Classes & Objects

Visual Basic .NET

By: Sayed Hassan Adelyar

Classes & Objects

Classes & Objects

- کلاسها **چوکات و قالب** آبجکتها است. آبجکتها داراي **خصوصيات** (Properties) اند که **طرز کار** آنها را تعين ميکند. در ويژول بسيک دات نت براي **ايجاد آبجکتها** در قدم اول **کلاس** همان آبجکت را ايجاد کنيد. در کلاس **خصوصيات** و **ميتودهاي** آبجکتها را درج کنيد.

Classes & Objects

```
Public Class person
    Private firstname As String
    Private lastname As String
    Private birthdate As Date
    Function completename() As String
        firstname = "Zobair"
        lastname = "Noor"
        completename = firstname & " , " & lastname
    End Function
End Class
```

Classes & Objects

```
Public Class Form1
Private Sub Button1_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles Button1.Click
        Dim newperson As New person
        TextBox1.Text = newperson.completename
End Sub
End Class
```

Classes & Objects

- در مثال ذیل متحولین را به شکل Public تعریف شده و میتواند مستقیما توسط آبجکتها استفاده شود.

```
Public Class person
    Public firstname As String
    Public lastname As String
    Public birthdate As Date
    Function completename() As String
        firstname = "Zobair"
        lastname = "Noor"
        completename = firstname & " , " & lastname
    End Function
End Class
```

Classes & Objects

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
        Dim newperson As New person
        TextBox1.Text = newperson.completename
        TextBox2.Text = newperson.firstname
        TextBox3.Text = newperson.lastname
        TextBox4.Text = newperson.birthdate
    End Sub
End Class
```

Constructor

- **Constructor جهت** دادن قیمت اولیه **(Initialization)** متحولین و آبجکتها استفاده میشود.
- هر وقت که یک آبجکت توسط کلمه **New** ایجاد میشود، Constructor به طور **اتومات** اجرا میشود وآبجکت ایجاد شده را قیمت اولیه میدهد.
- در مثال ذیل از Constructor استفاده گردیده است:

Classes & Objects

```
Public Class person
    Private firstname As String
    Private lastname As String
    Public birthdate As Date
    Sub New(ByVal first As String, ByVal last As String, ByVal birth As
     Date)
        firstname = first
        lastname = last
        birthdate = birth
    End Sub
    Function completename() As String
        completename = firstname & " , " & lastname
    End Function
End Class
```

Classes & Objects

```
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles Button1.Click
        Dim newperson As New person("Zobair", "Noor", "2000-
    3-6")
        TextBox1.Text = newperson.completename
        TextBox2.Text = newperson.birthdate
        TextBox3.Text = Year(Now()) -
    Year(newperson.birthdate)
End Sub
End Class
```

Properties

Classes & Objects

روش دیگر که توسط آن قیمت ها را گرفته میتوانیم عبارت از استفاده از خصوصیات میباشد.

```
Public Class rectangle
    Private m_width, m_length, area As Integer
    Public Property width() As Integer
        Get
            Return m_width
        End Get
        Set(ByVal value As Integer)
            m_width = value
        End Set
    End Property
    Public Property length() As Integer
        Get
            Return m_length
        End Get
        Set(ByVal value As Integer)
            m_length = value
        End Set
    End Property
```

```
Public Class Form1
    Private Sub Form1_Load(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
MyBase.Load
        Dim rec1 As rectangle = New
rectangle()
        Dim rec2 As rectangle = New
rectangle()
        rec1.width = 18
        rec1.length = 30
        TextBox1.Text = rec1.width
        TextBox2.Text = rec1.length
    End Sub
End Class
```

Properties

Classes & Objects

روش دیگر که توسط آن قیمت ها را گرفته میتوانیم عبارت از استفاده از خصوصیات میباشد.

```
Public Class rectangle
    Private m_width, m_length, area As Integer
    Public Property width() As Integer
        Get
            Return m_width
        End Get
        Set(ByVal value As Integer)
            m_width = value
        End Set
    End Property
    Public Property length() As Integer
        Get
            Return m_length
        End Get
        Set(ByVal value As Integer)
            m_length = value
        End Set
    End Property
```

```
Public Sub New(ByVal width
As Integer, ByVal length As
Integer)
        m_width = width
        m_length = length
        area = m_width *
m_length
    End Sub
End Class
```

Classes & Objects

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        Dim rec1 As rectangle = New rectangle(12, 25)
        Dim rec2 As rectangle = New rectangle(18, 30)
        TextBox1.Text = rec1.width
        TextBox2.Text = rec1.length
    End Sub
End Class
```

Properties

Classes & Objects

روش دیگر که توسط آن قیمت ها را گرفته میتوانیم عبارت از استفاده از خصوصیات میباشد.

```
Public Class rectangle
    Private m_width, m_length, area As Integer
    Public Property width() As Integer
        Get
            Return m_width
        End Get
        Set(ByVal value As Integer)
            m_width = value
        End Set
    End Property
    Public Property length() As Integer
        Get
            Return m_length
        End Get
        Set(ByVal value As Integer)
            m_length = value
        End Set
    End Property
```

```
Public Sub New(ByVal width
As Integer, ByVal length As
Integer)
        m_width = width
        m_length = length
        area = m_width *
m_length
    End Sub
End Class
```

Classes & Objects

```
Public Class Form1
Private Sub Form1_Load(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        Dim rec1 As rectangle = New rectangle(12, 20)
        Dim rec2 As rectangle = New rectangle(15, 25)
        Dim m_library() As rectangle = New rectangle()
    {rec1, rec2}
        listoffigures.DataSource = m_library
End Sub
```

Classes & Objects

```
Private Sub listoffigures_SelectedIndexChanged(ByVal
    sender As System.Object, ByVal e As System.EventArgs)
    Handles listoffigures.SelectedIndexChanged
        Dim therec As rectangle =
    CType(listoffigures.SelectedItem, rectangle)
        titlelabel1.Text = therec.width
        titlelabel2.Text = therec.length
        Label1.Text = therec.area
    End Sub
End Class
```

Classes & Objects

```
Public Class Time
    Private mHour As Integer
    Private mMinute As Integer
    Private mSecond As Integer
    Public Sub New(ByVal h As
     Integer, ByVal m As
     Integer, ByVal s As Integer)
        mHour = h
        mMinute = m
        mSecond = s
    End Sub
```

```
Public Sub SetTime(ByVal hourValue As Integer, _
      ByVal minuteValue As Integer, ByVal secondValue As
Integer)
        If (hourValue >= 0 AndAlso hourValue < 24) Then
            mHour = hourValue
        Else
            mHour = 0
        End If
        If (minuteValue >= 0 AndAlso minuteValue < 60) Then
            mMinute = minuteValue
        Else
            mMinute = 0
        End If
        If (secondValue >= 0 AndAlso secondValue < 60) Then
            mSecond = secondValue
        Else
            mSecond = 0
        End If
    End Sub
```

Classes & Objects

```
Public Function StandardTime() As String
        Dim suffix As String = " PM"
        Dim format As String = "{0}:{1:D2}:{2:D2}"
        Dim standardHour As Integer
        SetTime(mHour, mMinute, mSecond)
        If mHour < 12 Then
            suffix = " AM"
        End If
        If (mHour = 12 OrElse mHour = 0) Then
            standardHour = 12
        Else
            standardHour = mHour Mod 12
        End If
        Return String.Format(format, standardHour,
      mMinute, _
          mSecond) & suffix
    End Function
End Class
```

```
Public Function
UniversalTime() As String
        SetTime(mHour,
mMinute, mSecond)
        Return
String.Format("{0}:{1:D2}:{2
:D2}", _
            mHour, mMinute,
mSecond)
    End Function
```

Classes & Objects

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
        Dim newtime As New Time(Val(TextBox1.Text),
    Val(TextBox2.Text), Val(TextBox3.Text))
        TextBox4.Text = newtime.UniversalTime()
        TextBox5.Text = newtime.StandardTime()
    End Sub
End Class
```

Classes & Objects

- مثال: مثال ذيل روش ايجاد يک کلاس براي حساب بانکي (Bank Account) را نشان ميدهد. کلاس آن داراي اعضاي (Members) ذيل ميباشد، Owner، ID، Balance، Deposit، Withdraw، Constructor

Classes & Objects

```
Public Class BankAccount
    Private m_owner As String
    Public ReadOnly Property ID() As String
        Get
            Return m_owner
        End Get
    End Property
    Private m_balance As Decimal
    Public ReadOnly Property balance() As
      Decimal
        Get
            Return m_balance
        End Get
    End Property
    Public Sub New(ByVal owner As String)
        m_owner = owner
        m_balance = 0D
    End Sub
```

```
    Public Function deposit(ByVal
amount As Decimal) As Decimal
        m_balance += amount
        Return m_balance
    End Function
    Public Function
withdraw(ByVal amount As
Decimal) As Decimal
        m_balance -= amount
        Return m_balance
    End Function
End Class
```

Classes & Objects

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load

        Dim account As BankAccount = New
    BankAccount("Faisal")

        account.deposit(25D)

        MessageBox.Show(String.Format("{0:C}",
    account.balance))

    End Sub

End Class
```

Classes & Objects

```
Public Class SavingAccount
  Inherits BankAccount
  Public Overrides ReadOnly Property ID() As String
    Get
      Return Me.m_owner & "-S"
    End Get
  End Property
  Public Sub New(ByVal owner As String)
    MyBase.New(owner)
  End Sub
  Private m_interest As Decimal = 0.01D
  Public Property interest() As Decimal
    Get
      Return m_interest
    End Get
    Set(ByVal value As Decimal)
      m_interest = value
    End Set
  End Property
```

```
Public Function
additerest() As Decimal

Me.deposit(m_interest *
Me.balance)
      Return Me.balance
  End Function
End Class
```

Classes & Objects

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load
        Dim savings As SavingAccount = New
    SavingAccount("Adelyar")
        savings.deposit(150D)
        savings.withdraw(50D)
        savings.interest = 0.05D
        savings.additerest()
        MessageBox.Show(String.Format("{0}: {1:C}",
    savings.ID, savings.balance))
    End Sub
End Class
```

Classes & Objects

- اکنون میخواهیم کلاس دیگر بنام **CheckingAccount** را ایجاد نمائیم. این کلاس مانند BankAccount بوده با استثناي اینکه هر **CheckingAccount** از **Withdrawal** به مقدار **0.25** دالر مصرف دارد.

Classes & Objects

```
Public Class BankAccount
    Protected m_owner As String
    Public Overridable ReadOnly Property ID() As
      String
        Get
            Return m_owner
        End Get
    End Property
    Private m_balance As Decimal
    Public ReadOnly Property balance() As Decimal
        Get
            Return m_balance
        End Get
    End Property
    Public Sub New(ByVal owner As String)
        m_owner = owner
        m_balance = 0D
    End Sub
```

```
    Public Function deposit(ByVal
amount As Decimal) As
Decimal
            m_balance += amount
            Return m_balance
        End Function
        Public Overridable Function
withdraw(ByVal amount As
Decimal) As Decimal
            m_balance -= amount
            Return m_balance
        End Function
End Class
```

Classes & Objects

```
Public Class CheckingAccount
    Inherits BankAccount
    Public Sub New(ByVal owner As String)
        MyBase.New(owner)
    End Sub
    Public Overrides Function withdraw(ByVal amount As Decimal) As Decimal
        MyBase.withdraw(amount)
        MyBase.withdraw(0.25D)
        Return Me.balance
    End Function
    Public Overrides ReadOnly Property ID() As String
        Get
            Return Me.m_owner & "-C"
        End Get
    End Property
End Class
```

Classes & Objects

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles MyBase.Load
        Dim Checking As CheckingAccount = New
     CheckingAccount("Adelyar")
        Checking.deposit(50D)
        Checking.withdraw(5D)
        MessageBox.Show(String.Format("{0}: {1:C}", Checking.ID,
     Checking.balance))
    End Sub
End Class
```

Inherit from a Control

Classes & Objects

- You can inherit from a class which exist in the .NET Framwork. That is you can inherit from a control.
- For example, to create a round button we can inherit from the Button.
- Create a new project.

```
Public Class RoundButton
    Inherits Button
    Protected Overrides Sub OnPaint(ByVal pevent As
     System.Windows.Forms.PaintEventArgs)
        Me.Size = New Size(50, 50)
        Dim aCircle As System.Drawing.Drawing2D.GraphicsPath = New
     System.Drawing.Drawing2D.GraphicsPath()
        aCircle.AddEllipse(New System.Drawing.RectangleF(0, 0, 50, 50))
        Me.Region = New Region(aCircle)
    End Sub

End Class
```

Classes & Objects

```
Public Class newtextbox
    Inherits TextBox
    Protected Overrides Sub OnPaint(ByVal pevent As
System.Windows.Forms.PaintEventArgs)
        Me.Size = New Size(150, 150)
        Dim arectangle As
System.Drawing.Drawing2D.GraphicsPath = New
System.Drawing.Drawing2D.GraphicsPath()
        arectangle.AddEllipse(New
System.Drawing.RectangleF(100, 100, 50, 50))
        Me.Region = New Region(arectangle)
    End Sub
End Class
```

Classes & Objects

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load
        Dim rb As New RoundButton()
        Me.Controls.Add(rb)
        AddHandler rb.Click, AddressOf RoundButton_click
    End Sub
    Private Sub RoundButton_click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
        MessageBox.Show("Hello")
    End Sub
End Class
```

Classes & Objects

- For this purpose we override the **OnPaint** method of the Button control. The **OnPaint** method is **called each time** the **control** is **drawn** on the form. By **overriding** the OnPaint method, you can **determine** the **appearance** of the button.
- To **make** a **control** assume a **particular shape**, in this case **round**, you must define its **Region property** so that it achieves the shape. You can create a shape using the **GraphicsPath** object. The **GraphicsPath** object allow you to **create** a **shape** by drawing. In this example, you create a **drawing** by adding a **circle** to **GraphicsPath**.

Responding to Changes with Events and Exceptions

Classes & Objects

- **MS-Windows interface** is **event driven**.
- In this **section** we will **create** a control that **appear** in the **toolbox**.
- The control will have **events** that you can choose to **respond** to or **ignore** in your code.

Classes & Objects

- **Project description**
  - A **train runs** a long a **track** across the **screen**. At **regular intervals** but **random locations**, the track **catches fire**. The old **fire goes out** when the **new fire appears** so that there's always one fire on the **track** at any point in time.
  - You can **adjust** the **speed** of the of the **train** using a **slider control**.
  - The **object** of the **game** is to **get** the **train** to the end of the **track without running** into a **fire**.

Classes & Objects

- **Classes**:
  - **Track**
  - **Train**
  - **Fire**
  - **Form**
- To **describe** how or when the train **moves** or when the **fire** will appear and where, we need **events**, **signals** from **one object** to another that **something** has **happened**.
- We need **these events**:

Classes & Objects

- A **caughtOnFire** event for the **track class**. This **event generated** by the **track**, will be **received** by the **form** so that the code in the form can move the fire on the **track**. The **frequency property** will be moved to the **track class** to **indicate** how often the track should **raise** a **caughtOnFire** event.
- A **DistanceChanged event** for the **Train class**. This **event** will be **generated periodically** to let the form **know** where the **train** is on the **track**. The **location** of the train **depends** on the **speed** of the **train** and how **long** it has been **running**.

Classes & Objects

- Using the **CaughtOnFire** and **DistanceChanged events**, the **form code** can **coordinate** the **behavior** of the **track**, the **train**, and the **fire**.
- The **event** can carry **information** in **parameters**. In the **case** of the **CaughtOnFire**, the **event carries information** about the **location** of the **fire**.

Classes & Objects

- The **Track class**:
  - **Project, Add User Control**
  - **Add** the **following code**:

Classes & Objects

```
<System.ComponentModel.DefaultEvent("CaughtOnFire")> Public Class Track
    Inherits System.Windows.Forms.UserControl
    Private m_fireFrequency As Integer = 1
    Public Property FireFrequency() As Integer
        Get
            Return m_fireFrequency
        End Get
        Set(ByVal Value As Integer)
            If Value >= 1 Then
                m_fireFrequency = Value
                Timer1.Interval = m_fireFrequency * 1000
            End If
        End Set
    End Property
    Private Const TrackHeight As Integer = 15
    Private Const BarWidth As Integer = TrackHeight \ 5
    Private Const BarSpacing As Integer = BarWidth * 2
```

Classes & Objects

```
Private Sub Track_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
    Me.Height = 15
End Sub

Protected Overrides Sub OnSizeChanged(ByVal e As
System.EventArgs)
    Me.Height = TrackHeight
    Dim nBars As Integer = Me.Width \ BarSpacing
    Me.Width = nBars * BarSpacing
End Sub
```

Classes & Objects

```
Protected Overrides Sub OnPaint(ByVal e As System.Windows.Forms.PaintEventArgs)
    MyBase.OnPaint(e)
    Dim gp As New System.Drawing.Drawing2D.GraphicsPath()
    gp.FillMode = Drawing.Drawing2D.FillMode.Winding
    Dim height As Integer = TrackHeight \ 5
    Dim nBars As Integer = Me.Width \ BarSpacing
    Dim bar As Integer
    For bar = 0 To nBars - 1
        gp.AddRectangle(New System.Drawing.Rectangle(bar * BarSpacing, height,
  BarSpacing, height))
        gp.AddRectangle(New System.Drawing.Rectangle(bar * BarSpacing, height * 3,
  BarSpacing, height))
        gp.AddRectangle(New System.Drawing.Rectangle(bar * BarSpacing, 0, BarWidth,
  TrackHeight))
    Next
    e.Graphics.FillPath(System.Drawing.Brushes.SaddleBrown, gp)
End Sub
```

Classes & Objects

```
Public Event CaughtOnFire(ByVal sender As Object,
    ByVal e As CaughtOnFireEventsArgs)

    Private Sub Timer1_Tick(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Timer1.Tick

        Dim randomNumber As New System.Random()

        RaiseEvent CaughtOnFire(Me, New
CaughtOnFireEventsArgs(randomNumber.Next(0,
Me.Width)))

    End Sub

End Class
```

Classes & Objects

```
Public Class CaughtOnFireEventsArgs
    Inherits System.EventArgs

    Private m_location As Integer = 0
    Public ReadOnly Property Location() As Integer
        Get
            Return m_location
        End Get
    End Property

    Public Sub New(ByVal location As Integer)
        m_location = location
    End Sub
End Class
```

Classes & Objects

```
<System.ComponentModel.DefaultEvent("DistanceChanged")> Public
    Class Train
    Inherits System.Windows.Forms.UserControl
    Private m_speed As Integer = 0
    Public Property Speed() As Integer
        Get
            Return m_speed
        End Get
        Set(ByVal Value As Integer)
            If Value >= 0 Then
                m_speed = Value
            End If
        End Set
    End Property
```

Classes & Objects

```
Private m_distance As Integer = 0
    Public ReadOnly Property Distance() As Integer
        Get
            Return m_distance
        End Get
    End Property

    Public Sub ReStart()
        m_distance = 0
    End Sub
    Public Event DistanceChanged(ByVal sender As Object, _
        ByVal e As DistanceChangedEventArgs)
```

Classes & Objects

```
Private Sub Timer1_Tick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Timer1.Tick
    If m_speed > 0 Then
        m_distance +=
Convert.ToInt32(Convert.ToInt32(m_speed) _
            * (Convert.ToDouble(Timer1.Interval) / 1000.0F))
        RaiseEvent DistanceChanged(Me, _
            New DistanceChangedEventArgs(m_distance))
    End If
End Sub
```

Classes & Objects

```
Public Class DistanceChangedEventArgs
    Inherits System.EventArgs
    Private m_distance As Integer
    Public ReadOnly Property Distance() As Integer
        Get
            Return m_distance
        End Get
    End Property
    Public Sub New(ByVal distance As Integer)
        m_distance = distance
    End Sub
  End Class
End Class
```

Classes & Objects

```
Public Class Form1
    Private Sub Track1_CaughtOnFire(ByVal sender As System.Object, ByVal e As
     TrainGame.CaughtOnFireEventsArgs) Handles Track1.CaughtOnFire
        fire.Location = New System.Drawing.Point(Track1.Left + e.Location,
     Track1.Top - fire.Height)
    End Sub

    Private Sub Train1_DistanceChanged(ByVal sender As System.Object, ByVal e As
     TrainGame.Train.DistanceChangedEventArgs) Handles Train1.DistanceChanged
        Train1.Left = Track1.Left + e.Distance
        If Train1.Right >= Track1.Right Then
            Train1.Speed = 0
            Throttle.Value = 0
        End If
    End Sub
```

Classes & Objects

```
Private Sub Throttle_ValueChanged(ByVal sender As Object, ByVal e As
    System.EventArgs) Handles Throttle.ValueChanged
      If Train1.Right < Track1.Right Then
         Train1.Speed = Throttle.Value
      Else
         Throttle.Value = 0
      End If
 End Sub
Private Sub Reset_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Reset.Click
      Train1.ReStart()
      Throttle.Value = 0
      Train1.Speed = 0
      Train1.Left = Track1.Left
   End Sub
End Class
```

Classes & Objects