Programação em C# com Visual Studio .NET

Prof. Alessandro Brawerman

Avançando WinForms

- WinForms é uma coleção de classes e tipos que encapsulam e estendem a API win32 em um modelo de objetos organizados.
- Os componentes usados para criar aplicações GUI do Windows são fornecidos como classes .NET e disponibilizados por herança.

Avançando WinForms

- Os forms são tipos de interface Windows e podem ser os seguintes:
  - MDI (Multiple Document Interface) – suporta múltiplos documentos abertos simultaneamente
  - SDI (Single Document Interface) – permite a abertura de apenas um documento de cada vez.
- Dentro de SDIs:
  - Forms normais: permitem que o usuário trabalhe com eles e faça algo em outro form com ele ainda ativo.

Avançando WinForms

- Forms modais: não permitem que se faça nada dentro do sistema, exceto manipular o form ativo.
- Os sistemas comerciais em sua grande maioria são SDI.

Um Exemplo MDI

- Windows Application com os seguintes elementos:
  - O form principal (MainForm)
  - Os forms filhos
  - O menu do form principal

Um Exemplo MDI

- Form principal (MainForm)
  - Name: MainForm
  - IsMdiContainer: true
  - Text: Formulário Principal
- Adicione o menu ao form principal.
  - Arquivo – submenus: Novo, Fechar e Sair
  - Janela – submenus: Horizontal, Vertical e Cascata

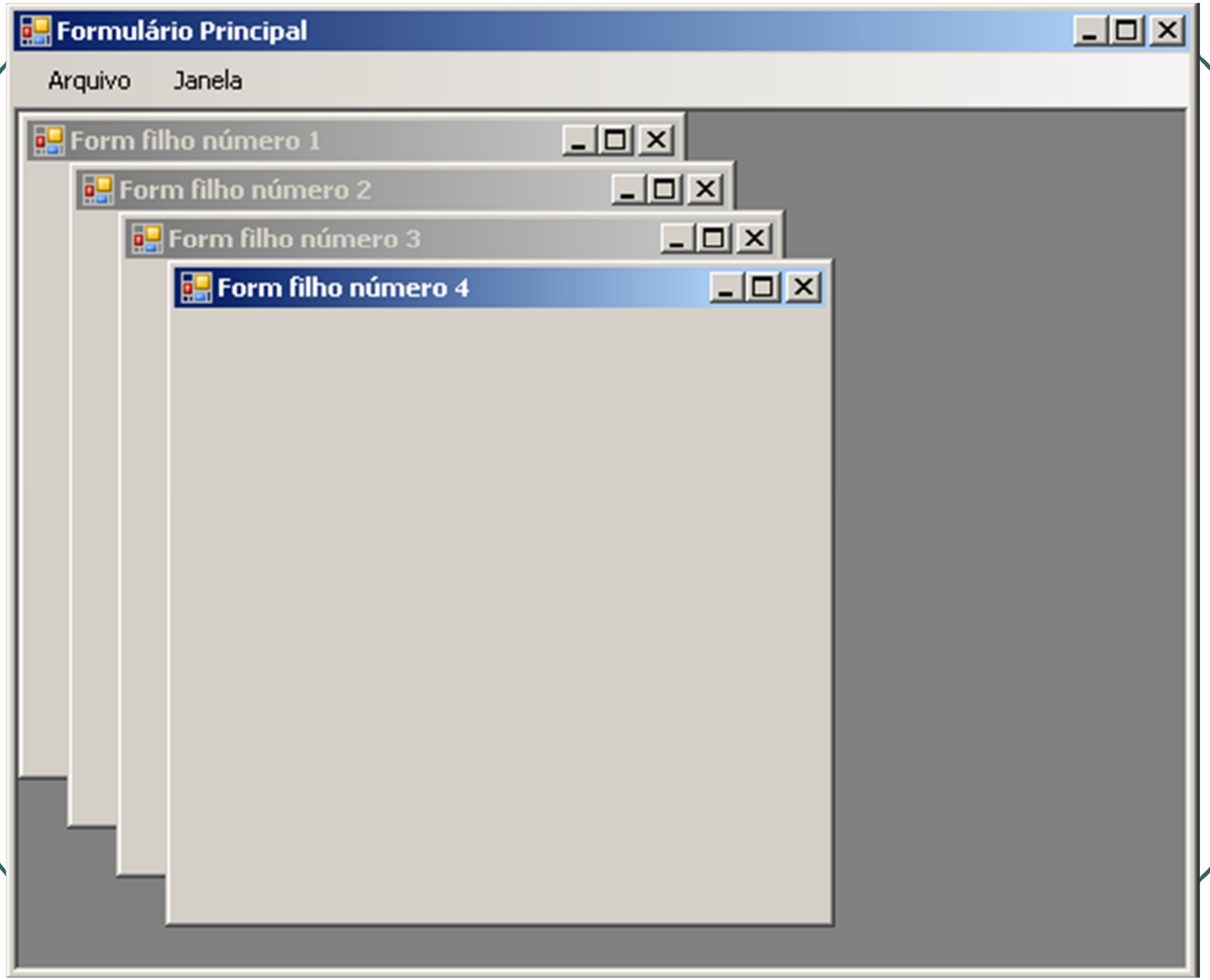
Um Exemplo MDI

- Form filho
  - No solution explorer, clique no nome do projeto com o botão esquerdo e em Add/New Item
    - Selecione Windows Forms
  - Name: formSon
- Crie uma variável do tipo inteiro privada para armazenar a quantidade de forms filhos criados dinamicamente.
  - private int formCount = 1;

Um Exemplo MDI

- Clique no item de menu Novo e insira o seguinte código que criará nossos forms filhos:

```
private void novoToolStripMenuItem_Click(object sender, EventArgs e)
{
    formSon myForm = new formSon();
    myForm.MdiParent = this;
    myForm.Text = "Form filho número " + formCount.ToString();
    myForm.Show();
    formCount++;
}
```


Formulário Principal
Arquivo
Janela
Form filho número 1
Form filho número 2
Form filho número 3
Form filho número 4

Um Exemplo MDI

- Para fechar os forms, clique no submenu fechar e insira o seguinte código:

```
private void fecharToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (this.MdiChildren.Length != 0)
        this.ActiveMdiChild.Close();
}
```

Um Exemplo MDI

- Para organizar as janelas:

```
private void horizontalToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileHorizontal);
}

private void verticalToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileVertical);
}

private void cascataToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.Cascade);
}
```

O Projeto SDI

- Crie um Windows Application
- Renomeie o form para mainForm.
- Insira um componente menu:
  - Arquivo – Verde, Azul e Fechar.
- Adicione 2 novos forms.
- Renomeie-os para frmVerde e frmAzul.
- Altere a cor de fundo para a cor de batismo.

O Projeto SDI

```
private void verdeToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmVerde verde = new frmVerde();
    //verde.Show();
    try
    {
        verde.ShowDialog();
    }
    finally
    {
        verde.Dispose();
    }
}
```

O Projeto SDI

- Show() – não deixa o foco fixo no formulário criado.
- ShowDialog() – o foco é somente no formulário criado, não podendo fazer mais nada até fechá-lo.

Melhorando o Projeto SDI

- Insira mais um elemento amarelo (menu e form)
- Vamos generalizar para não haver repetição de código.

Melhorando o Projeto SDI

```
private void amareloToolStripMenuItem_Click(object sender, EventArgs e)
{
    Execute(typeof(frmAmarelo));
}

private void Execute(Type TipoForm)
{
    Form f = (Form)Activator.CreateInstance(TipoForm);
    try
    {
        f.ShowDialog();
    }
    finally
    {
        f.Dispose();
    }
}
```

Passando Valores entre Forms

- O form chamador deve sempre conter o valor a ser passado.
- Imagine que uma rotina será implementada no frmAmarelo quando o mainForm o chamar.
- Depois notamos que o frmAzul também chamará o frmAmarelo.
- Se a rotina está no frmAmarelo teremos que acrescentar um if para ver se quem chamou o Amarelo foi o mainForm.

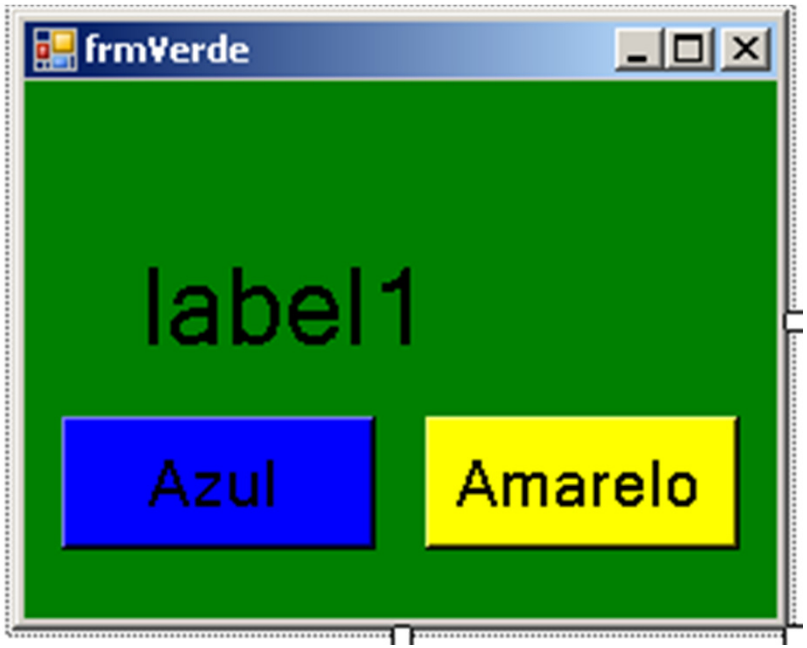
Passando Valores entre Forms

- Imagine a situação que você tem vários forms chamando outros e precisará de vários ifs para saber quem chamou quem.
- Portanto, para passar valor entre forms é mais aconselhável deixar o valor ou rotina no form chamador.
- Continue o exercício anterior colocando um label em cada um dos forms coloridos

Passando Valores entre Forms

- A idéia é que este label apresente na sua propriedade Text quem o invocou.
- Deixe todos os labels com o mesmo nome.
- Insira 2 botões em cada um dos forms coloridos para chamar os outros forms coloridos.
- Coloque no Text dos botões a cor do form a ser chamado.

Passando Valores entre Forms



Passando Valores entre Forms

- Alterar a chamada dos forms coloridos no mainForm.
- Alterar o modificador dos labels para public.
- Inserir o seguinte código:

```
frmAzul azul = new frmAzul();
try
{
    azul.label1.Text = "mainForm";
    azul.ShowDialog();
}
finally
{
    azul.Dispose();
}
```

Propriedades Mais Comuns de um Form

- AcceptButton
  - Determina que um botão do form é clicado quando a tecla Enter é pressionada.
- AllowDrop
  - Determina se o form aceita drag-n-drop messages.
- AutoScale
  - Determina se o form pode escalar dependendo da fonte.

Propriedades Mais Comuns de um Form

- AutoScroll
  - Habilita barra de rolagem se algum componente estiver em algum lugar não visível do form.
- BackColor e ForeColor
  - Determina as cores do form.
- BackgoundImage
  - Determina uma imagem como background no form.

Propriedades Mais Comuns de um Form

- CancelButton
  - Determina que um botão do form é ciclado quando a tecla Esc é pressionada.
- ControlBox, MazimixeB
  - Maximizar, minimizar, fechar e help no captio do form.
- FormBorderStyle
  - Especifica como a borda do form será apresentada.

Propriedades Mais Comuns de um Form

- IsMdiContainer
  - Determina se o form é um MDIContainer ou não.
- Language
  - Determina a linguagem do form.
- Locked
  - Habilita a habilidade de mover e redesenhar componentes do form.

Propriedades Mais Comuns de um Form

- Opacity
  - Habilita a transparência do form. 0% é transparente e 100% é opaco.
- ShowInTaskBar
  - Habilita mostrar o form na TaskBar do Windows.
- Size
  - Determina as dimensões iniciais do form.

Propriedades Mais Comuns de um Form

- StartPosition
  - Posição inicial do form na área de trabalho.
- Text
  - Texto que será mostrado na caption bar.
- Window State
  - É o estado inicial da janela (minimizado, normal ou maximizado).

Componentes Dinâmicos

- Adicionar componentes durante a execução de um programa pode muitas vezes ser útil.
- Não só componentes, mas também eventos e delegações.
- Suponha um projeto Windows Application e que desejamos inserir um novo botão a partir de um clique no form com o botão esquerdo.

Componentes Dinâmicos

```
private void Form1_Click(object sender, EventArgs e)
{
    Button b1 = new Button();
    b1.Text = "Botão Dinâmico";
    b1.Location = new Point(10, 10);
    b1.Size = new Size(140, 20);
    this.Controls.Add(b1);
}
```

Componentes Dinâmicos

Componentes Dinâmicos

- Para criarmos eventos de forma dinâmica precisamos fazer a assinatura do evento, ou seja, fornecer o código que será executado quando um evento é gerado no formato idêntico ao manipulador de eventos.
- A assinatura faz parte da definição do evento e é especificada por uma delegação (delagation).
- Delegation é um handle para uma função em memória.

Componentes Dinâmicos

- Algo como tipos por referência porém ao invés de conter valores para variáveis, contém um endereço de uma função.
- Vamos criar uma calculadora em que todos os seus botões e eventos Click serão criados dinamicamente.
- Crie um projeto Windows Application e coloque apenas um TextBox próximo ao topo.

Componentes Dinâmicos

- Name = tbDisplay.
- BorderStyle = none
- Enabled = false
- Text = 0
- Vamos inicialmente declarar as variáveis que vamos utilizar:


```
// Variáveis para guardar os valores
private float Valor1 = 0;
private float Valor2 = 0;
// Variáveis enumerações para operadores e números
private enum FlagOperador { Vazio, Mais, Menos, Mult, Div };
private enum KeyType { Vazio, Numero, Cancelar, Operador, Igual };
private FlagOperador TipoOperador;
```

Componentes Dinâmicos

- O próximo passo é criar a rotina que vai gerar, denominar e mostrar os botões no form.

```
private void AdicionaBotoes(int Left, int Top, string key, KeyType Tipo)
{
    //Cria botão e adiciona ao form
    Button novoBotao = new Button();
    this.Controls.Add(novoBotao);

    //Valores para o novo botão
    novoBotao.Top = Top;
    novoBotao.Left = Left;
    novoBotao.Width = 35;
    novoBotao.Height = 35;
    novoBotao.Text = key;
}
```

Componentes Dinâmicos

```
private void CriaBotoes()
{
    // Loop para criar os botões com números
    int j;
    for (int i = 0; i < 9; i++)
    {
        j = (int)(i / 3);
        this.AdicionaBotoes(25 + (i%3)*50, 50 + ((2-j)*50), (i+1).ToString(), KeyType.Numero);
    }

    //Adiciona os demais operadores e especiais
    this.AdicionaBotoes(25, 200, "0", KeyType.Numero);
    this.AdicionaBotoes(75, 200, "C", KeyType.Cancelar);
    this.AdicionaBotoes(125, 200, "=", KeyType.Igual);
    this.AdicionaBotoes(175, 50, "+", KeyType.Operador);
    this.AdicionaBotoes(175, 100, "-", KeyType.Operador);
    this.AdicionaBotoes(175, 150, "*", KeyType.Operador);
    this.AdicionaBotoes(175, 200, "/", KeyType.Operador);
}
```

Componentes Dinâmicos

- Devemos agora inserir os eventos de Click em cada um dos botões.
- Adicione o código abaixo no método AdicioneBotoes.

Componentes Dinâmicos

```
switch (Type)
{
    case KeyType.Numero:
        novoBotao.Click += new System.EventHandler(ClickNumero);
        break;
    case KeyType.Operador:
        novoBotao.Click += new System.EventHandler(ClickOperador);
        break;
    case KeyType.Igual:
        novoBotao.Click += new System.EventHandler(ClickIgual);
        break;
    case KeyType.Cancelar:
        novoBotao.Click += new System.EventHandler(ClickCancelar);
        break;
}
```

Componentes Dinâmicos

- O objeto usado para gerar eventos é uma instância da classe System.EventHandler.
- Ao escrever a função disparada pelo evento, como ClickNumero, devemos nos preocupar com que a assinatura da função seja idêntica à delegação padrão do .NET para o evento Click.

Componentes Dinâmicos

- Os valores serão acumulados na variável Valor1 até que um operador seja clicado.
- Então o tipo de operador é armazenado e os próximos valores lançados são colocados no Valor2.
- Ao clicar em =, o operador armazenado no TipoOperador é resgatado para podermos fazer a operação em si.
- O valor resultado volta para o Valor1 que é então impresso na tela.

Componentes Dinâmicos

```
private void ClickNumero(Object sender, System.EventArgs e)
{
    string Key = ((Button)sender).Text;
    Valor1 = Valor1 * 10 + Convert.ToInt32(Key);
    tbDisplay.Text = Valor1.ToString();
}

private void ClickCancelar(Object sender, System.EventArgs e)
{
    Valor1 = 0;
    Valor2 = 0;
    tbDisplay.Text = "0";
}
```

Componentes Dinâmicos

```
private void ClickOperador(Object sender, System.EventArgs e)
{
    string Key = ((Button)sender).Text;
    switch (Key)
    {
        case "+":
            TipoOperador = FlagOperador.Mais;
            break;
        case "-":
            TipoOperador = FlagOperador.Menos;
            break;
        case "*":
            TipoOperador = FlagOperador.Mult;
            break;
        case "/":
            TipoOperador = FlagOperador.Div;
            break;
    }
    Valor2 = Valor1;
    Valor1 = 0;
}
```

Componentes Dinâmicos

```
private void ClickIgual(Object sender, System.EventArgs e)
{
    switch (TipoOperador)
    {
        case FlagOperador.Mais:
            Valor1 = Valor2 + Valor1;
            break;
        case FlagOperador.Menos:
            Valor1 = Valor2 - Valor1;
            break;
        case FlagOperador.Mult:
            Valor1 = Valor2 * Valor1;
            break;
        case FlagOperador.Div:
            Valor1 = Valor2 / Valor1;
            break;
    }
    TipoOperador = FlagOperador.Vazio;
    tbDisplay.Text = Valor1.ToString();
    Valor2 = 0;
}
```