

**UNIVERSIDADE FEDERAL DE SANTA MARIA**  
**CENTRO DE TECNOLOGIA**  
**DEPARTAMENTO DE PROCESSAMENTO DE ENERGIA ELÉTRICA**



# **Revisão lógica programação usando C# 1**

---

Curso de Engenharia de Controle e Automação  
DPEE1090 - Programação orientada a objetos para automação

---

Prof. Renan Duarte

1º semestre de 2024

# Aviso

---

## Repositório disciplina

- Criado repositório no GitHub onde serão disponibilizados os materiais de cada aula
- Material também será postado no Moodle



[https://github.com/duarterr/DPEE1090\\_POO\\_Automacao/](https://github.com/duarterr/DPEE1090_POO_Automacao/)

Git básico: <https://www.freecodecamp.org/portuguese/news/tutorial-de-git-e-github-controle-de-versao-para-iniciantes/>

Solução de problemas: [https://ohshitgit.com/pt\\_br/swears/](https://ohshitgit.com/pt_br/swears/)

# Sumário

---

## Revisão lógica programação usando C#

- Introdução à linguagem de programação C# (.NET, *assembly*, *namespace*, etc.)
- Convenções de nomenclatura em C#
- Tipos de dados (tipo valor e tipo referência)
- Entrada e saída de dados

# C# e .NET

---

## O que são?

### C#

- Uma linguagem de programação (regras sintáticas)

### .NET (2002)

- Uma plataforma de desenvolvimento para se criar diversos tipos de aplicações, podendo usar várias linguagens de programação



# .NET

---

## Composição

### BCL - *Base Class Library*

- Biblioteca básica de classes → Definições e métodos já implementados (tipos de dados, ParseInt, ToString...)
- <https://learn.microsoft.com/pt-br/dotnet/standard/class-library-overview>

### CLR - *Common Language Runtime*

- Máquina Virtual que executa os códigos desenvolvidos em .NET
- Possui *garbage collection*
- Similar ao JVM do Java
- <https://learn.microsoft.com/pt-br/dotnet/standard/clr>

# .NET

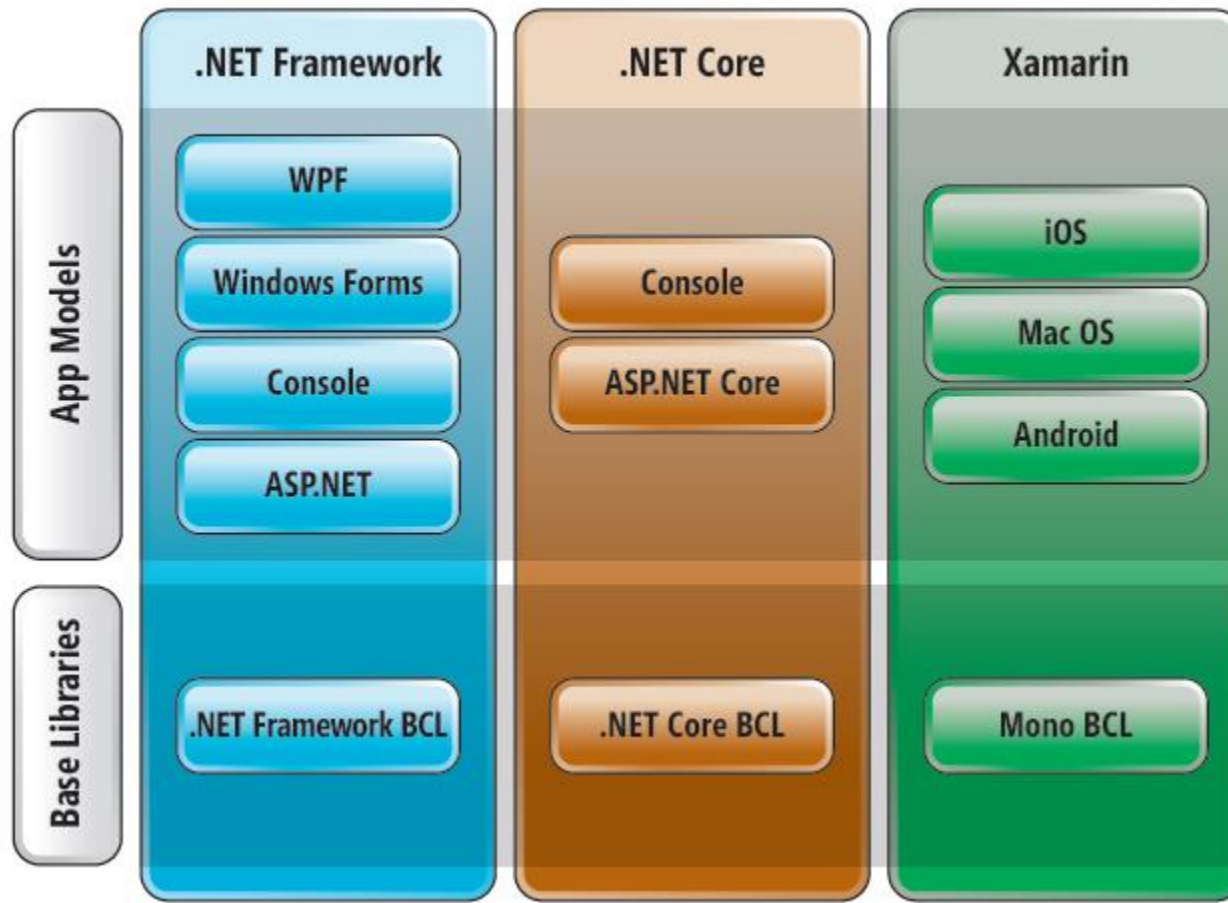
## Implementações

.NET é apenas uma especificação que define quais recursos uma implementação do .NET deve conter



# .NET

## Implementações



# Modelo de execução

---

## Compilação e interpretação

### Linguagens compiladas

- C
- C++

### Linguagens interpretadas

- PHP
- JavaScript

### Linguagens pré-compiladas + máquina virtual

- Java,
- C#



# Modelo de execução

## Compilação

```
#include <iostream>

int main() {
    double x, y, average;

    cout << "Enter first number: "; cin >> x;
    cout << "Enter second number: "; cin >> y;
    average = (x + y) / 2.0;
    cout << "Average = " << average << endl; return 0;
}
```

Compilador 1

Compilador 2

Compilador 3

Executável (.exe)

Windows

Hardware

Executável (.dmg)

MacOS

Hardware

Executável (.elf)

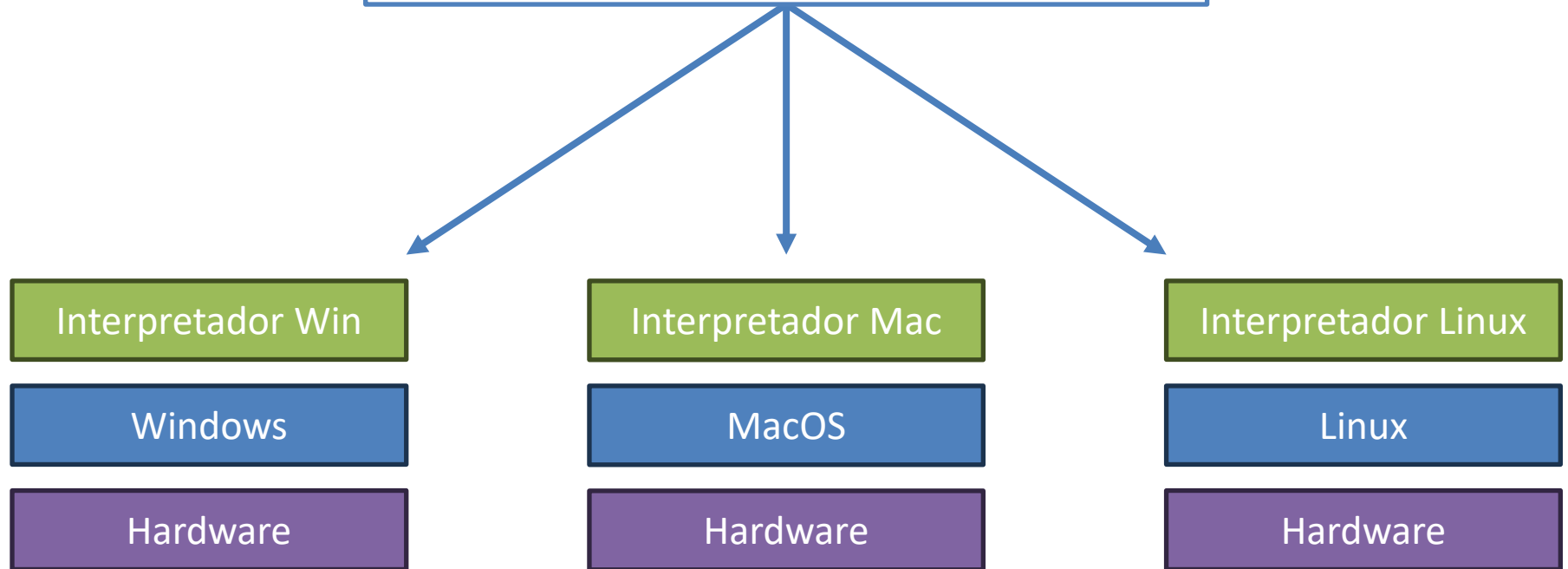
Linux

Hardware

# Modelo de execução

## Interpretação

```
<?php
  print "Enter first number: ";
  $x = trim(fgets(STDIN)); print "Enter second
  number: ";
  $y = trim(fgets(STDIN));
  $average = ($x + $y) / 2; print "Average =
  $average";
?>
```



# Modelo de execução

## Híbrido

```
using System;

namespace Course { class Program {
    static void Main(string[] args) {
        double x, y, average;
        Console.WriteLine("Enter first number: ");
        x = int.Parse(Console.ReadLine());
        Console.WriteLine("Enter second number: ");
        y = int.Parse(Console.ReadLine());
        average = (x + y) / 2.0;
        Console.WriteLine("Average = " + average);
    }
}
```

Compilador

Bytecode

Common Intermediate Language (CIL)

.NET CLR Windows

Windows

Hardware

.NET CLR Mac

MacOS

Hardware

.NET CLR Linux

Linux

Hardware

# Modelo de execução

## Resumo C#

```
using System;

namespace Course { class Program {
    static void Main(string[] args) {
        Console.WriteLine("Hello world!");
    }
}
```

*Pré-compilação*

Compilador

Common Intermediate  
Language (CIL)

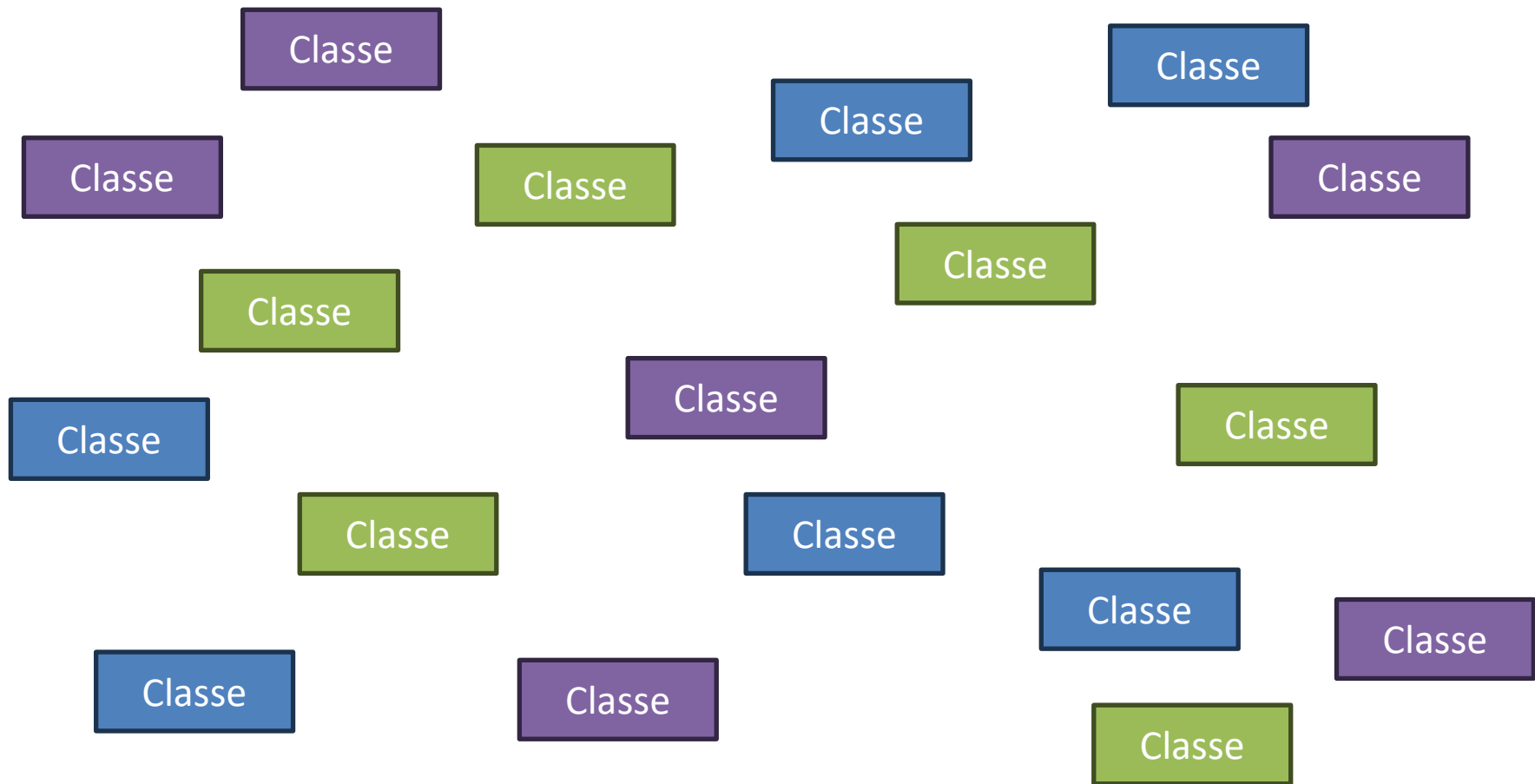
*Compilação just-in-time (JIT)*  
Muito mais rápido que a interpretação

*.NET Common Language Runtime (CLR)*  
Específica ao sistema operacional

Código de máquina

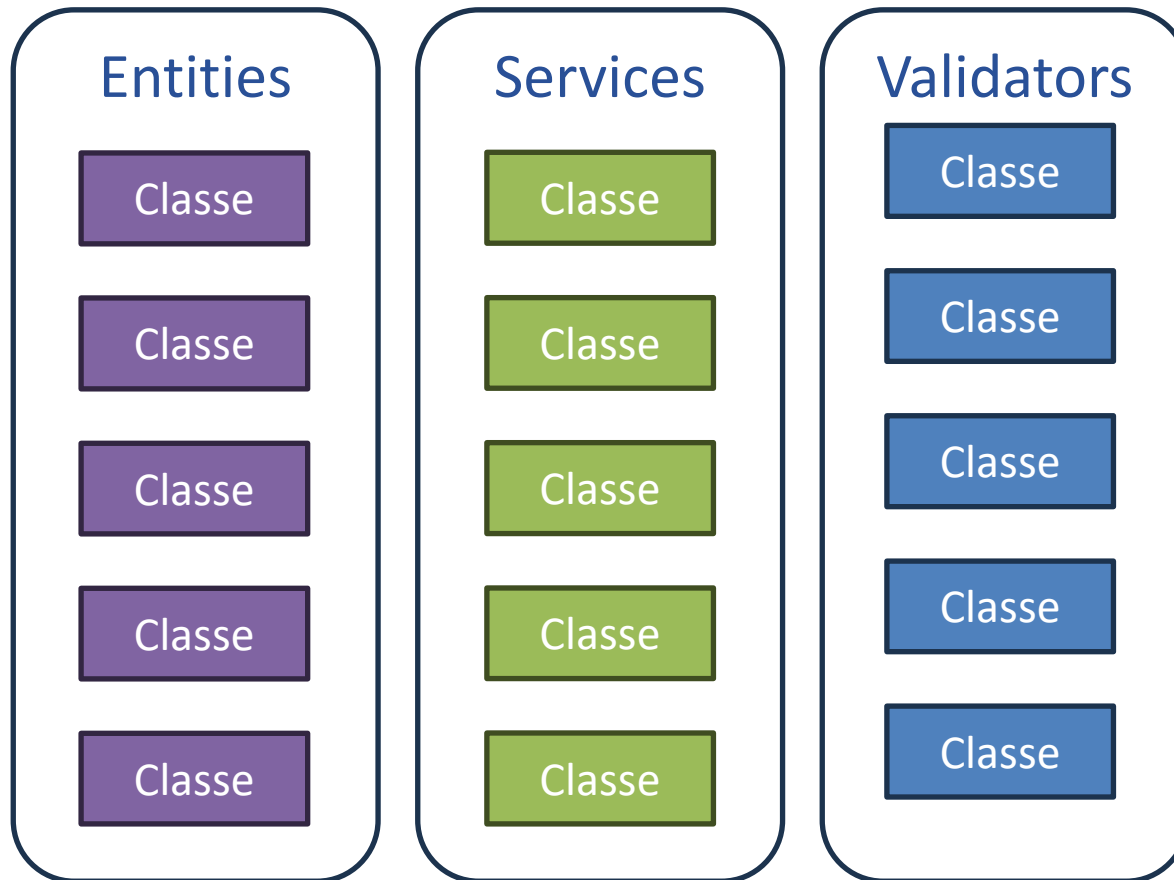
# Estrutura de uma aplicação C# .NET

Uma aplicação é composta por classes



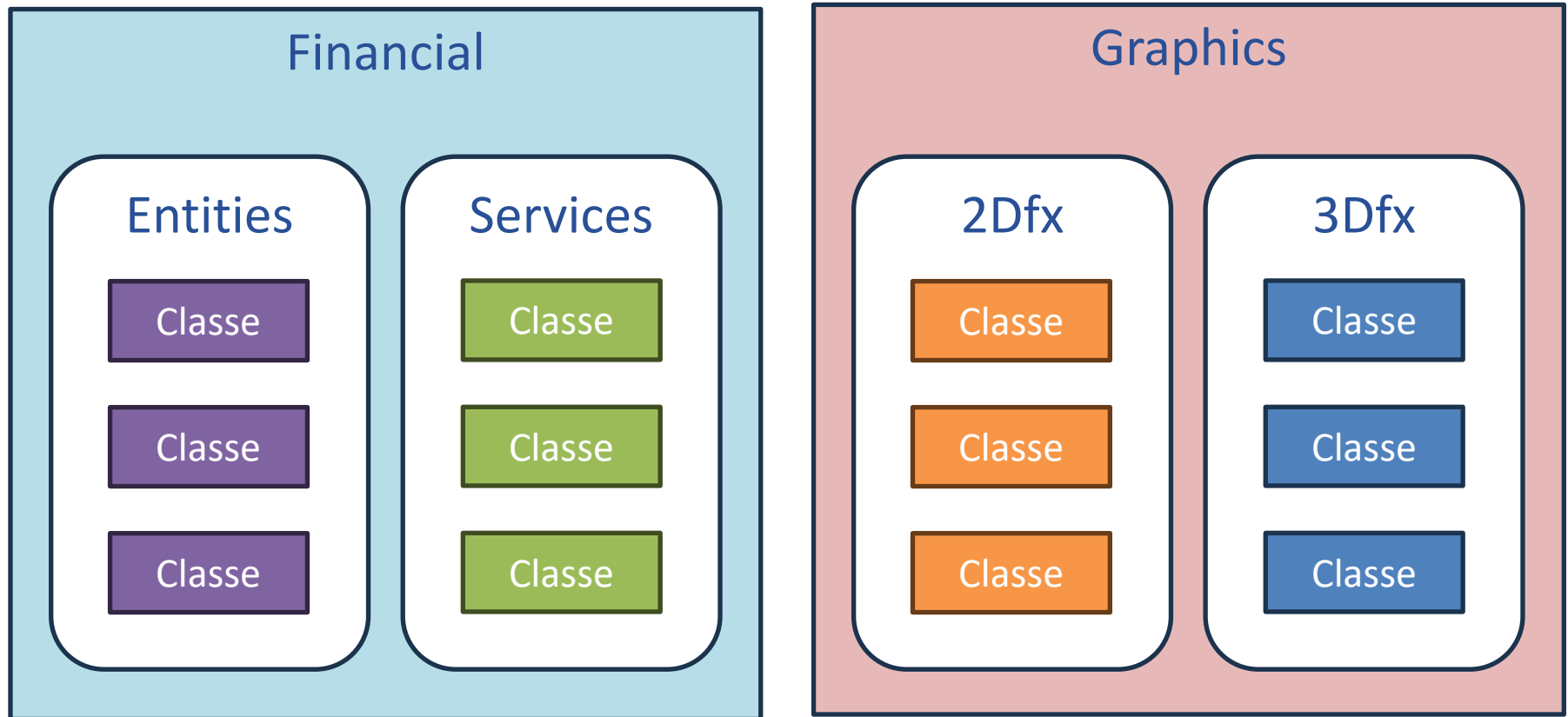
# Estrutura de uma aplicação C# .NET

Um *namespace* é um agrupamento lógico de classes relacionadas



# Estrutura de uma aplicação C# .NET

Um *assembly* é um agrupamento lógico de classes relacionadas  
DLL ou EXE

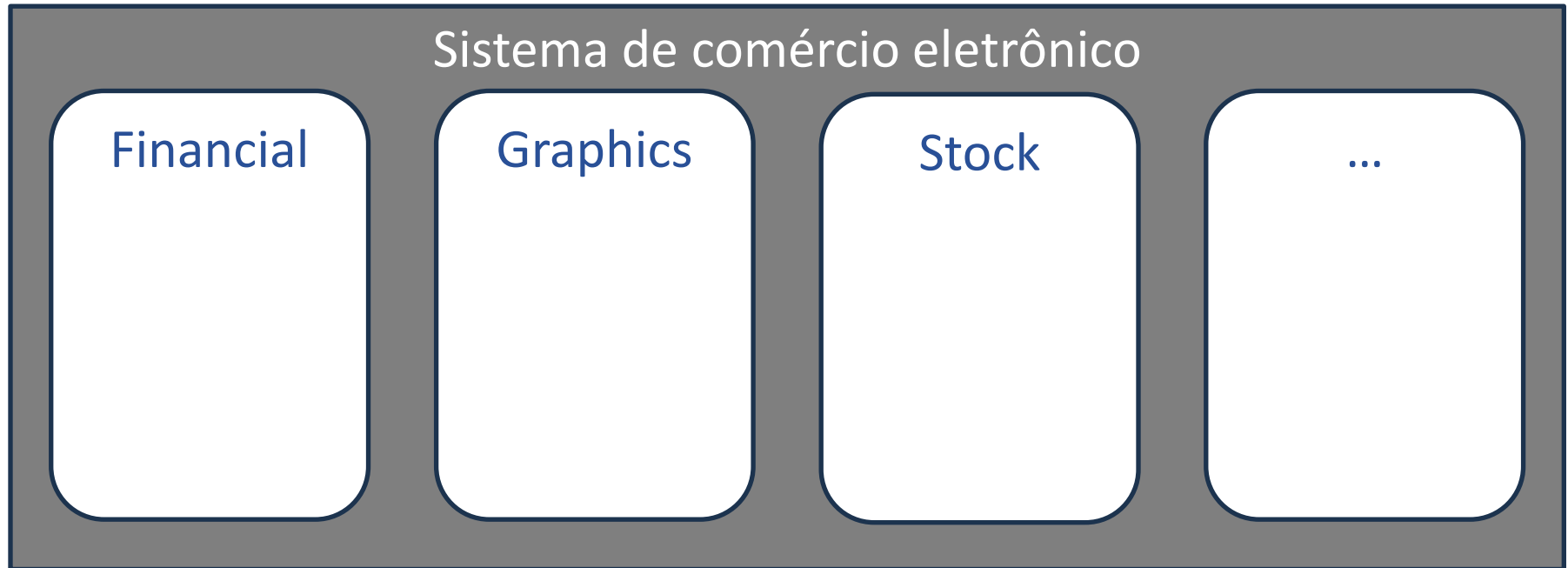


# Estrutura de uma aplicação C# .NET

Uma *aplicação* é um agrupamento de *assemblies* relacionados

No VisualStudio:

- Aplicação → Solution
- Assembly → Projects





# Introdução ao C#

---

## Criando um projeto no VisualStudio Community

- Clicar em “Criar novo projeto” (ou Arquivo → Novo → Projeto)
- Selecionar “Aplicativo do console”
- Dar um nome ao projeto (simples e sem caracteres especiais)
- Escolher onde salvar
- Clicar em “Próximo”
- Selecionar versão do .NET (ex. 8.0)
- Clicar em “Criar”

# Introdução ao C#

---

## Dicas do VisualStudio Community

### Troca do idioma

- Tools → Options → Environment → International Settings → Language

### Como fechar e reabrir o projeto

- Abra o arquivo .sln

### Indentação automática

- CTRL + K + D

### Quebra de linha nas chaves

- Tools → Options → Text Editor → C# → Code Estiling → Formatting → New Lines

# Introdução ao C#

---

## Convenções de nomenclatura

camelCase

- nomeDoUsuario

snake\_case

- nome\_do\_usuario

PascalCase ou UpperCamelCase

- NomeDoUsuario

UPPERCASE

- TAXA\_DE\_JUROS

lowercase:

- nome, contador

# Introdução ao C#

## Convenções de nomenclatura no C#



<https://bit.ly/3xgsgYo>

- Na dúvida: PascalCase

Identificador	Convenção	Exemplo
Namespace	PascalCase	MeuNamespace
Classe	PascalCase	MinhaClasse
Método	PascalCase	MeuMetodo
Propriedades de classe	PascalCase	MinhaPropriedade
Variáveis de métodos	camelCase	minhaVariavel
Parâmetros de métodos	camelCase	meuParametro
Variável interna	_camelCase	_minhaVariavel

# Introdução ao C#

## Tipos de dados em C# - Tipos VALOR

C# Type	.Net Framework Type	Signed	Bytes	Possible Values
sbyte	System.Sbyte	Yes	1	-128 to 127
short	System.Int16	Yes	2	-32768 to 32767
int	System.Int32	Yes	4	$-2^{31}$ to $2^{31} - 1$
long	System.Int64	Yes	8	$-2^{63}$ to $2^{63} - 1$
byte	System.Byte	No	1	0 to 255
ushort	System.Uint16	No	2	0 to 65535
uint	System.Uint32	No	4	0 to $2^{32} - 1$
ulong	System.Uint64	No	8	0 to $2^{64} - 1$
float	System.Single	Yes	4	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ with 7 significant figures
double	System.Double	Yes	8	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ with 15 or 16 significant figures
decimal	System.Decimal	Yes	12	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$ with 28 or 29 significant figures
char	System.Char	N/A	2	Any Unicode character
bool	System.Boolean	N/A	1/2	true or false

# Introdução ao C#

## Tipos de dados em C# - Tipos REFERÊNCIA

Tipo C#	Tipo .NET	Descrição
string	System.String	Uma cadeia de caracteres Unicode <b>IMUTÁVEL</b> (segurança, simplicidade, thread safe)
object	System.Object	Um objeto genérico (toda classe em C# é subclasse de object)

# Introdução ao C#

---

## Saída de dados

### Forma básica

```
Console.WriteLine(valor);  
Console.Write(valor);
```

### Concatenação

```
Console.WriteLine("x vale " + x + " e y vale " + y);
```

### Placeholder

```
Console.WriteLine("x vale {0} e y vale {1}", x, y);
```

### Interpolação

```
Console.WriteLine($"x vale {x} e y vale {y}");
```

# Introdução ao C#

---

## Saída de dados

### Formatação de casas decimais

```
Console.WriteLine(x.ToString("F2"));
```

### Formatação de separador decimal → Usa System.Globalization

```
using System.Globalization;
```

```
Console.WriteLine(x.ToString(CultureInfo.InvariantCulture));
```

### Combinando os dois

```
using System.Globalization;
```

```
Console.WriteLine("F2"), x.ToString(CultureInfo.InvariantCulture));
```



# Introdução ao C#

---

## Entrada de dados

### Forma básica – String

```
valor = Console.ReadLine();
```

### Forma básica – Int, bool, etc.

```
int valor = int.Parse(Console.ReadLine());
```

### Formatação de separador decimal → Usa System.Globalization

```
using System.Globalization;
```

```
double valor = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
```

# Revisão

---

## Próxima aula

### Revisão lógica programação usando C#

- Estudo dos operadores aritméticos, de atribuição, comparativos e lógicos
- Conversão implícita e casting de tipos de dados
- Implementação de estruturas condicionais: if-else e operador ternário
- Utilização de laços de repetição: while, do-while, for e foreach

# Dúvidas?

---

[renan.duarte@gedre.ufsm.br](mailto:renan.duarte@gedre.ufsm.br)

GEDRE – Prédio 10 – CTLAB