10

Mãos na massa

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Vá para o projeto **ByteBank.SistemaAgencia** e adicione um novo formulário, com as seguintes propriedades e valores:

Propriedade	Valor
Name	Frm_Login
Text	Janela Principal
StartPosition	CenterScreen
WindowState	Normal
Icon	Bitcoin64x64.ico

2) No novo formulário, adicione um *PictureBox*, com as seguintes propriedades:

Propriedade	Valor
Name	Pic_Logo
Image	Bitcoin64x64.ico

3) Adicione também um *GroupBox*, de `Name Grp_Login` . Dentro do *GroupBox*, adicione dois *Label*, de `Name Lbl_Login` e `Lbl_Senha` , respectivamente, dois *TextBox*, de `Name Txt_Login` e `Txt_Senha` (esse último, com a propriedade `PasswordChar` com o valor `*` (asterisco)), respectivamente, dois *RadioButton*, de `Name Rb_Gerente` e `Rb_Representante` , respectivamente, e um *Button*, de `Name Btn_Logar` .

4) No código-fonte do formulário, adicione o `Imports` das classes:

```
Imports ByteBank.Bibliotecas.Classes.Externos
Imports ByteBank.Bibliotecas.Classes.Funcionarios
```

5) Adicione também duas variáveis `String` , para representar um gerente e um representante:

```
Dim vNomeRepresentante As String = "Representante X"
Dim vNomeGerente As String = "Gerente Y"
```

6) Dentro da sub-rotina `New()` , inicialize as propriedades `Text` dos componentes:

```
Me.Text = "Login Sistema Interno"
Grp_Login.Text = "Login"
Lbl_Login.Text = "Login"
Lbl_Senha.Text = "Senha"
```

```
Rb_Gerente.Text = "Gerente"
Rb_Representante.Text = "Representante"
Btn_Logar.Text = "Logar"
Rb_Gerente.Checked = True
Rb_Representante.Checked = False
Txt_Login.ReadOnly = True
```

7) Dê um duplo clique no `Rb_Gerente` e no seu código, caso ele esteja selecionado, imprima o seu nome no `Txt_Login` :

```
Private Sub Rd_Gerente_CheckedChanged(sender As Object, e As EventArgs) Handles Rb_Gerente.CheckedCh
    If Rb_Gerente.Checked Then
        Txt_Login.Text = vNomeGerente
    End If
End Sub
```

8) Dê um duplo clique no `Rb_Representante` e no seu código, caso ele esteja selecionado, imprima o seu nome no `Txt_Login` :

```
Private Sub Rb_Representante_CheckedChanged(sender As Object, e As EventArgs) Handles Rb_Representar
    If Rb_Representante.Checked Then
        Txt_Login.Text = vNomeRepresentante
    End If
End Sub
```

9) No código do botão `Btn_Logar_Click` , instancie um `Representante` e um `Gerente` , juntamente com as suas senhas. Em seguida, crie uma variáve booleana, com o valor `False` . Se o `Rb_Gerente` estiver selecionado, autentique o gerente e guarde o retorno na variável criada anteriormente. Se ele não estiver selecionado, autentique o representante e guarde o retorno na mesma variável. Por fim, se a variável booleanda for `True` , imprima uma mensagem de sucesso, se não, imprima uma mensagem de falha:

```
Private Sub Btn_Logar_Click(sender As Object, e As EventArgs) Handles Btn_Logar.Click

    Dim Representante As New Representante()
    Representante.senha = "rrr"

    Dim Gerente As New Gerente(11111111)
    Gerente.senha = "ggg"

    Dim vRetorno As Boolean = False

    If Rb_Gerente.Checked Then
        vRetorno = Gerente.Autenticar(Txt_Senha.Text)
    Else
        vRetorno = Representante.Autenticar(Txt_Senha.Text)
    End If

    If vRetorno = True Then
        MsgBox("Usuário efetuou o login no sistema interno.", MsgBoxStyle.Information)
    Else
        MsgBox("Usuário não tem autorização a entrar no sistema interno.", MsgBoxStyle.Critical)
    End If

End Sub
```

```
End Sub
```

10) No formulário principal, inclua mais um item no Menu, **Aula 02**, com o subitem **Vídeo 02:**

- Aula 02
  - Vídeo 02

11) No código do clique do subitem **Vídeo 02**, abra o formulário `Frm_Login` :

```
Private Sub Vídeo02ToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles Vídeo02ToolStri
    Dim F As New Frm_Login
    F.MdiParent = Me
    F.Show()
End Sub
```

12) Vá para o projeto **ByteBank.Bibliotecas**. No **Gerenciador de Soluções**, adicione uma nova classe, clicando com o botão direito do mouse sobre **Classes/Bibliotecas** e selecionando **Adicionar --> Classe**. Dê o nome da classe de `AutenticacaoHelper` :

```
Namespace Classes.Bibliotecas

    Friend Class AutenticacaoHelper

        Public Function Autenticar(senhaTentativa As String, senhaUsuario As String) As Boolean

            If senhaUsuario = senhaTentativa Then
                Return True
            End If

            Return False

            'Dim X As New ModificadoresTeste
            'X.MetodoPublico()
            'X.MetodoPrivado()
            'X.MetodoProtegico()
            'X.MetodoInterno()

        End Function


End Class

End Namespace
```

13) Modifique o código da classe `Representante` , para que ela passe a instanciar a classe `AutenticacaoHelper` . Para isto, inclua no código uma nova propriedade `autenticacao` , que instancia `AutenticacaoHelper` . E dentro do método `Autenticar` , chame e retorne o método `Autenticar` da propriedade `autenticacao` :

```
Namespace Classes.Externos

    Public Class Representante
        Implements IAutenticavel
```

```
    #Region "PROPRIEDADES"

        Public Property senha As String
        Private autenticacao As New AutenticacaoHelper()

    #End Region

    #Region "MÉTODOS"

        Public Function Autenticar(senhaTentativa As String) As Boolean
                Implements IAutenticavel.Autenticar
            Return autenticacao.Autenticar(senhaTentativa, senha)
        End Function

    #End Region

    End Class

End Namespace
```

14) Repita o mesmo procedimento para a `FuncionarioAutenticavel`, conforme abaixo:

```
Namespace Classes.Funcionarios

    Public MustInherit Class FuncionarioAutenticavel
        Inherits Funcionario
        Implements IAutenticavel

    #Region "PROPRIEDADES"

        Public Property senha As String
        Private autenticacao As New AutenticacaoHelper()

    #End Region

    #Region "CONSTRUTORES"

        Public Sub New(_cpf As String, _salario As Double)

            MyBase.New(_cpf, _salario)
        End Sub

    #End Region

    #Region "MÉTODOS"

        Public Function Autenticar(senhaTentativa As String) As Boolean
                Implements IAutenticavel.Autenticar
            Return autenticacao.Autenticar(senhaTentativa, senha)
        End Function

    #End Region

    End Class

End Namespace
```

15) No diterório **Classes/Bibliotecas**, adicione mais uma classe, chamada `Teste` :

```
Public Class ClasseTeste

    Sub Teste()

        'Dim X As New ModificadoresTeste
        'X.MetodoPublico()
        'X.MetodoPrivado()
        'X.MetodoProtegico()

    End Sub

End Class
```

16) Crie uma segunda classe, chamada `ModificadoresTeste` , com quatro métodos, cada um declarado com cada uma das quatro maneiras possíveis (`Public` , `Private` , `Protected` e `Friend`):

```
Public Class ModificadoresTeste

    Sub Teste()

        Dim X As New ModificadoresTeste
        X.MetodoPublico()
        X.MetodoPrivado()
        X.MetodoProtegico()
        X.MetodoInterno()

    End Sub

    Public Sub MetodoPublico()
        ' Método acessado pela própria classe, pela classe do mesmo projeto
        ' e por classes de outros projetos
    End Sub

    Private Sub MetodoPrivado()
        ' Método acessado pela própria classe, não é acessado pela classe do mesmo projeto
        ' e não é acessado por classes de outros projetos.
    End Sub


    Protected Sub MetodoProtegico()
        ' Método acessado pela própria classe, não é acessado pela classe do mesmo projeto
        ' e não é acessado por classes de outros projetos. Mas pode ser acessado pelas
        ' classes derivadas.
    End Sub

    Friend Sub MetodoInterno()
        ' Método acessado pela própria classe, acessado pela classe do mesmo projeto
        ' e não é acessado por classes de outros projetos.
    End Sub

End Class
```

17) Insira uma última classe, derivada da classe `ModificadoresTeste`, que será utilizada para demonstrar a diferença entre a declaração `Private` e `Protected`:

```
Public Class ClassDerivada
    Inherits ModificadoresTeste

    Sub Teste2()
        MetodoProtegico()
    End Sub

End Class
```

18) Vá para o projeto **ByteBank.SistemaAgencias** e crie uma nova pasta, chamada **Classes**. Dentro dessa pasta, crie a classe de `Estagiário`, que é um `Funcionario`, com o seguinte código:

```
Imports ByteBank.Bibliotecas.Classes.Funcionarios

Namespace Classes

    Public Class Estagiario
        Inherits Funcionario

#Region "PROPRIEDADES"

#End Region

#Region "CONSTRUTORES"

        Public Sub New(_cpf As String)
            MyBase.New(_cpf, 2000)
        End Sub

#End Region

#Region "MÉTODOS"

        Protected Overrides Function GetBonificacao() As Double
            Return (salario * 0.2)
        End Function

        Public Overrides Sub AumentarSalario()
            salario = salario * 1.1
        End Sub

#End Region

    End Class

End Namespace
```

19) Vá para o projeto **ByteBank.Bibliotecas** e modifique a declaração do método `GetBonificacao()` da classe `Funcionario` e de todas as classes derivadas (`Gerente`, `Designer`, `Desenvolvedor` e `Auxiliar`), para `Protected Friend`, conforme abaixo:

```
Protected Friend MustOverride Function GetBonificacao() As Double
```

```
    Protected Friend MustOverride Function GetBonificacao() As Double
```