08

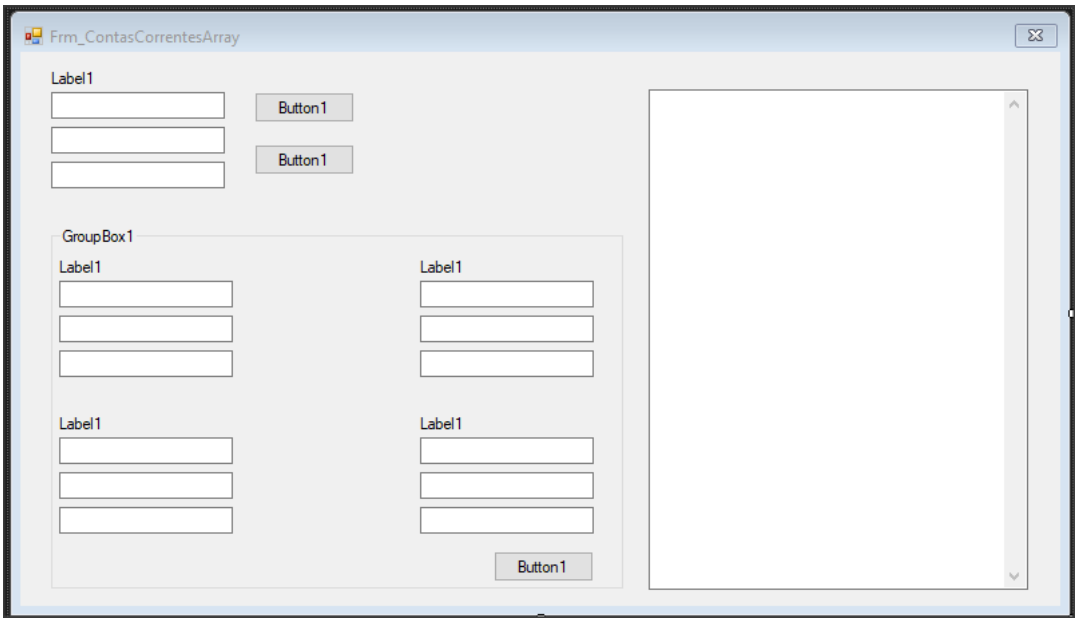
Mãos na massa

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

1) Comece baixando a solução **ByteBank** aqui (https://caelum-online-public.s3.amazonaws.com/1025-vb-net-listas-dicionarios-conjuntos/01/ByteBank.zip). Descompacte o ZIP e abra o arquivo **ByteBank.sln**.

2) No modo *Designer* do formulário `Frm_ContasCorrentesListas`, remova os *Label* `Lbl_Conta`, `Lbl_Conta1`, `Lbl_Conta2`, `Lbl_Conta3` e `Lbl_Conta4`.

3) Adicione cinco *TextBox*, abaixo de cada par de *TextBox* já existente, com `Name` `Txt_Nome`, `Txt_Nome1`, `Txt_Nome2`, `Txt_Nome3` e `Txt_Nome4`. O formulário deve ficar com o seguinte layout:



4) O objetivo agora é usar a classe `List`, do **Visual Basic .NET**. No projeto **ByteBank.SistemaAgencia**, acesse o código fonte do formulário `Frm_ContasCorrentesListas`. Na declaração genérica da variável `Lista` substitua:

```
Dim Lista As New Lista(Of ContaCorrente)
```

Por:

```
Dim Lista As New List(Of ContaCorrente)
```

Com isso, não é mais utilizada a classe `Lista`, que simula um array genérico com tamanho variável, e sim a classe `List`, do Visual Basic .NET.

5) Com estas mudanças, os métodos também sofrem modificações. Substitua o método `Adicionar` por `Add`. Por exemplo, substitua:

```
Lista.Adicionar(conta)
```

Por:

```
Lista.Add(conta)
```

6) Faça o mesmo com o método `Remover`, substitua por `Remove`. Por exemplo, substitua:

```
Lista.Remover(conta)
```

Por:

```
Lista.Remove(conta)
```

7) Para obter o tamanho das listas, era utilizado o método `Tamanho`. Substitua-o por `Count`. Por exemplo, substitua:

```
For i As Integer = 0 To Lista.tamanho - 1
```

Por:

```
For i As Integer = 0 To Lista.Count - 1
```

8) Na sub-rotina `Btn_AdicionaVarios_Click`, substitua o código original pelo abaixo, que instancia uma lista temporária, adiciona as contas nele e passa-a como parâmetro para o `AddRange`, para adicionar uma lista de contas na sua `Lista`:

```
Private Sub Btn_AdicionaVarios_Click(sender As Object, e As EventArgs)
        Handles Btn_AdicionaVarios.Click

    Dim ListaTemporaria As New List(Of ContaCorrente)

    If Txt_Agencia1.Text <> "" And Txt_Conta1.Text <> "" Then
        Dim Conta1 As New ContaCorrente(Txt_Agencia1.Text, Txt_Conta1.Text)
        ListaTemporaria.Add(Conta1)
        If Txt_Agencia2.Text <> "" And Txt_Conta2.Text <> "" Then
            Dim Conta2 As New ContaCorrente(Txt_Agencia2.Text, Txt_Conta2.Text)
            ListaTemporaria.Add(Conta2)
            If txt_Agencia3.Text <> "" And Txt_Conta3.Text <> "" Then
                Dim Conta3 As New ContaCorrente(txt_Agencia3.Text, Txt_Conta3.Text)
                ListaTemporaria.Add(Conta3)
                If Txt_Agencia4.Text <> "" And Txt_Conta4.Text <> "" Then
                    Dim Conta4 As New ContaCorrente(Txt_Agencia4.Text, Txt_Conta4.Text)
                    ListaTemporaria.Add(Conta4)
                    Lista.AddRange(ListaTemporaria)
                    Txt_Array.Text = EscreverElementosArrayIndexador()
                Else
                    Lista.AddRange(ListaTemporaria)
                    Txt_Array.Text = EscreverElementosArrayIndexador()
                End If
            Else
                Lista.AddRange(ListaTemporaria)
```

```
                Txt_Array.Text = EscreverElementosArrayIndexador()
            End If
        Else
            Lista.AddRange(ListaTemporaria)
            Txt_Array.Text = EscreverElementosArrayIndexador()
        End If
    End If

End Sub
```

9) Na sub-rotina `New()`, há alguns erros ocasionadas pela retirada dos componentes no passo 2. Então, apague todas as referências a esses componentes retirados e modifique a inicialização dos *Label* `Lbl_Agencia`, `Lbl_Agencia1`, `Lbl_Agencia2`, `Lbl_Agencia3` e `Lbl_Agencia4`:

```
Lbl_Agencia.Text = "Conta Corrente"
Lbl_Agencia1.Text = "Conta Corrente 1"
Lbl_Agencia2.Text = "Conta Corrente 2"
Lbl_Agencia3.Text = "Conta Corrente 3"
Lbl_Agencia4.Text = "Conta Corrente 4"
```

10) Na classe `ContaCorrente`, do projeto **ByteBank.Bibliotecas**, tem que ser possível passar o nome do cliente para o seu construtor. Então, adicione o seguinte construtor:

```
Public Sub New(CodigoAgencia As Integer, NumeroConta As Integer, NomeCliente As String)

    If (CodigoAgencia <= 0) Then

        Dim vParametro As String
        vParametro = NameOf(CodigoAgencia)

        Dim Erro As New ArgumentException("Código da agência menor que zero !!!!", vParametro)
        Throw Erro


    ElseIf (NumeroConta <= 0) Then

        Dim vParametro As String
        vParametro = NameOf(NumeroConta)

        Dim Erro As New ArgumentException("Número da conta menor que zero !!!!", vParametro)
        Throw Erro

    ElseIf (NomeCliente = "") Then

        Dim vParametro As String
        vParametro = NameOf(NomeCliente)

        Dim Erro As New ArgumentException("Nome do Cliente em Branco !!!!", vParametro)
        Throw Erro

    End If

    agencia = CodigoAgencia
    Me.numero = NumeroConta
    m_TotalDeContasCriadas += 1
```

```
        m_TaxaOperacao = 30 / m_TotalDeContasCriadas

        Dim Cliente As New Cliente
        Cliente.nome = NomeCliente

        titular = Cliente

    End Sub
```

11) Ainda na classe `ContaCorrente`, implemente o método `ToString`, que é derivado de `Object`, para escrever a agência, o número e o nome do correntista:

```
    Public Overrides Function ToString() As String
        Return $"Agencia: {agencia.ToString} Conta: {numero.ToString} Nome: {titular.nome}"
    End Function
```

12) Volte ao código fonta de `Frm_ContasCorrentesListas` e modifique as instâncias da classe `ContaCorrente`, passando o nome do correntista que estará contido nos novos *TextBox* adicionados. Por exemplo, de:

```
    Dim conta As New ContaCorrente(Txt_Agencia.Text, Txt_Conta.Text)
```

Para:

```
    Dim conta As New ContaCorrente(Txt_Agencia.Text, Txt_Conta.Text, Txt_Nome.Text)
```

Utilize o *TextBox* correto. Se estiver usando `Txt_Conta`, use `Txt_Nome`. Se for `Txt_Conta1`, use `Txt_Nome1`, e assim por diante.

13) Na função `EscreverElementosArrayIndexador` e inclua o uso da função `ToString`:

```
    Public Function EscreverElementosArrayIndexador() As String

        Dim elementos As String = String.Empty
        For i As Integer = 0 To Lista.Count - 1
            'elementos += Lista(i).agencia.ToString + " - " + Lista(i).numero.ToString + vbCrLf
            elementos += Lista(i).ToString + vbCrLf
        Next
        Return elementos

    End Function
```

14) Inclua a função `ToString` também na função `EscreverElementosArray`:

```
    Public Function EscreverElementosArray() As String

        Dim elementos As String = String.Empty
        For i As Integer = 0 To Lista.Count - 1
            'elementos += Lista(i).agencia.ToString + " - " + Lista(i).numero.ToString + vbCrLf
```

```
            elementos += Lista(i).ToString + vbCrLf
        Next
        Return elementos

    End Function
```