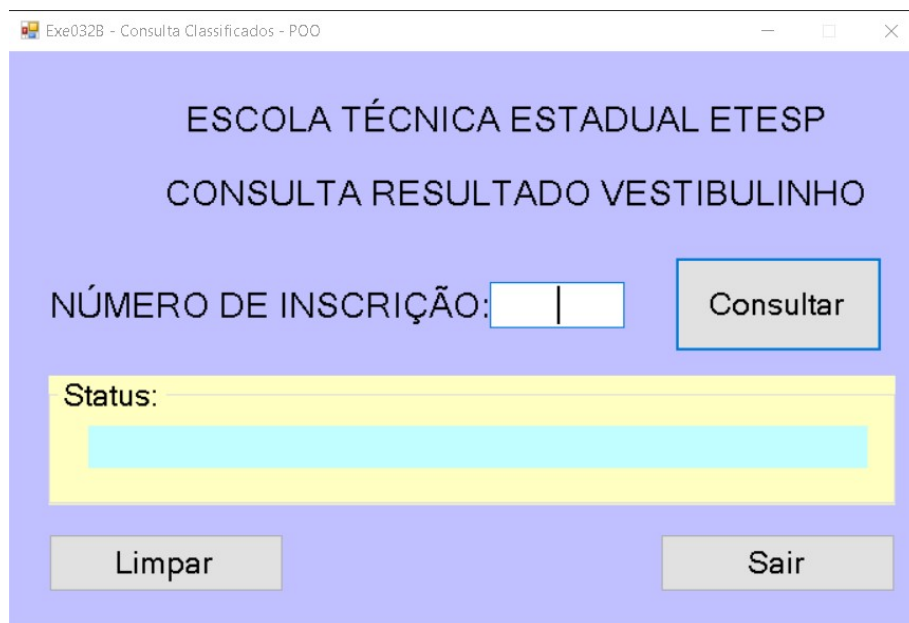


Escola Técnica Estadual “São Paulo” - ETESP

Linguagem C# - Visual Studio 2019

Atividade prática: Vamos reforçar os conceitos estudados até o momento... **EXE032B:** Vamos “revisitar” um projeto já resolvido (Consulta Lista Classificados no Vestibulinho) e “converter” para Programação Orientada a Objetos.

INTERFACE GRÁFICA

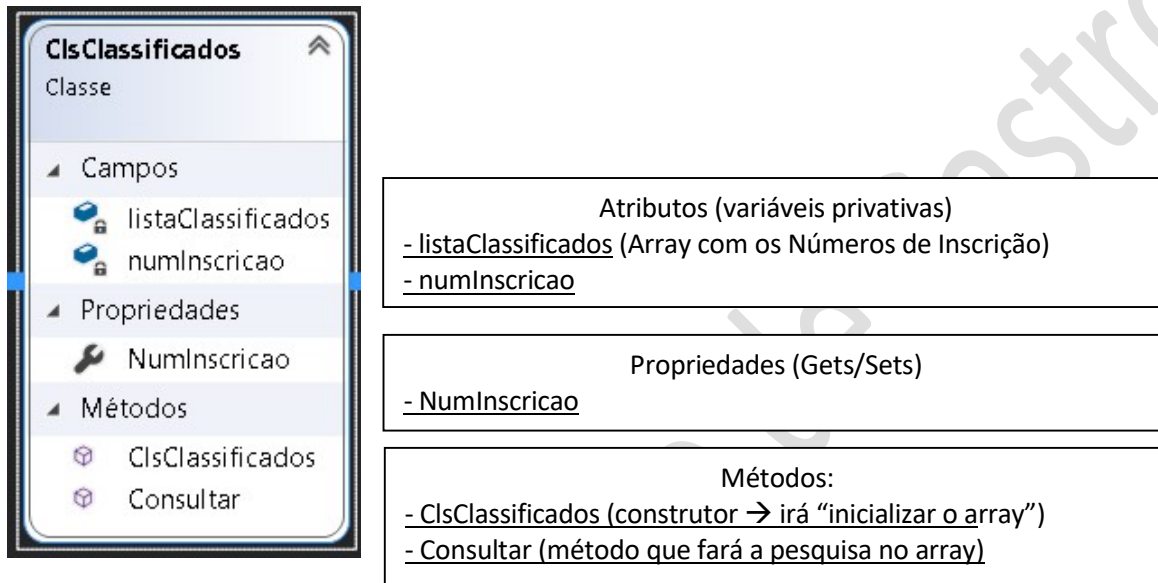


LISTA DE CLASSIFICADOS	
Número de Ordem	(Número de Inscrição)
1	10514
2	30343
3	8240
4	3125
5	50525
6	23289
7	7310
8	9281
9	49524
10	33001

Considerações:

- Se o número de inscrição existir na “lista de classificados”, indica que o candidato foi classificado;
- A “posição” dentro da lista, indica qual é a “classificação” do candidato no vestibulinho..
- Se o número de inscrição NÃO existir na “lista de classificados”, exibir a mensagem “Lista de Espera”.
- Importante destacar que para a solução deste problema NÃO poderemos utilizar diretamente o “número de inscrição” como índice do array.

Diagrama de classe:



Principais etapas:

1. Desenho da interface gráfica
2. Criação e programação da classe “ClsClassificados”:
 - a) Declaração dos atributos/campo: Variáveis “privativas” da classe.
 - b) Declaração dos métodos gets/sets: São as “propriedades visíveis ao mundo externo”. Funcionam como um “elo de ligação” entre as variáveis privativas da classe (atributos) e o programa “cliente”. São elas que recebem e enviam dados.
 - c) Declaração do construtor: O construtor é um método “especial” utilizado para inicializar os objetos da classe quando estes são criados (instanciados). Este método possui o mesmo nome da Classe e não tem nenhum tipo de retorno, nem mesmo void. Para este projeto, o “construtor” irá preencher o array com a lista dos classificados!
 - d) Declaração do método (procedimentos) Consultar → Fará uma pesquisa no array (Array.IndexOf) e irá retornar um texto, informando se o candidato está classificado ou não.
3. Programação do “botão” consultar (formulário principal).

SOLUÇÃO

Classe:

```
using System;

namespace Exe025
{
    public class ClsClassificados
    {
        //Declaração dos Atributos
        int numInscricao;

        //Declaração/criação GETS/SETS
        public int NumInscricao { get => numInscricao; set => numInscricao = value; }

        //Declaração do Array - Lista de Classificados
        int[] listaClassificados = new int[10];

        //Declaração do método construtor e o preenchimento do Array
        public ClsClassificados()
        {
            listaClassificados[0] = 10514;
            listaClassificados[1] = 30343;
            listaClassificados[2] = 8240;
            listaClassificados[3] = 3125;
            listaClassificados[4] = 50525;
            listaClassificados[5] = 23289;
            listaClassificados[6] = 7310;
            listaClassificados[7] = 9281;
            listaClassificados[8] = 49524;
            listaClassificados[9] = 33001;
        }

        //Declaração do método Consultar (retorna um texto)
        public string Consultar()
        {
            //localiza a posição do candidato no array. Se não existir, índice será = -1
            //Importante destacar que o "numInscricao" que o método utiliza
            //é o atributo privativo da classe!!
            int posicao = Array.IndexOf(listaClassificados, numInscricao);

            if (posicao < 0) //Inscrição não localizada
            {
                return "Lista de espera!!";
            }
            else
            {
                posicao += 1; //pois o array inicia em ZERO
                return "Parabéns!! Você está classificado na " + posicao.ToString() + " a. posição!!";
            }
        }
    }
}
```

Programa principal:

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

namespace Exe025
{
    public partial class FrmExe025 : Form
    {
        public FrmExe025()
        {
            InitializeComponent();
        }

        private void BtnSair_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void BtnLimpar_Click(object sender, EventArgs e)
        {
            TxtNumero.Text = "";
            LblResultado.Text = "";
            TxtNumero.Focus();
        }

        //Esta solução utiliza o método INDEXOF do Array para localizar a inscrição
        private void BtnConsultar_Click(object sender, EventArgs e)
        {
            try
            {
                int numInscricao = Convert.ToInt32(TxtNumero.Text);

                //Gera a instância da classe
                ClsClassificados objClassificados = new ClsClassificados();

                //
                //Transfere o Número de inscrição (do formulário) para
                //a classe --> propriedade NumInscricao
                //
                objClassificados.NumInscricao = numInscricao;

                //Chama o método para verificar classificação, que retorna uma string
                string resultado = objClassificados.Consultar();

                LblResultado.Text = resultado;
            }
            catch
            {
                MessageBox.Show("Digite somente números!!!", "ATENÇÃO");
                TxtNumero.Text = "";
                TxtNumero.Focus();
            }
        }
    }
}
```

Overload

Overload significa sobrecarga de métodos. Métodos com os mesmos nomes, porém com “assinatura diferente”. **OBS:** Assinatura de um método é como ele foi declarado (se recebe parâmetros; se retorna com valores...), seu nome completo, nome e sobrenome (parâmetros).

Exercício proposto: EXE033 - Proposta de Orçamento

INTERFACE GRÁFICA

Exe033 - ORÇAMENTO - OOP

TACO LASCADO, SUJO E SOLTO
Manutenção e Limpeza de Pisos e Azulejos

Orçamento

Valor R\$: Num. Parcelas* 1

* 1 = A Vista com 15% desconto:

Proposta de Pagamento

Execução do projeto:

Exe033 - ORÇAMENTO - OOP

TACO LASCADO, SUJO E SOLTO
Manutenção e Limpeza de Pisos e Azulejos

Orçamento

Valor R\$: 1000 Num. Parcelas* 1

* 1 = A Vista com 15% desconto:

Proposta de Pagamento

O valor do orçamento com desconto a vista é de R\$850,00

Exe033 - ORÇAMENTO - OOP

TACO LASCADO, SUJO E SOLTO
Manutenção e Limpeza de Pisos e Azulejos

Orçamento

Valor R\$: 1000 Num. Parcelas* 3

* 1 = A Vista com 15% desconto:

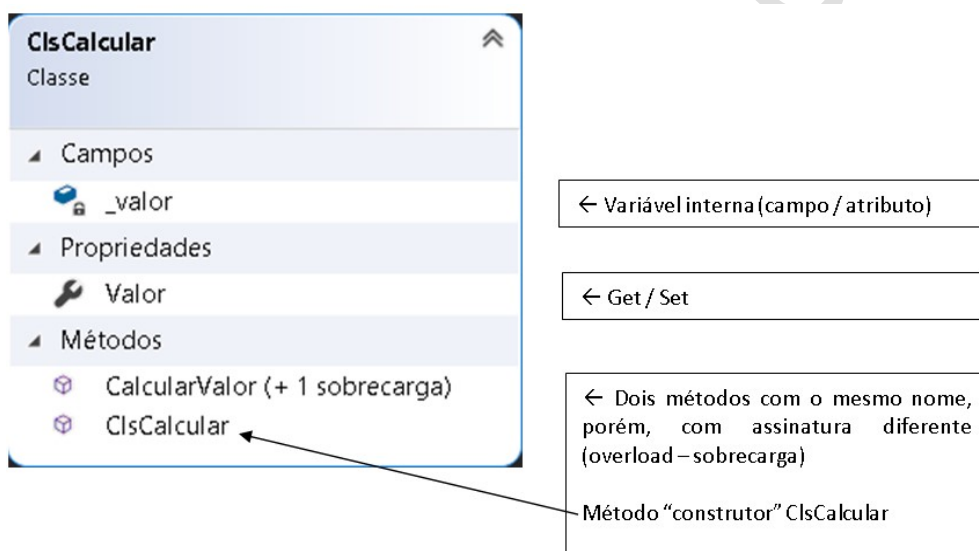
Proposta de Pagamento

O valor será dividido em 3 parcelas de R\$333,33

Principais Etapas:

- Incluir a “classe” no projeto (Project → Add Class). Nome da Classe: ClsCalcular
- Declaração dos atributos/campo: Variáveis “privativas” da classe.
- Declaração dos métodos gets/sets: São as “propriedades visíveis ao mundo externo”. Funcionam como um “elo de ligação” entre as variáveis da classe e o programa “cliente”. São elas que recebem e enviam dados.
- Declaração do construtor: O construtor é um método utilizado para inicializar os objetos da classe quando estes são criados (instanciados). Este método possui o mesmo nome da Classe e não tem nenhum tipo de retorno, nem mesmo void.
- Declaração dos métodos (procedimentos): São as ações que nossos objetos podem executar.

Veja o “diagrama da Classe”:



Programação da Classe "ClsCalcular"

```
namespace Exe033_OOP_Orcamento
{
    class ClsCalcular
    {
        //Declaração do atributo / campo
        decimal _valor = 0;

        //Declaração da propriedade
        public decimal Valor { get => _valor; set => _valor = value; }

        //Método construtor
        public ClsCalcular()
        {
            _valor = 0;
        }

        //Este é um exemplo para OverLoad (sobrecarga de métodos).
        // Métodos com o mesmo nome, porém, com assinatura diferente!!!

        //Método para parcela a vista
        public string CalcularValor()
        {
            decimal valorParcela = (_valor * 85) / 100;
            string texto = "O valor do orçamento com desconto a vista é de " + valorParcela.ToString("C2");
            //"C2" --> exibe o número no formato de dinheiro com duas casas decimais

            return texto;
        }

        //Método para pagamento parcelado. Mesmo nome, porém, com "assinatura diferente" (recebe
        //parâmetro)
        public string CalcularValor(int numParcela)
        {
            decimal valorParcela = _valor / numParcela;

            string texto = "O valor será dividido em " + numParcela + " parcelas de " +
                valorParcela.ToString("C2");
            //"C2" --> exibe o número no formato de dinheiro com duas casas decimais

            return texto;
        }
    }
}
```

Programação do “projeto principal”

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

namespace Exe033_OOP_Orcamento
{
    public partial class FrmExe033 : Form
    {
        public FrmExe033()
        {
            InitializeComponent();

            private void FrmExe033_Load(object sender, EventArgs e)
            {
                //Preenche o ComboBox com as formas de pagamento
                for (int x = 1; x <= 6; x++)
                {
                    CboFormaPagto.Items.Add(x); ;
                }

                CboFormaPagto.SelectedIndex = 0;
            }

            private void BtnSair_Click(object sender, EventArgs e)
            {
                Application.Exit();
            }

            private void BtnNovo_Click(object sender, EventArgs e)
            {
                LblResultado.Text = "";
                TxtValor.Text = "";
                CboFormaPagto.SelectedIndex = 0;
                TxtValor.Focus();
            }

            private void BtnConsultar_Click(object sender, EventArgs e)
            {
                //Cria a instância da Classe. Neste momento o método construtor é executado
                ClsCalcular ObjCalcular = new ClsCalcular();

                //Transfere o "valor" para a classe
                ObjCalcular.Valor = Convert.ToDecimal(TxtValor.Text);

                if (CboFormaPagto.SelectedIndex == 0)
                {
```


Prof. Roberto de Castro

//Executa o método CalcularValor, SEM PARAMETRO, para desconto a vista, que retorna um texto
LblResultado.Text = ObjCalculador.CalcularValor();

}
else

{

//Executa o método CalcularValor, COM PARAMETRO, para pagamento parcelado, que retorna um texto

LblResultado.Text = ObjCalculador.CalcularValor(CboFormaPagto.SelectedIndex+1);

}

}

}

}

Perceba, ao iniciar a digitação do nome do método, o aviso de sobrecarga.

