1 To 1
    gw = Rndm.Next(1, Value.Length)
    x &= Value(gw) & " "
Next
Return x
End Function
End Class

```

Part 3. Set to a secure random Key ?

Add another textbox

Code 'Working Example of True Polymorphism in VB.Net

'By GodWall

'Polymorphic Source

'2010

'This is just a basic example with random value

'Notice the application of the set key in message 2

'Parent class

Imports System.Security.Cryptography 'This is New

Public Class Form1

'Const GW1 As Object = "" 'This number would be your original value

'Const GW2 As Double = 9 'this number to set poly x10 "because of below #"

'l8ter we will randomly change the set# into a random key.

'Key is secure and working

'child first class

'GW1

Public Class BasePoly

Overridable Function PolyCrypt(ByVal Addpoly As String) As Double

Addpoly = Form1.TextBox1.Text 'Added this and changed Double to String

Return Addpoly * GW1 'In both PolyCrypt Functions

End Function

End Class

'child second class

'GW2

Public Class ModPoly

Inherits BasePoly

Private BaseAddpoly As Double

Overrides Function PolyCrypt(ByVal Addpoly As String) As Double

'changed Double to String

Addpoly = Form1.TextBox2.Text 'Added this

BaseAddpoly = MyBase.PolyCrypt(Addpoly)

Return Addpoly * (BaseAddpoly + Addpoly) + BaseAddpoly 'GW2

End Function

End Class

```

Sub TestPoly()
    Dim GodWall1 As New BasePoly
    Dim GodWall2 As New ModPoly
    ShowPoly(GodWall1, 10)      'This number is original value x10
    ShowPoly(GodWall2, 10)      'This number is original value x10 + Set#
    'The set is affected by this second number above
    'During Decryption these numbers and set numbers
    'Must stay the same so you can get the original value back
    'Changing the numbers randomly before encryption will give you
    'Random encryptions of the value each time
    'Feel free to play around with the numbers so you get the idea
End Sub

Sub ShowPoly(ByVal GodWall As BasePoly, ByVal Addpoly As Double)
    Dim polyplus As Double
    polyplus = GodWall.PolyCrypt(Addpoly)
    MsgBox("This is True Polymorphism " & Format(polyplus, "G"))
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    TextBox1.Text = GetPoly(True)
    TextBox2.Text = RNGCSP() 'Now our set should be a secure Random key
    TestPoly()
End Sub

Shared Function GetPoly(ByVal gw As Integer) As String
    'Added this function, will make Metamorphic l8ter
    Dim Value(9) As String
    Dim Rndm As New Random()
    Dim x As String = ""
    Dim i As Integer
    Value(1) = 100
    Value(2) = 200
    Value(3) = 300
    Value(4) = 400
    Value(5) = 500
    Value(6) = 600
    Value(7) = 700
    Value(8) = 800
    Value(9) = 900
    For i = 1 To 1
        gw = Rndm.Next(1, Value.Length)
        x &= Value(gw) & " "
    Next
    Return x
End Function

Private Function RNGCSP() 'This is also new
    'This is much more secure than just using Random
    Dim Rndm As New RNGCryptoServiceProvider()
    Dim y As Byte() = New Byte(0) {}
    Rndm.GetBytes(y)
    Dim x As Integer = Convert.ToInt32(y(0))
    Return x
End Function

```

End Class

Part 4. My Polymorphic Number Gen ?

Add 2 checkboxes

Code 'Working Example of True Polymorphism in VB.Net

'By GodWall

'Polymorphic Source

'2010

'This is just a basic example with random value

'Notice the application of the set key in message 2

'Ok now to make my Polymorphic Numbergen

'Parent class

Imports System.Security.Cryptography 'This is New

Public Class Form1

 'Const GW1 As Object = "" 'This number would be your original value

 'Const GW2 As Double = 9 'this number to set poly x10 "because of below #"

 'l8ter we will randomly change the set# into a random key.

 'Key is secure and working

 'child first class

 'GW1

 Public Class BasePoly

 Overridable Function PolyCrypt(ByVal Addpoly As String) As Double

 Addpoly = Form1.TextBox1.Text 'Added this and changed Double to String

 Return Addpoly * GW1 'In both PolyCrypt Functions

 End Function

 End Class

 'child second class

 'GW2

 Public Class ModPoly

 Inherits BasePoly

 Private BaseAddpoly As Double

 Overrides Function PolyCrypt(ByVal Addpoly As String) As Double

 'changed Double to String

 Addpoly = Form1.TextBox2.Text 'Added this

 BaseAddpoly = MyBase.PolyCrypt(Addpoly)

 Return Addpoly * (BaseAddpoly + Addpoly) + BaseAddpoly 'GW2

 End Function

 End Class

 Sub TestPoly()

 'Dim GodWall1 As New BasePoly

 Dim GodWall2 As New ModPoly

 >ShowPoly(GodWall1, 10) 'This number is original value x10

 ShowPoly(GodWall2, 10) 'This number is original value x10 + Set#

 'The set is affected by this second number above

 'During Decryption these numbers and set numbers

 'Must stay the same so you can get the original value back

 'Changing the numbers randomly be4 encryption will give you

```

'Random encryptions of the value each time
'Feel free to play around with the numbers so you get the idea
End Sub
Sub ShowPoly(ByVal GodWall As BasePoly, ByVal Addpoly As Double)
Dim polyplus As Double
polyplus = GodWall.PolyCrypt(Addpoly)
TextBox3.Text = ("" & Format(polyplus, "G")) 'Change This
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
If CheckBox1.Checked = True Then
    TextBox1.Text = GetPoly(True)
Else
    TextBox1.Text = Nothing
End If
If CheckBox2.Checked = True Then
    TextBox2.Text = RNGCSP()
Else
    TextBox2.Text = Nothing
End If
'TextBox1.Text = GetPoly(True)
'TextBox2.Text = RNGCSP() 'Now our set should be a secure Random key
'TestPoly()
End Sub
Shared Function GetPoly(ByVal gw As Integer) As String
'Added this function, will make Metamorphic l8ter
'Maybe in my next TUT
Dim Value(9) As String
Dim Rndm As New Random()
Dim x As String = ""
Dim i As Integer
Value(1) = 100
Value(2) = 200
Value(3) = 300
Value(4) = 400
Value(5) = 500
Value(6) = 600
Value(7) = 700
Value(8) = 800
Value(9) = 900
For i = 1 To 1
    gw = Rndm.Next(1, Value.Length)
    x &= Value(gw) & " "
Next
Return x
End Function
Private Function RNGCSP() 'This is also new
'This is much more secure than just using Random
Dim Rndm As New RNGCryptoServiceProvider()
Dim y As Byte() = New Byte(0) {}
Rndm.GetBytes(y)
Dim x As Integer = Convert.ToInt32(y(0))
Return x

```

```

End Function
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
TestPoly() ' Add This
End Sub
'Final note: As long as you have the Key and (GodWall2, 10) "the 10 is not changed" it will
decrypt back to the base number
'And now HF has a public polymorphic source, your all welcome.
'GodWall
End Class
Part 5. Final Source + DownLoad  ?

```

Download uploading now

```

Code 'Working Example of True Polymorphism in VB.Net
'By GodWall
'Polymorphic Source
'2010

```

```

'Parent class
Imports System.Security.Cryptography 'This is New
Public Class Form1

'child first class
Public Class BasePoly
Overridable Function PolyCrypt(ByVal Addpoly As String) As Double
Addpoly = Form1.TextBox1.Text
Return Addpoly
End Function
End Class

'child second class
Public Class ModPoly
Inherits BasePoly
Private BaseAddpoly As Double
Overrides Function PolyCrypt(ByVal Addpoly As String) As Double
Addpoly = Form1.TextBox2.Text
BaseAddpoly = MyBase.PolyCrypt(Addpoly)
Return Addpoly * (BaseAddpoly + Addpoly) + BaseAddpoly
End Function
End Class

Sub TestPoly()
Dim GodWall2 As New ModPoly
ShowPoly(GodWall2, 10)
End Sub
Sub ShowPoly(ByVal GodWall As BasePoly, ByVal Addpoly As Double)
Dim polyplus As Double
polyplus = GodWall.PolyCrypt(Addpoly)
TextBox3.Text = (" " & Format(polyplus, "G"))
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

```

```

If CheckBox1.Checked = True Then
    TextBox1.Text = GetPoly(True)
Else
    TextBox1.Text = Nothing
End If
If CheckBox2.Checked = True Then
    TextBox2.Text = RNGCSP()
Else
    TextBox2.Text = Nothing
End If
End Sub
Shared Function GetPoly(ByVal gw As Integer) As String
Dim Value(9) As String
Dim Rndm As New Random()
Dim x As String = ""
Dim i As Integer
Value(1) = 100
Value(2) = 200
Value(3) = 300
Value(4) = 400
Value(5) = 500
Value(6) = 600
Value(7) = 700
Value(8) = 800
Value(9) = 900
For i = 1 To 1
    gw = Rndm.Next(1, Value.Length)
    x &= Value(gw) & " "
Next
Return x
End Function
Private Function RNGCSP()
Dim Rndm As New RNGCryptoServiceProvider()
Dim y As Byte() = New Byte(0) {}
Rndm.GetBytes(y)
Dim x As Integer = Convert.ToInt32(y(0))
Return x
End Function
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
    TestPoly()
End Sub
End Class

```

Hope you all enjoyed this. ?

Wow sorry post was to big heres the link:

Download

<https://#.com/files/459255346/SchoolOfPoly.zip>

I will look into it.

I think it's enough complex and TRUE polymorphism.