

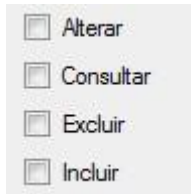
Classes de Componentes Comuns.

Caixas de verificação e botões de opção

Quando você cria a interface do usuário para seu programa, você geralmente precisa de uma maneira para apresentar opções para que o usuário faça a escolha.

Caixas de verificação

O controle CheckBox fornece uma representação visual que facilita criar essas opções.



O status da caixa de seleção pode ser recuperado usando a propriedade **Checked**.

Se a caixa exibe uma marca de seleção, a propriedade retornará True.
Se nenhuma seleção for exibida, a propriedade retorna False.

Botões de opção

Ao contrário das caixas de seleção, botões de opção sempre funcionam como parte de um grupo. Selecionar um botão de opção imediatamente limpa todos os outros botões de opção no grupo. Você pode usar grupos de controles RadioButton para permitir que os usuários escolham entre opções exclusivas.

Por exemplo pode ser utilizado para definir qual é o sexo do usuário.



Da mesma forma que com o CheckBox, você pode verificar a propriedade Checked do RadioButton.

```
string sexo = "";  
if (rbFeminino.Checked)  
{  
    sexo = rbFeminino.Text;  
}  
if (rbMasculino.Checked)  
{  
    sexo = rbMasculino.Text;  
}  
lblSexo.Text = "Sexo: " + sexo;
```

LTPII - Aula 02

Trabalhando com Caixas de Listas

Quando você deseja fornecer aos usuários uma lista de opções, você pode exibir a lista de itens em um controle **ListBox** ou em um controle **ComboBox**.

Controle ComboBox

Por padrão, a caixa de combinação é exibida como um caixa de texto, mas quando os usuários clicam na seta a direita, é exibida uma lista.

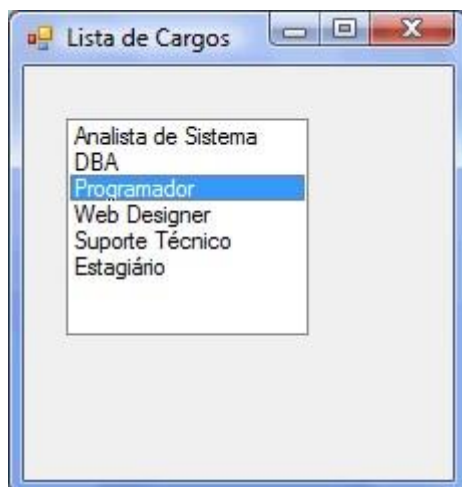
Quando um usuário selecionar um item, ele fica realçado e está visível na exibição padrão, como mostra a imagem abaixo.



Através da propriedade `DropDownStyle`, podemos definir o estilo de trabalho, se funcionará somente para leitura – `DropDownList` ou para Leitura e digitação.

Controle ListBox

Um controle `ListBox` (caixa de lista) permite que você exiba vários itens ao mesmo tempo, permitindo que os usuários percorram esta lista para selecionar o item desejado.



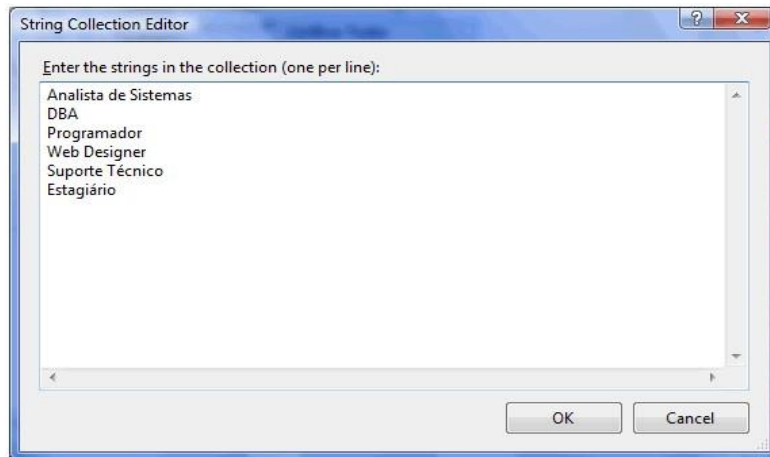
LTPII - Aula 02

Adicionando itens no ListBox e ComboBox

O processos para adicionar itens a caixas de lista e caixas de combinação são semelhantes. Você pode utilizar o String Collection Editor para adicionar os itens.

Clique no ListBox ou ComboBox e na janela properties

No atributo items clique no botão ao lado



Adicionando, excluindo e limpando um ListBox via programação

Método Add Para adicionar um item via programação utilize o método **Add**

Exemplo

```
lbCargo.Items.Add("Gerente de Projeto");
```

Método Remove Para remover um item via programação utilize o método **Remove**

Exemplo

```
lbCargo.Items.RemoveAt(1);
```

Observe que estamos mandando remover o item cujo índice é 1 (no caso o segundo da lista).

Observação

Para obter o índice do item selecionado basta utilizar a propriedade `SelectedIndex`

Método Clear

Para remover todos os itens via programação utilize o método **Clear**

LTPII - Aula 02

Exemplo

```
lbCargo.Items.Clear();
```

Observação

Estes métodos são exatamente os mesmos para a caixa combo.

Verificando se um item na lista existe

Quando você adiciona itens a uma lista, geralmente não deseja duplicar um item existente. Você pode usar o método **Contains** para determinar se o item já está na caixa de combinação ou na caixa de listagem.

Exemplo

```
if (!lbCargo.Items.Contains("Gerente de Projeto"))  
{  
    lbCargo.Items.Add("Gerente de Projeto");  
}
```

Obtendo o item selecionado

Para obter o item selecionado basta utilizar o método `SelectedItem`

```
19 private void lbCargo_SelectedIndexChanged(object sender, EventArgs e)  
20 {  
21     lblCargo.Text = "Cargo: " + lbCargo.SelectedItem.ToString();  
22 }
```

Observação

O evento `SelectedItemChanged` ocorre sempre que o item selecionado é alterado. Para selecionar um único item de cada vez, altere o atributo

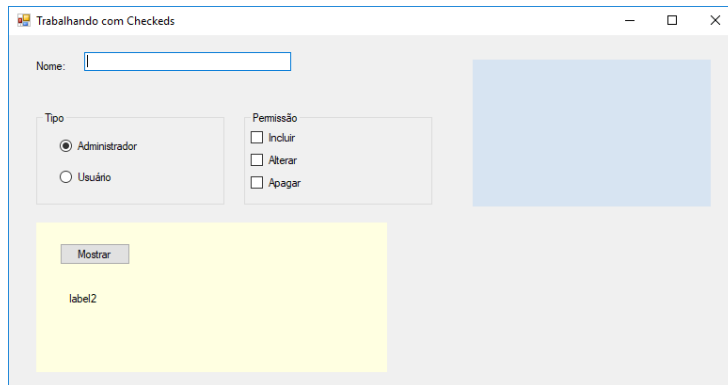
SelectionMode para `one`.

LTPII - Aula 02

Exercícios Resolvidos – Dentro da Solução Aula02 criar os Projetos 1, 2, 3 e 4

Projeto1 – CONFIGURAÇÃO DE USUÁRIO

Definir o nome do usuário, tipo e permissões de acesso. Mostrar as informações em tempo real e permitir também que o usuário veja as informações após a configuração de todos os itens.



Configurar as propriedades Text dos objetos de acordo com a interface e as propriedades Name de acordo com o código abaixo.

```
namespace Projeto1
{
    public partial class frmChecked : Form
    {
        public frmChecked()
        {
            InitializeComponent();
        }

        private void rbAdm_CheckedChanged(object sender, EventArgs e)
        {
            if (rbAdm.Checked == true)
                lblNomeTipo.Text = "Nome:" + txtNome.Text + " - " + rbAdm.Text;
        }

        private void rbUsu_CheckedChanged(object sender, EventArgs e)
        {
            if (rbUsu.Checked == true)
                lblNomeTipo.Text = "Nome:" + txtNome.Text + " - " + rbUsu.Text;
        }

        private void ckIncluir_CheckedChanged(object sender, EventArgs e)
        {
            if (ckIncluir.Checked)
                lblIncluir.Text = "Incluir";
            else
                lblIncluir.Text = "";
        }
    }
}
```

LTPII - Aula 02

```
private void ckAlterar_CheckedChanged(object sender, EventArgs e)
{
    if (ckAlterar.Checked)
        lblAlterar.Text = "Alterar";
    else
        lblAlterar.Text = "";
}

private void ckApagar_CheckedChanged(object sender, EventArgs e)
{
    if (ckApagar.Checked)
        lblApagar.Text = "Apagar";
    else
        lblApagar.Text = "";
}

private void cmdMostrar_Click(object sender, EventArgs e)
{
    string aux = "";
    aux = txtNome.Text;

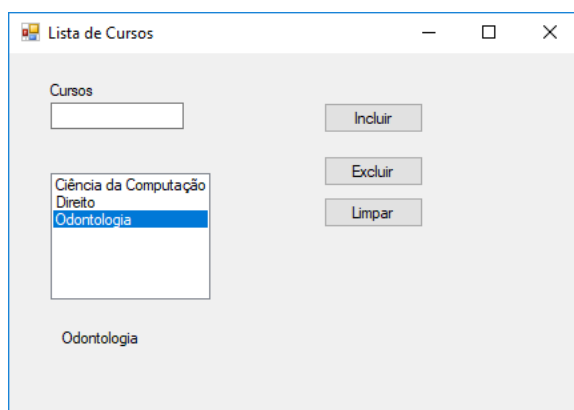
    foreach (RadioButton item in gbTipo.Controls)
    {
        if (item.Checked)
            aux += "\nTipo: " + item.Text;
    }

    foreach (CheckBox item in gbPermissao.Controls)
    {
        if (item.Checked)
            aux += "\nPermissão: " + item.Text;
    }

    lblMostrar.Text = aux;
}
}
```

Projeto2 – TRABALHANDO COM LISTBOX

Permitir que o próprio usuário monte a sua lista de cursos. Para isto crie a seguinte interface:



LTPII - Aula 02

Configurar as propriedades Text dos objetos de acordo com a interface e as propriedades Name de acordo com o código abaixo.

```
namespace Projeto2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (listBox1.SelectedItem != null)
                lblSelecionado.Text = listBox1.SelectedItem.ToString();
            else
                lblSelecionado.Text = "";
        }

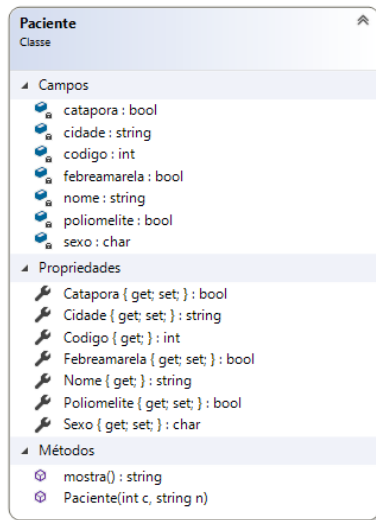
        private void cmdIncluir_Click(object sender, EventArgs e)
        {
            if (listBox1.Items.Contains(txtCurso.Text) == false)
                listBox1.Items.Add(txtCurso.Text);
            else
                MessageBox.Show("Disciplina já existente!");
        }

        private void cmdExcluir_Click(object sender, EventArgs e)
        {
            if (listBox1.SelectedItem != null)
                listBox1.Items.Remove(listBox1.SelectedItem);
        }

        private void cmdLimpar_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
        }
    }
}
```

Projeto3 – TRABALHANDO COM MINHA PRÓPRIA CLASSE PACIENTE

Classe Paciente



```
class Paciente
{
    private int codigo;
    private string nome;
    private char sexo;
    private string cidade;
    private bool febreamarela;
    private bool poliometelite;
    private bool catapora;

    public int Codigo { get => codigo; }
    public string Nome { get => nome; }
    public char Sexo { get => sexo; set => sexo = value; }
    public string Cidade { get => cidade; set => cidade = value; }
    public bool Febreamarela { get => febreamarela; set => febreamarela = value; }
    public bool Poliomelite { get => poliometelite; set => poliometelite = value; }
    public bool Catapora { get => catapora; set => catapora = value; }

    public Paciente(int c, string n)
    {
        codigo = c;
        nome = n;
    }
    public string mostra()
    {
        string msg;
        msg = "Nome: " + Nome + "    Vacinas: ";
        msg += Febreamarela == true ? "Febre Amarela " : "";
        msg += Poliomelite == true ? "Poliometelite " : "";
        msg += Catapora==true?"Catapora ":"";
        return msg;
    }
}
```


LTPII - Aula 02

Interface da classe Paciente

Pacientes

Código Cidade

Nome

Sexo

☐ Masculino

☐ Feminino

Vacinas

☐ Febre Amarela ☐ Poliomelite

☐ Catapora

Número de Pacientes de Alfenas

label1

Número de Pacientes que vacinaram de Catapora

label2

Adicionar

Configurar as propriedades Text dos objetos de acordo com a interface e as propriedades Name de acordo com o código abaixo. Alterar também as propriedades mostrada a seguir:

Properties

rbMasculino System.Windows.Forms.RadioButton

(ApplicationSettings)

(DataBindings)

(Name) rbMasculino

AccessibleDescription

AccessibleName

AccessibleRole Default

AllowDrop False

Anchor Top, Left

Appearance Normal

AutoCheck True

AutoEllipsis False

AutoSize True

BackColor Control

BackgroundImage (none)

BackgroundImageLayout Tile

CausesValidation True

CheckAlign MiddleLeft

Checked True

ContextMenuStrip (none)

Properties

cmbCidade System.Windows.Forms.ComboBox

AutoCompleteCustomS (Collection)

AutoCompleteMode None

AutoCompleteSource None

BackColor Window

CausesValidation True

ContextMenuStrip (none)

Cursor Default

DataSource (none)

DisplayMember (none)

Dock None

DrawMode Normal

DropDownHeight 106

DropDownStyle DropDownList

DropDownWidth 141

LTPII - Aula 02

Código da Interface

```
namespace Projeto3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            Paciente P1 = null;
            List<Paciente> Lista = new List<Paciente>();

            private void cmdAdicionar_Click(object sender, EventArgs e)
            {
                if (txtCodigo.Text == String.Empty || txtNome.Text == String.Empty)
                {
                    MessageBox.Show("Código e Nome são campos obrigatórios");
                }
                else
                {
                    P1 = new Paciente(int.Parse(txtCodigo.Text), txtNome.Text);
                    if (cmbCidade.SelectedItem != null)
                    {
                        P1.Cidade = cmbCidade.SelectedItem.ToString();
                    }
                    if (rbMasculino.Checked == true)
                    {
                        P1.Sexo = 'M';
                    }
                    else
                    {
                        P1.Sexo = 'F';
                    }
                    P1.Febreamarela = ckbFa.Checked;
                    P1.Catapora = ckbCatapora.Checked;
                    P1.Poliomelite = ckbPoli.Checked;
                    Lista.Add(P1);
                    lbPacientes.Items.Add(P1.mostra());

                    foreach (Control item in this.Controls)
                    {
                        if(item is TextBox)
                        {
                            item.Text = "";
                        }
                    }

                    foreach (Control item in panel1.Controls)
                    {
                        if (item is CheckBox)
                        {
                            CheckBox c = (CheckBox)item;
                            c.Checked = false;
                        }
                    }
                }
            }
        }
    }
}
```

LTPII - Aula 02

```
    }

    }

}

private void txtCodigo_KeyPress(object sender, KeyPressEventArgs e)
{
    //Permite 0 a 9 e backspace
    if (!Char.IsDigit(e.KeyChar) && e.KeyChar != (char)8)
    {
        e.Handled = true; // ERRO
    }
}

private void cmdAlfenas_Click(object sender, EventArgs e)
{
    int c = 0;
    foreach (Paciente item in Lista)
    {
        if(item.Cidade == "Alfenas")
        {
            c++;
        }
    }
    lblPesquisa1.Text = c.ToString();
}

private void cmdCatapora_Click(object sender, EventArgs e)
{
    int c = 0;
    foreach (Paciente item in Lista)
    {
        if (item.Catapora == true)
        {
            c++;
        }
    }
    lblPesquisa2.Text = c.ToString();
}
}
```

Projeto4 – TRABALHANDO COM LIST<> , LINQ E LISTBOX

O que é o LINQ?

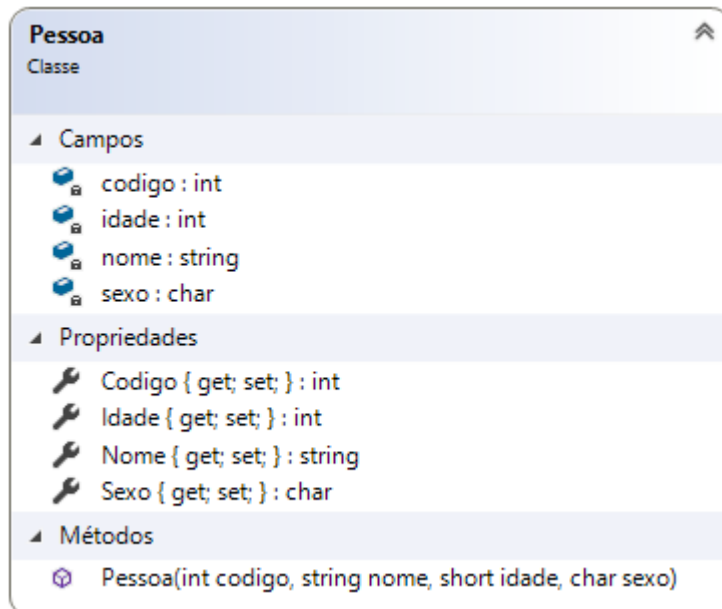
O modelo de programação LINQ ou Language Integrated Query ou linguagem de consulta integrada é um recurso disponível integralmente a partir da versão 3.5 da plataforma .NET que permite realizar consultas de dados em coleções de objetos.

LINQ to Objects : usada para consultar coleções de objetos em memória.

LTPII - Aula 02

O movimento em direção ao LINQ teve início na versão 3.0 da plataforma .NET quando foram introduzidas as coleções genéricas. A linguagem LINQ foi construída com o objetivo de realizar consultas de forma rápida, simples e intuitiva em informações que estão em coleções.

Classe Pessoa

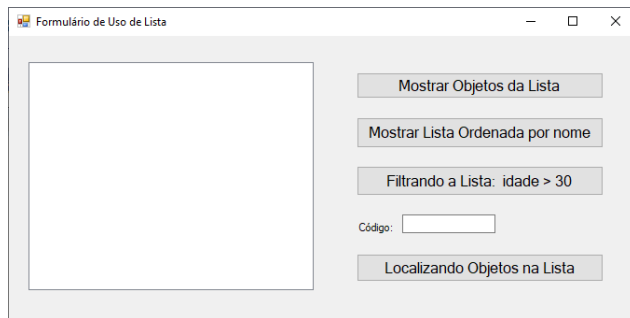


```
class Pessoa
{
    private int codigo;
    private string nome;
    private int idade;
    private char sexo;
    //CTRL R E
    public string Nome { get => nome; set => nome = value; }
    public int Idade { get => idade; set => idade = value; }
    public char Sexo { get => sexo; set => sexo = value; }
    public int Codigo { get => codigo; set => codigo = value; }

    public Pessoa(int codigo, string nome, short idade, char sexo)
    {
        this.codigo = codigo;
        this.nome = nome;
        this.idade = idade;
        this.sexo = sexo;
    }
}
```

LTPII - Aula 02

Interface do Projeto4



Configurar as propriedades Text dos objetos de acordo com a interface e as propriedades Name de acordo com o código abaixo.

Código da Interface

```
namespace Projeto4
{
    public partial class FrmLista : Form
    {
        public FrmLista()
        {
            InitializeComponent();
        }

        private List<Pessoa> pLista = new List<Pessoa>();
        private void carregaLista()
        {
            pLista.Add(new Pessoa(1, "João", 29, 'M'));
            pLista.Add(new Pessoa(2, "Marcos", 35, 'M'));
            pLista.Add(new Pessoa(3, "Americo", 25, 'M'));
            pLista.Add(new Pessoa(4, "Katia", 21, 'F'));
            pLista.Add(new Pessoa(5, "Lena", 33, 'F'));
            pLista.Add(new Pessoa(6, "Suzana", 45, 'F'));
            pLista.Add(new Pessoa(7, "Jim", 38, 'M'));
            pLista.Add(new Pessoa(8, "Jane", 32, 'F'));
            pLista.Add(new Pessoa(9, "Roberto", 31, 'M'));
            pLista.Add(new Pessoa(10, "Cintia", 25, 'F'));
        }

        private void FrmLista_Load(object sender, EventArgs e)
        {
            carregaLista();
        }

        private void cmdMostrar_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            foreach (var item in pLista)
            {
                listBox1.Items.Add(item.Codigo+"-"+item.Nome+"-"+item.Idade+"-"+item.Sexo);
            }
        }
    }
}
```

LTPII - Aula 02

```
private void cmdNome_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    var resultado = from p in pLista orderby p.Nome select p;
    foreach (var c in resultado)
    {
        listBox1.Items.Add(c.Nome);
    }
}

private void cmdIdade_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    var resultado = from c in pLista where c.Idade>30 select c;
    foreach (var c in resultado)
    {
        listBox1.Items.Add(c.Nome + " - " + c.Idade);
    }
}

private void cmdObjeto_Click(object sender, EventArgs e)
{
    int Codigo = int.Parse(txtCodigo.Text);
    listBox1.Items.Clear();
    var resultado = from c in pLista where c.Codigo == Codigo select c;
    foreach (var c in resultado)
    {
        listBox1.Items.Add(c.Codigo + " - " + c.Nome + " - " + c.Idade);
    }
}
}
```