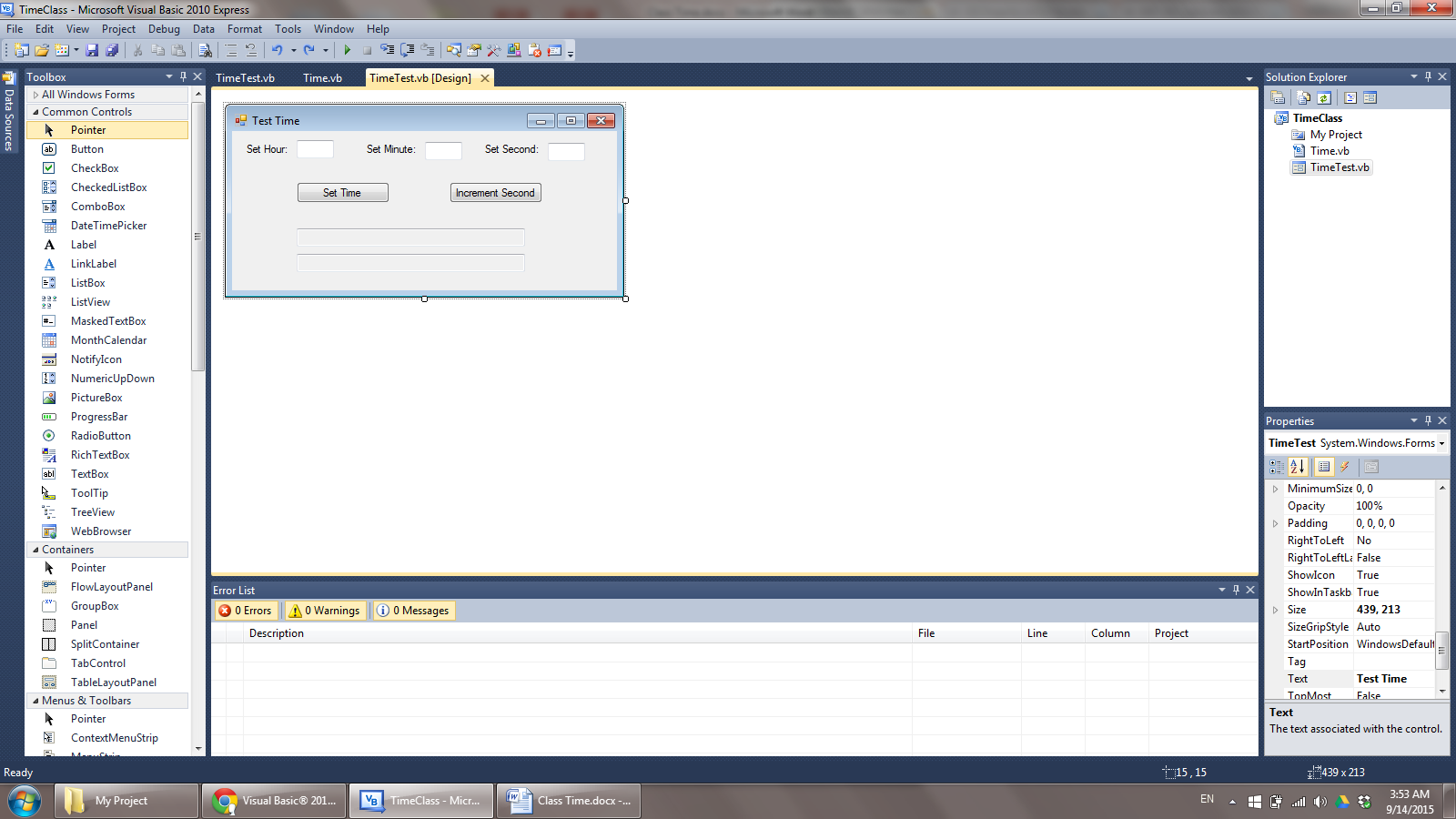
CT1411

Example2:

* classes Time and TimeTest.
* The Time class represents properties Hour , Minute and Second to control access to instance variables hourValue, minuteValue and secondValue, respectively. Each of these properties contains a Get  and a Set .
* The TimeTest class creates and uses an object of class Account. You can set the hour, minute and second using the three TextBoxes. If you specify an invalid value, an exception occurs. The program catches the exception and displays an appropriate message in a MessageBox.
* You can also press the **Increment Second** Button to add one second to the time.
* When you update the time, the program displays the new values of the hour, minute and second in txt1 and displays the standard time in the txt2.



Public Class Time

' declare Integer instance variables for the hour, minute and second

Private hourValue As Integer ' 0 - 23

Private minuteValue As Integer ' 0 - 59

Private secondValue As Integer ' 0 - 59

' Time constructor with hour, minute and second as optional parameters

Public Sub New(Optional ByVal h As Integer = 12,

Optional ByVal m As Integer = 0, Optional ByVal s As Integer = 0)

SetTime(h, m, s) ' call SetTime with three arguments

End Sub ' New

' set a new time value using universal time, check validity of the data

Public Sub SetTime(ByVal h As Integer, ByVal m As Integer,

ByVal s As Integer)

Hour = h ' Set accessor validates the hour

Minute = m ' Set accessor validates the minute

Second = s ' Set accessor validates the second

End Sub ' SetTime

' property Hour

Public Property Hour() As Integer

Get ' return hourValue

Return hourValue

End Get

Set(ByVal value As Integer) ' set hourValue

If (value >= 0 AndAlso value < 24) Then ' in range 0-23?

hourValue = value ' value is valid

Else ' invalid hour

Throw New ArgumentOutOfRangeException("hour must be 0-23")

End If

End Set

End Property ' Hour

' property Minute

Public Property Minute() As Integer

Get ' return minuteValue

Return minuteValue

End Get

Set(ByVal value As Integer) ' set minuteValue

If (value >= 0 AndAlso value < 60) Then ' in range 0-59?

minuteValue = value ' value is valid

Else ' invalid minute

Throw New ArgumentOutOfRangeException("minute must be 0-59")

End If

End Set

End Property ' Minute

' property Second

Public Property Second() As Integer

Get ' return secondValue

Return secondValue

End Get

Set(ByVal value As Integer) ' set secondValue

If (value >= 0 AndAlso value < 60) Then ' in range 0-59?

secondValue = value ' value is valid

Else ' invalid second

Throw New ArgumentOutOfRangeException("second must be 0-59")

End If

End Set

End Property ' Second

' return Time as a String in standard-time (12-hour clock) format

Public Overrides Function ToString() As String

Dim suffix As String ' AM or PM suffix

Dim standardHour As Integer ' a standard hour in the range 1-12

' determine whether the 12-hour clock suffix should be AM or PM

If Hour() < 12 Then

suffix = "AM"

Else

suffix = "PM"

End If

' convert hour from universal-time format to standard-time format

If (Hour = 12 OrElse Hour = 0) Then

standardHour = 12 ' noon or midnight

Else

standardHour = Hour() Mod 12 ' 1 through 11, AM or PM

End If

Return String.Format("{0}:{1:D2}:{2:D2} {3}",

standardHour, Minute, Second, suffix)

End Function ' ToString

End Class

Public Class TimeTest

' Accessing data via properties.

Dim time As New Time() ' construct Time with zero arguments

' update time display

Private Sub UpdateDisplay()

txtH.Text = Convert.ToString(time.Hour)

txtM.Text = Convert.ToString(time.Minute)

txtS.Text = Convert.ToString(time.Second)

txt1.Text = ("Hour: " & time.Hour & "; Minute: " &

time.Minute & "; Second: " & time.Second)

txt2.Text = ("Standard time is: " & time.ToString() )

End Sub ' UpdateDisplay

' invoked when user clicks the Add 1 to Second button

Private Sub btnSec\_Click(sender As System.Object, e As System.EventArgs) Handles btnSec.Click

time.Second = (time.Second + 1) Mod 60 ' add 1 to Second

' add one minute if 60 seconds have passed

If time.Second = 0 Then

time.Minute = (time.Minute + 1) Mod 60 ' add 1 to Minute

' add one hour if 60 minutes have passed

If time.Minute = 0 Then

time.Hour = (time.Hour + 1) Mod 24 ' add 1 to Hour

End If

End If

UpdateDisplay() ' update the TextBoxes and output Labels

End Sub

' set time based on TextBox values

Private Sub btnSet\_Click(sender As System.Object, e As System.EventArgs) Handles btnSet.Click

' ensure that hour, minute and second are in range

Try

If txtH.Text <> String.Empty Then

time.Hour = Convert.ToInt32(txtH.Text)

End If

If txtM.Text <> String.Empty Then

time.Minute = Convert.ToInt32(txtM.Text)

End If

If txtS.Text <> String.Empty Then

time.Second = Convert.ToInt32(txtS.Text)

End If

Catch ex As ArgumentOutOfRangeException

MessageBox.Show("The hour, minute or second was out of range",

"Out of Range", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

UpdateDisplay() ' update the TextBoxes and output Labels

End Sub

End Class