

Escola Técnica Estadual “São Paulo” - ETESP

Linguagem C# - Visual Studio 2019

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:

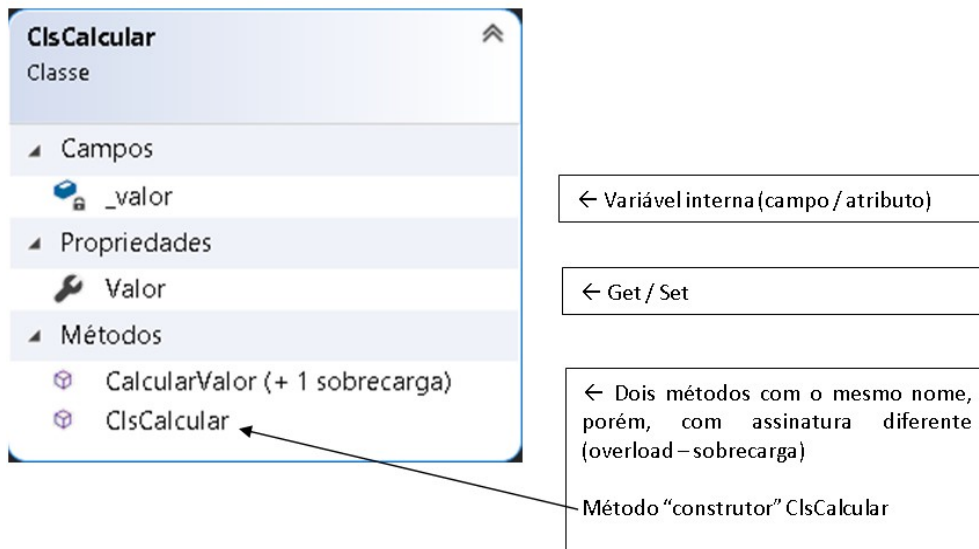
Proposta de Pagamento

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 (formulário)"

```
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();
}

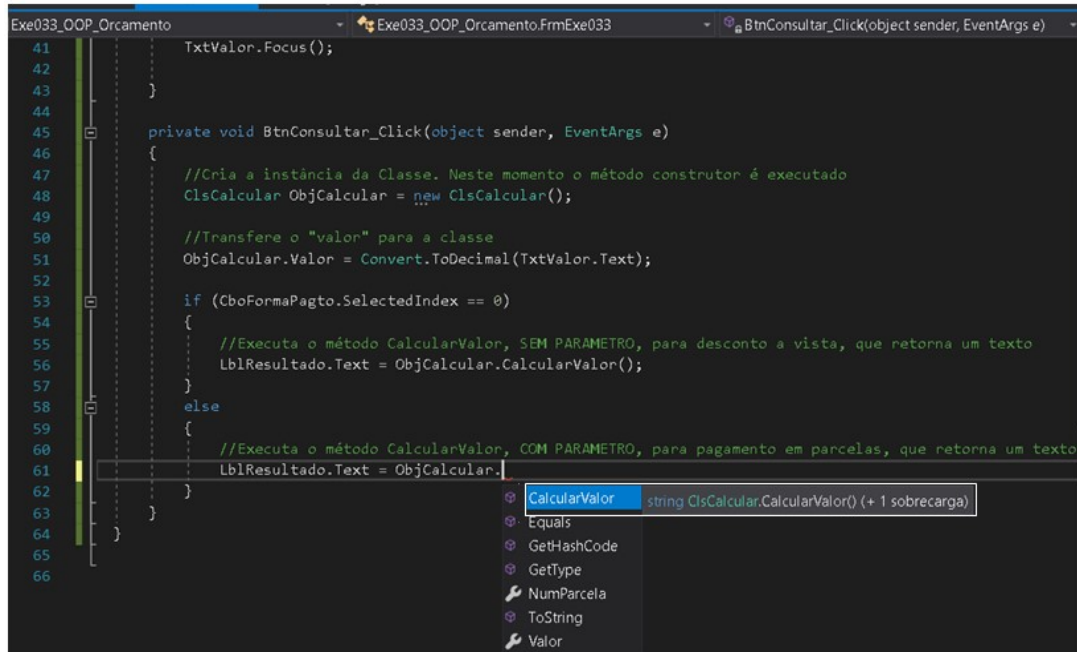
private void BtnNovo_Click(object sender, EventArgs e)
{
    LblResultado.Text = "";
    TxtValor.Text = "";
    CboFormaPagto.SelectedIndex = 0;
    TxtValor.Focus();
}

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)
    {
        //Executa o método CalcularValor, SEM PARAMETRO, para desconto a vista, que retorna um texto
        LblResultado.Text = ObjCalcular.CalcularValor();
    }
    else
    {
        //Executa o método CalcularValor, COM PARAMETRO, para pagamento parcelado, que retorna
        um texto
        LblResultado.Text = ObjCalcular.CalcularValor(CboFormaPagto.SelectedIndex+1);
    }
}
}
```

Perceba o aviso de sobrecarga que é exibido, ao iniciar a digitação do nome do método:



```
41 TxtValor.Focus();
42 }
43
44
45 private void BtnConsultar_Click(object sender, EventArgs e)
46 {
47     //Cria a instância da Classe. Neste momento o método construtor é executado
48     ClsCalcular ObjCalcular = new ClsCalcular();
49
50     //Transfere o "valor" para a classe
51     ObjCalcular.Valor = Convert.ToDecimal(TxtValor.Text);
52
53     if (CboFormaPagto.SelectedIndex == 0)
54     {
55         //Executa o método CalcularValor, SEM PARAMETRO, para desconto a vista, que retorna um texto
56         LblResultado.Text = ObjCalcular.CalcularValor();
57     }
58     else
59     {
60         //Executa o método CalcularValor, COM PARAMETRO, para pagamento em parcelas, que retorna um texto
61         LblResultado.Text = ObjCalcular.CalcularValor();
62     }
63 }
64
65
66
```

6.4 - Herança e Overrind (sobrescrita):

Herança → Pode ser definida como a capacidade de uma classe “herdar” atributos e comportamentos de uma outra classe. É utilizado quando necessitamos de uma “NOVA” classe, que tenha todas as características de outra classe, com algumas modificações em seu comportamento, ou mesmo algumas funcionalidades adicionais.

Overrind (sobrescrita): Substituir o método existente da classe pai (classe base), por um novo método escrito na classe filha.

IMPORTANTE: Para ocorrer a sobrescrita, precisamos aplicar o conceito de Herança!!!

***** Você só pode usar o override em métodos que permitam serem sobrescritos!!! *****

A proposta agora é “aproveitarmos” a classe já existente (ClsOperacao) para o próximo exercício. Vamos fazer uma cópia do projeto anterior “Somar dois valores” para não perdermos o que já está pronto.

Nosso projeto já possui a classe “ClsOperacao” que faz a soma entre dois valores.

Neste novo projeto incluiremos uma nova classe: ClsOperacoes, que:

- 1) Herdará todos os atributos(campos), propriedades e métodos da classe “ClsOperacao”.
- 2) Irá “reescrever” o método “Somar” (exibirá uma mensagem, informando que está executando este novo método).
- 3) Terá “novos” métodos para efetuar a Subtração, Multiplicação e Divisão.

Notação UML das classes:

ClsOperacao

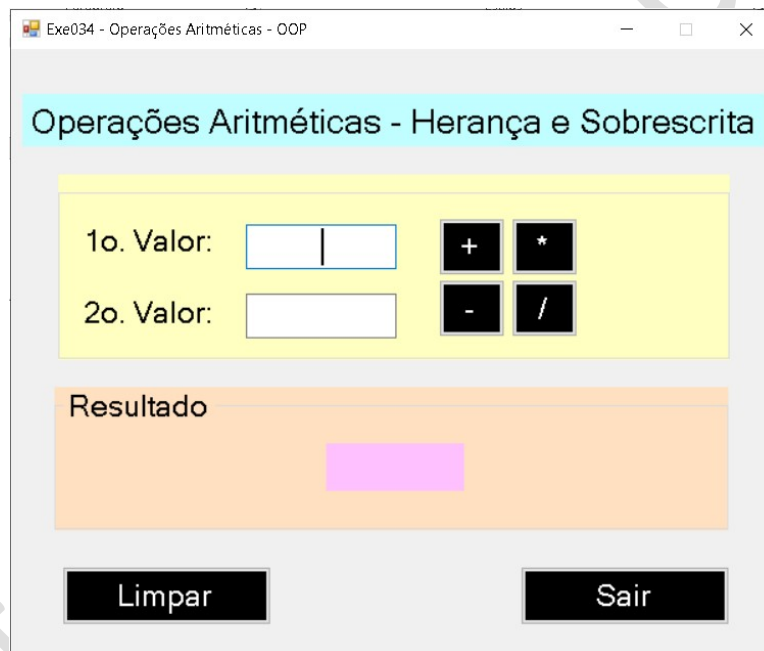
Prof. Roberto de Castro

-valor1: int
-valor2: int
+ somar(valor1:int, valor2) int



ClsCalculadora
+ somar() int + subtrair() int + multiplicar() int + dividir() int

Interface gráfica:



Programação da classe ClsOperacao (com MODIFICAÇÕES):

```
class ClsOperacao
{
    //Declaração dos atributos (variáveis)e dos gets e sets
    private int _valor1;
    private int _valor2;

    //Declaração das propriedades Gets / Sets
    public int Valor1 { get => _valor1; set => _valor1 = value; }
    public int Valor2 { get => _valor2; set => _valor2 = value; }
```

Prof. Roberto de Castro

```
//Declaração do Método Construtor - este método possui o mesmo nome da classe e NÃO possui
//VOID
//Irá inicializar os atributos - variáveis privativas
public ClsOperacao()
{
    _valor1 = 0;
    _valor2 = 0;
}

//Declaração do método que irá efetuar a soma entre os dois valores e devolver o resultado
//O termo "virtual" é necessário para que ocorra a sobrescrita. Este método será reescrito em
//nova classe
public virtual int Somar()
{
    return _valor1 + _valor2;
}
}
```

Programação da Classe "ClsCalculos" (Que herda as propriedades e métodos da classe ClsOperacao)

```
using System.Windows.Forms;

namespace Exe032_OOP_Somar2Valores
{
    //Observe que a classe ClsCalculos está "herdando" todos os métodos e as propriedades da classe
    //ClsOperacao
    class ClsCalculos: ClsOperacao
    {
        //Construtor
        public ClsCalculos()
        {
            //Será executado o construtor da classe base
        }

        //Este método está fazendo a sobrescrita do método Somar existente na classe ClsOperacao
        public override int Somar()
        {
            MessageBox.Show ("É o método somar da classe ClsCalculos que está em execução!!");
            return Valor1 + Valor2;
        }

        //Método para calcular a subtração e retornar o resultado
        public int Subtrair()
        {
            return Valor1 - Valor2;
        }
    }
}
```

//Método para calcular a multiplicação e retornar o resultado

```
public int Multiplicar()
{
    return Valor1 * Valor2;
}
```

//Método para calcular a divisão e retornar o resultado

```
public int Dividir()
{
    return Valor1 / Valor2;
}
```

```
}
```

Programação do Formulário:

```
using System;
```

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

```
namespace Exe032_OOP_Somar2Valores
```

```
{
```

```
    public partial class FrmExe034 : Form
```

```
    {
```

```
        public FrmExe034()
```

```
        {
```

```
            InitializeComponent();
```

```
        }
```

```
        private void BtnSair_Click(object sender, EventArgs e)
```

```
        {
```

```
            Application.Exit();
```

```
        }
```

```
        private void BtnLimpar_Click(object sender, EventArgs e)
```

```
        {
```

```
            TxtValor1.Text = "";
```

```
            TxtValor2.Text = "";
```

```
            LblResultado.Text = "";
```

```
            TxtValor1.Focus();
```

```
        }
```

//Este procedimento é executado pelos botões Soma, Subtração, Multiplicação e Divisão

```
        private void BtnCalcular_Click(object sender, EventArgs e)
```

```
        {
```

```
            //Gera a instância da classe
```

```
            ClsCalculos ObjOperacao = new ClsCalculos();
```

```
            ObjOperacao.Valor1 = Convert.ToInt32(TxtValor1.Text);
```

```
            ObjOperacao.Valor2 = Convert.ToInt32(TxtValor2.Text);
```



```
int resultado = 0;

Button botaoOperacao = sender as Button;

if (botaoOperacao.Text == "+")
{
    //Executa o "novo" método Somar da classe ClsCalculos
    resultado = ObjOperacao.Somar();
}
else if (botaoOperacao.Text == "-")
{
    //Executa o método Subtrair
    resultado = ObjOperacao.Subtrair();
}
else if (botaoOperacao.Text == "*")
{
    //Executa o método Multiplicar
    resultado = ObjOperacao.Multiplicar();
}
else
{
    //Executa o método Dividir
    resultado = ObjOperacao.Dividir();
}

//Exibe o resultado da operação
LblResultado.Text = resultado.ToString();
}
}
```