📌

Revisão de Programação Orientada à Objetos II💻

👨🏾‍🏫**Professor:** Salatiel Luz Marinho

📩**e-mail:** salatiel.marinho@docente.unip.br

📜Referencial Teórico da Aula

✅ O que é ListBox em uma aplicação Windows Forms?

Uma `ListBox` é um controle em uma aplicação Windows Forms que permite exibir uma lista de itens. O usuário pode selecionar um ou mais itens da lista, dependendo da configuração do controle. A `ListBox` é frequentemente usada para exibir listas de dados, como nomes, números ou objetos.

Exemplo de Uso da ListBox em C#

Abaixo está um exemplo simples de como usar uma `ListBox` em uma aplicação Windows Forms usando a linguagem de programação C#.

1. **Adicionar uma ListBox ao Formulário:**

   Primeiro, arraste e solte uma `ListBox` da caixa de ferramentas para o seu formulário no designer do Visual Studio.

2. **Adicionar Itens à ListBox:**

   No evento `Load` do formulário, você pode adicionar itens à `ListBox` programaticamente.

```
private void Form1_Load(object sender, EventArgs e)
{
    // Adiciona itens à ListBox
    listBox1.Items.Add("Item 1");
```

```
    listBox1.Items.Add("Item 2");
    listBox1.Items.Add("Item 3");
}
```

3. **Manipular a Seleção de Itens na ListBox:**

   Você pode usar o evento `SelectedIndexChanged` para executar ações quando um item é selecionado.

```
private void listBox1_SelectedIndexChanged(object sende
r, EventArgs e)
{
    // Obtém o item selecionado
    string selectedItem = listBox1.SelectedItem.ToString
();
    MessageBox.Show("Você selecionou: " + selectedItem);
}
```

Exemplo

Aqui está um exemplo completo de uma aplicação Windows Forms que usa uma `ListBox` para exibir e selecionar itens.

```
using System;
using System.Windows.Forms;

namespace ExemploListBox
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Adiciona itens à ListBox
            listBox1.Items.Add("Maçã");
```

```
            listBox1.Items.Add("Banana");
            listBox1.Items.Add("Laranja");
            listBox1.Items.Add("Uva");
        }

        private void listBox1_SelectedIndexChanged(object s
ender, EventArgs e)
        {
            // Exibe o item selecionado em uma MessageBox
            if (listBox1.SelectedItem != null)
            {
                string selectedItem = listBox1.SelectedIte
m.ToString();
                MessageBox.Show("Você selecionou: " + selec
tedItem);
            }
        }
    }
}
```

Neste exemplo, a `ListBox` é preenchida com uma lista de frutas quando o formulário é carregado. Quando o usuário seleciona uma fruta na `ListBox`, o nome da fruta selecionada é exibido em uma `MessageBox`.

✅O que é um Label em uma aplicação Windows Forms?

Um `Label` é um controle em uma aplicação Windows Forms que serve para exibir texto estático no formulário. É comumente usado para fornecer descrições, rótulos ou instruções para outros controles no formulário, como caixas de texto ou botões.

Exemplo de Uso do Label em C#

Abaixo está um exemplo simples de como usar um `Label` em uma aplicação Windows Forms usando a linguagem de programação C#.

1. **Adicionar um Label ao Formulário:**

   Primeiro, arraste e solte um `Label` da caixa de ferramentas para o seu formulário no designer do Visual Studio.

2. **Configurar o Texto do Label:**

   No evento `Load` do formulário, você pode configurar o texto do `Label` programaticamente.

```
private void Form1_Load(object sender, EventArgs e)
{
    // Configura o texto do Label
    label1.Text = "Bem-vindo à minha aplicação!";
}
```

Exemplo

Aqui está um exemplo completo de uma aplicação Windows Forms que usa um `Label` para exibir uma mensagem de boas-vindas.

```
using System;
using System.Windows.Forms;

namespace ExemploLabel
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Configura o texto do Label
            label1.Text = "Bem-vindo à minha aplicação!";
        }
    }
}
```

Neste exemplo, quando o formulário é carregado, o `Label` exibe a mensagem "Bem-vindo à minha aplicação!".

✅ O que é TextBox em uma aplicação Windows Forms?

Um `TextBox` é um controle em uma aplicação Windows Forms que permite ao usuário inserir texto. É comumente usado para coletar entradas de texto do usuário, como nomes, endereços, números de telefone, entre outros.

Exemplo de Uso do TextBox em C#

Abaixo está um exemplo simples de como usar um `TextBox` em uma aplicação Windows Forms usando a linguagem de programação C#.

1. **Adicionar um TextBox ao Formulário:**

   Primeiro, arraste e solte um `TextBox` da caixa de ferramentas para o seu formulário no designer do Visual Studio.

2. **Configurar o TextBox:**

   No evento `Load` do formulário, você pode configurar o `TextBox` programaticamente, por exemplo, definindo seu texto inicial.

   ```
   private void Form1_Load(object sender, EventArgs e)
   {
       // Configura o texto inicial do TextBox
       textBox1.Text = "Digite seu nome aqui";
   }
   ```

3. **Obter o Texto do TextBox:**

   Você pode obter o texto inserido no `TextBox` quando, por exemplo, um botão é clicado.

   ```
   private void button1_Click(object sender, EventArgs e)
   {
       // Obtém o texto do TextBox e exibe em uma MessageBo
   x
       string enteredText = textBox1.Text;
       MessageBox.Show("Você digitou: " + enteredText);
   }
   ```

Exemplo

Aqui está um exemplo completo de uma aplicação Windows Forms que usa um `TextBox` para coletar e exibir o texto inserido pelo usuário.

```
using System;
using System.Windows.Forms;

namespace ExemploTextBox
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Configura o texto inicial do TextBox
            textBox1.Text = "Digite seu nome aqui";
        }

        private void button1_Click(object sender, EventArgs
e)
        {
            // Obtém o texto do TextBox e exibe em uma Mess
ageBox
            string enteredText = textBox1.Text;
            MessageBox.Show("Você digitou: " + enteredTex
t);
        }
    }
}
```

Neste exemplo, um `TextBox` é usado para permitir que o usuário digite seu nome. Quando o botão é clicado, o texto inserido no `TextBox` é exibido em uma `MessageBox`.

✅ Exemplos de Validadores em C#

Para validar se um e-mail é válido em C#, você pode usar a classe `System.Net.Mail.MailAddress`. Aqui está um exemplo de como fazer isso:

```
using System;
using System.Net.Mail;

public class EmailValidator
{
    public static bool IsValidEmail(string email)
    {
        try
        {
            var mailAddress = new MailAddress(email);
            return true;
        }
        catch (FormatException)
        {
            return false;
        }
    }

    public static void Main()
    {
        string email = "exemplo@dominio.com";
        bool isValid = IsValidEmail(email);
        Console.WriteLine("O e-mail é válido? " + isValid);
    }
}
```

Neste exemplo, a função `IsValidEmail` tenta criar uma instância de `MailAddress` com o e-mail fornecido. Se o e-mail não estiver no formato correto, uma `FormatException` será lançada, e a função retornará `false`.

Para garantir que o `TextBox` só permita a **inserção de números em C#**, podemos utilizar evento `KeyPress` e verificar se a tecla pressionada é um número ou uma tecla de controle (como Backspace). Aqui está um exemplo de como fazer isso:

```
private void textBox1_KeyPress(object sender, KeyPressEvent
Args e)
{
    // Permitir apenas números e teclas de controle (como B
ackspace)
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyCh
ar))
    {
        e.Handled = true;
    }
}
```

Neste exemplo, a função `textBox1_KeyPress` verifica se a tecla pressionada é um número (`char.IsDigit(e.KeyChar)`) ou uma tecla de controle (`char.IsControl(e.KeyChar)`). Se não for, a propriedade `e.Handled` é definida como `true`, impedindo que o caractere seja inserido no `TextBox`.

Exemplo de como tratar a exceção de divisão por zero em C#:

```
using System;

public class DivisaoPorZero
{
    public static void Main()
    {
        try
        {
            // Solicita ao usuário que insira dois números
            Console.Write("Digite o numerador: ");
            int numerador = Convert.ToInt32(Console.ReadLin
e());

            Console.Write("Digite o denominador: ");
            int denominador = Convert.ToInt32(Console.ReadL
ine());

            // Tenta realizar a divisão
            int resultado = Dividir(numerador, denominado
```

```
r);

            // Exibe o resultado
            Console.WriteLine("O resultado da divisão é: "
+ resultado);
        }
        catch (DivideByZeroException)
        {
            // Captura a exceção de divisão por zero e exib
e uma mensagem de erro
            Console.WriteLine("Erro: Não é possível dividir
por zero.");
        }
        catch (FormatException)
        {
            // Captura a exceção de formato inválido e exib
e uma mensagem de erro
            Console.WriteLine("Erro: Formato de número invá
lido.");
        }
        catch (Exception ex)
        {
            // Captura qualquer outra exceção e exibe uma m
ensagem de erro
            Console.WriteLine("Erro: " + ex.Message);
        }
    }

    public static int Dividir(int numerador, int denominado
r)
    {
        // Realiza a divisão e retorna o resultado
        return numerador / denominador;
    }
}
```

Neste exemplo, o código solicita ao usuário que insira dois números, tenta realizar a divisão e trata a exceção `DivideByZeroException` caso o usuário insira

zero como denominador. Além disso, o código também trata a exceção `FormatException` para entradas de formato inválido e qualquer outra exceção genérica.

Implementação de MessageBox

Para implementar mensagens de confirmação em uma aplicação Windows Forms utilizando a linguagem de programação C#, você pode utilizar a classe `MessageBox`. A seguir, é apresentado um exemplo de como exibir uma mensagem de confirmação antes de realizar uma ação, como a remoção de um item em uma ListBox.

Exemplo de Uso do `MessageBox` para Confirmação

1. **Adicionar uma ListBox ao Formulário:**

   Primeiro, arraste e solte uma `ListBox` e um `Button` da caixa de ferramentas para o seu formulário no designer do Visual Studio.

2. **Adicionar Itens à ListBox:**

   No evento `Load` do formulário, você pode adicionar itens à `ListBox` programaticamente.

   ```
   private void Form1_Load(object sender, EventArgs e)
   {
       // Adiciona itens à ListBox
       listBox1.Items.Add("Item 1");
       listBox1.Items.Add("Item 2");
       listBox1.Items.Add("Item 3");
   }
   ```

3. **Adicionar Evento de Clique ao Botão:**

   Adicione um evento de clique ao botão para exibir a mensagem de confirmação antes de remover o item selecionado na `ListBox`.

   ```
   private void button1_Click(object sender, EventArgs e)
   {
       // Verifica se há um item selecionado na ListBox
       if (listBox1.SelectedItem != null)
       {
   ```

```
        // Exibe uma mensagem de confirmação
        DialogResult result = MessageBox.Show("Você tem
certeza que deseja remover o item selecionado?", "Confir
mação", MessageBoxButtons.YesNo, MessageBoxIcon.Questio
n);

        // Verifica a resposta do usuário
        if (result == DialogResult.Yes)
        {
            // Remove o item selecionado
            listBox1.Items.Remove(listBox1.SelectedIte
m);
            MessageBox.Show("Item removido com sucess
o.", "Informação", MessageBoxButtons.OK, MessageBoxIcon.
Information);
        }
        else
        {
            MessageBox.Show("Ação cancelada.", "Informaç
ão", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    else
    {
        MessageBox.Show("Por favor, selecione um item pa
ra remover.", "Erro", MessageBoxButtons.OK, MessageBoxIc
on.Error);
    }
}
```

Explicação do Código

- **MessageBox.Show:** Este método é usado para exibir uma caixa de mensagem ao usuário.
  - O primeiro parâmetro é a mensagem que será exibida.
  - O segundo parâmetro é o título da caixa de mensagem.

    - O terceiro parâmetro especifica os botões que serão exibidos na caixa de mensagem (`MessageBoxButtons.YesNo` neste caso).
    - O quarto parâmetro especifica o ícone da caixa de mensagem (`MessageBoxIcon.Question` neste caso).
- **DialogResult:** A resposta do usuário (Yes ou No) é capturada e armazenada na variável `result`.
- **Verificação da Resposta:** Dependendo da resposta do usuário, a ação apropriada é realizada (remover o item ou cancelar a ação).

✅ Boas práticas de nomenclatura de componentes:

- Button - btn (exemplo: btnConsultar)
- Label - lbl (exemplo: lblNome)
- TextBox - txt (exemplo: txtNome)
- ListBox - lst (exemplo: lstNotas)
- GroupBox - grp (exemplo: grpNotas)

📚 Exercício 1: Calculadora Simples

Enunciado:

Desenvolva uma aplicação Windows Forms em C# que funcione como uma calculadora simples. A aplicação deve permitir ao usuário realizar operações básicas como adição, subtração, multiplicação e divisão.

Requisitos:

1. **Interface do Usuário:**
    - Crie uma interface com botões para os números de `0` a `9`.
    - Adicione botões para as operações `+`, `-`, `*`, `/`, `=`, e `C` (limpar).
    - Inclua um `TextBox` para exibir os números digitados e o resultado.
2. **Funcionalidade:**
    - Quando um botão de `número` é clicado, o número correspondente deve aparecer no TextBox.

- Quando uma `operação` é clicada, a operação deve ser armazenada e o `TextBox` deve ser limpo para a entrada do próximo número.
- Ao clicar no botão `=`, a operação deve ser realizada e o resultado deve ser exibido no TextBox.
- O botão `C` deve limpar o `TextBox` e `resetar qualquer operação em andamento`.

3. **Validação:**
   - Implemente validações para evitar erros como divisão por zero.
   - Garanta que entradas inválidas sejam tratadas adequadamente, exibindo mensagens de erro amigáveis.

📚 Exercício 2: Gerenciamento de Contatos

Enunciado:

Desenvolva uma aplicação Windows Forms em C# para gerenciar contatos. A aplicação deve permitir ao usuário adicionar, editar, remover e listar contatos.

Requisitos:

1. **Interface do Usuário:**
   - Crie uma interface com TextBoxes para os campos `Nome`, `Telefone`, e `Email`.
   - Adicione botões para `Adicionar`, `Editar`, `Remover` e `Listar` contatos.
   - Inclua uma `ListBox` para exibir a lista de contatos.
2. **Funcionalidade:**
   - O botão `Adicionar` deve criar um novo contato e adicioná-lo à lista.
   - O botão `Editar` deve permitir a edição dos detalhes do contato selecionado na ListBox.
   - O botão `Remover` deve excluir o contato selecionado na ListBox.
   - O botão `Listar` deve exibir todos os contatos na ListBox.
3. **Validação:**
   - Verifique se todos os campos estão preenchidos antes de adicionar ou editar um contato.

- Garanta que o formato do email seja válido.
- Implemente mensagens de confirmação para remoção de contatos.

📚 Exercício 3: Sistema de Gerenciamento de Tarefas

Enunciado:

Desenvolva uma aplicação Windows Forms em C# para gerenciar tarefas. A aplicação deve permitir ao usuário adicionar, editar, remover e marcar tarefas como concluídas.

Requisitos:

1. **Interface do Usuário:**
   - Crie uma interface com TextBoxes para o título e a descrição da tarefa.
   - Adicione botões para `Adicionar`, `Editar`, `Remover` e `Marcar como Concluída`.
   - Inclua uma `ListBox` para exibir a lista de tarefas.
2. **Funcionalidade:**
   - O botão `Adicionar` deve criar uma nova tarefa e adicioná-la à lista.
   - O botão `Editar` deve permitir a edição dos detalhes da tarefa selecionada na ListBox.
   - O botão `Remover` deve excluir a tarefa selecionada na ListBox.
   - O botão `Marcar como Concluída` deve alterar o status da tarefa selecionada para concluída.
3. **Validação:**
   - Verifique se todos os campos estão preenchidos antes de adicionar ou editar uma tarefa.
   - Implemente mensagens de confirmação para remoção e conclusão de tarefas.

📚 Exercício 4: Cálculo de Notas de um Aluno

Enunciado:

Desenvolva uma aplicação Windows Forms em C# para calcular as notas de um aluno. A aplicação deve permitir ao usuário inserir as notas de diferentes

disciplinas e calcular a média final do aluno.

Requisitos:

1. **Interface do Usuário:**
   - Crie uma interface com dois `extBox` para os campos `Disciplina` e `Nota`.
   - Adicione botões para `Adicionar Nota`, `Calcular Média` e `Limpar`.
   - Inclua uma `ListBox` para exibir as disciplinas e suas respectivas notas.
   - Adicione um `Label` para exibir a média final.
2. **Funcionalidade:**
   - O botão `Adicionar Nota` deve adicionar a disciplina e a nota à ListBox.
   - O botão `Calcular Média` deve calcular a média das notas inseridas e exibir o resultado no Label - Cálculo da Média (Nota1 + Nota2 + Nota3 + Nota4) / 4. Se média maior igual a 7(sete) aluno aprovado, senão aluno reprovado.
   - O botão `Limpar` deve limpar todos os campos e a ListBox.
3. **Validação:**
   - Verifique se os campos `Disciplina` e `Nota` estão preenchidos antes de adicionar uma nota.
   - Garanta que a nota inserida seja um valor numérico.
   - Implemente mensagens de erro para entradas inválidas e mensagens de confirmação para a adição de notas.

📚 Exercício 5: Cálculo do IMC

Enunciado:

Desenvolva uma aplicação Windows Forms em C# que calcule o Índice de Massa Corporal (IMC) de uma pessoa. A aplicação deve permitir ao usuário inserir seu peso e altura, e calcular o IMC com base nesses valores.

Requisitos:

1. **Interface do Usuário:**

- Crie uma interface com dois `TextBox` para os campos `Peso` (em kg) e `Altura` (em metros).
- Adicione um botão `Calcular IMC`.
- Inclua um `Label` para exibir o resultado do IMC e a classificação correspondente.

2. **Funcionalidade:**
   - O botão `Calcular IMC` deve calcular o IMC com a fórmula: IMC = Peso / (Altura * Altura).
   - Exibir o resultado do IMC no Label junto com a classificação correspondente (Abaixo do peso, Peso normal, Sobrepeso, Obesidade).

`Referência - https://capital.sp.gov.br/web/saude/w/noticias/332991#:~:text=Para obter o IMC%2C basta,%2C5 e 24%2C9.&text=- obesidade mórbida (acima de 40) .`

   - magreza leve (entre 17 e 18,4);
   - magreza moderada (entre 16 e 16,9) ;
   - magreza grave (menor que 16);
   - sobrepeso (índice de 25 a 29,9);
   - obesidade grau 1 (30 a 34,9);
   - obesidade severa (35 a 39,9)
   - obesidade mórbida (acima de 40).

3. **Validação:**
   - Verifique se os campos `Peso` e `Altura` estão preenchidos antes de calcular o IMC.
   - Garanta que os valores inseridos sejam numéricos.
   - Implemente mensagens de erro para entradas inválidas e mensagens de confirmação para o cálculo do IMC.