LINGUAGEM DE PROGRAMAÇÃO

Orientada a objeto

Roteiro da aula

Revisão:

- Comandos de decisão
- Loop
- Desvio de Fluxo
- Windows Form

Comandos de Decisão

- IF – Else
- Switch

Revisão

- A2-Exercicio 5 e 6 (opcional).
- A2-Exercicio 7 e 8

Loops

- For
- While
- Do-While
- Foreach-in

  Loop que permite varrer todos os elementos de um array ou coleção

Desvio no fluxo do programa

- É feito usando instruções próprias para o desvio do fluxo do programa que causa transferência imediata do controle do programa:

1. Break
2. Continue
3. Goto
4. Return

Break

```
private void cmdListar_Click(object sender, EventArgs e)
{

    int i = 1;
    while (i <= 10)
    {
        lblStatus.Text += i + " ";
        i++;

        if (i == 5)
        {
            break;
        }

    }

}
```

Continue

```
private void cmdListar_Click(object sender, EventArgs e)
{

    int i = 0;
    while (i < 10)
    {

        i++;
        if (i == 5)
        {
            continue;
        }

        lblStatus.Text += i + " ";
    }

}
```

Return

```
private void cmdListar_Click(object sender, EventArgs e)
 {
        lblStatus.Text = " Antes do return ";
        return;
        lblStatus.Text = " Após o retum ";

 }
```

Goto

```
private void cmdListar_Click(object sender, EventArgs e)
{

    int i = 1;

    if (i == 1) goto compra;

    if (i == 2) goto venda;

    compra:
        lblStatus.Text = "Compra ";
        return;
    venda:
        lblStatus.Text = "Venda ";
        return;

}
```

Windows form

- Atalhos em botões → propriedade Text - &Fechar (ctrl+F)
- Fechar janela → Close() ou this.Close() ou Application.Exit();
- Propriedade TextBox → Multiline permite varias linhas.
  ScrollBars para barras de rolagem
- CheckBox e RadioButton → Exemplos:

```
string permite= " ";
if ( chkAltera.Checked)
    permite+= chkAltera.Text;
if (chkConsulta.Checked)
    permite+= chkConsulta.Text
```

```
string sexo = " ";
if ( rbFeminino.Checked)
    sexo = rbFeminino.Text;
else
    sexo = rbMasculino.Text
```

Windows form

- ListBox e ComboBox→ exibe vários itens. Com um duplo clique na lista ou combo tenho o exemplo

lblRes.Text = “Selecionado ” + lbCarro.SelectedItem.ToString()

- Obter item → int indice=0 indice = lbCarro.SelectedIndex
- Propriedade Sorted → Lista ordenada
- Metodos :

    Items.Add → adiciona à lista

    Items.RemoveAt(1) → remove o índice indicado

    Items.Clear() → remove todos items.

    Items.Contains() →verifica se contem

```
If (! (lbCarro.Items.Contains(“Fiat”) )
lbCarro.Items.Add(“Fiat”);
```

Windows Form

- **MenuStrip** → criação de menus.

  Para linha separadora clique na seta a direita do TypeHere.

  Acesso rapido use “&” antes da letra (Alt + Letra).

  Atalho através da propriedade ShortcutKeys.

  Adicionar imagem no menu, propriedade Image

- **ContextMenuStrip** → menu de atalho com tecla direita do mouse

Roteiro da aula

Revisão:

- **Tratamento de erros**
- **Debug**

Tratamento de erros

- Localizar e corrigir os erros é chamado de depuração
- Erros impedem que o programa seja iniciado, interrompem ou travam a execução ou não fornecem os resultados esperados.
- Exemplo projeto calculadora com digitação de caracter
- Tipos de erro:

    Erro de compilação: não executa o programa(duble)

    Erro de Tempo de Execução: durante execução.(char)

    Erro de lógica: mais difíceis de corrigir

Debug

- Criação de um ponto de interrupção (breakpoint) para executar passo a passo as instruções do programa
- Clique na barra cinza, a esquerda no numero da linha para marcar o ponto de interrupção.
- Execute e use F11 ou “Step Into” para prosseguir.

Tratamento de Exceções

- Maneira de lidar com qualquer situação inesperada que apareça durante a execução do programa(acesso ao Banco, impressora não localizada)
- Try
- Catch
- Finally
- Exemplo calculadora

valor2 =0

```
private void cmdDividir_Click(object sender, EventArgs e)
{
    try
    {
        int v1 = Convert.ToInt32(txtValor1.Text);
        int v2 = Convert.ToInt32(txtValor2.Text);
        int r = v1 / v2;
        lblResultado.Text = "Resultado: " + r;
    }
    catch
    {
        lblResultado.Text = "Valores incorretos";
    }
}
```

Classe Exception

- A classe Exception contem informações detalhada sobre o erro.

```
...
catch(Exception x)
{
    lblStatus.Text = x.Message;
}
```

Criando as próprias exceções

- Método para Validação

```
private static void ValidarNome(string nome)
{
    if (nome.Trim().Length ==0)
    {
        string msg = "O nome não pode estar em branco";
        ApplicationException e = new ApplicationException(msg);
        throw e;
    }
}

private static void ValidarNota(double nota)
{
    if ((nota < 0) || (nota > 10))
    {
        string msg = "A nota deve estar entre 0 e 10";
        ApplicationException e = new ApplicationException(msg);
        throw e;
    }
}
```

Criando as próprias exceções

- Chamada aos métodos

```
private void cmdIncluir_Click(object sender, EventArgs e)
{
    try
    {
        string nome = txtNome.Text;
        ValidarNome(nome);
        double nota = Convert.ToDouble(txtNota.Text);
        ValidarNota(nota);
        lblStatus.Text = "Nota registrada!";
    }
    catch (ApplicationException x)
    {
        lblStatus.Text = x.Message;
    }
    catch (Exception x)
    {
        lblStatus.Text = x.Message;
    }
}
```

Bloco Finally

- Instruções desse bloco sempre serão executadas, independente de ocorrer ou não um erro (fechar BD)