

Nivelamento de Lógica de Programação e OO – Aula 4

Samira Antunes

Tópicos de hoje:

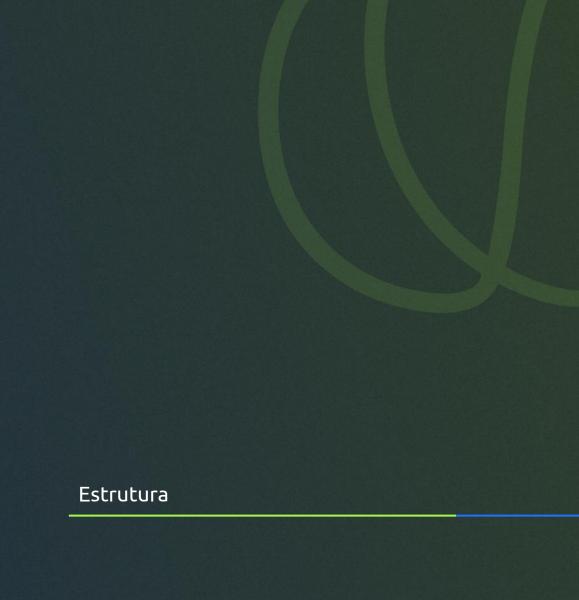
- Estrutura
- Métodos e Construtores
- Encapsulamento
- Propriedades

Combinados & Recados

Passaremos muito tempo juntos

- Câmera aberta, se possível.
- Levantar a mão em caso de dúvida.
- O representante da turma é o Ricardo Fazoli.
- Não esqueçam que teremos a rubrica de autoavaliação, avaliação da instrutora e avaliação do curso.
- A Caixa terá acesso dessa avaliação.





Estrutura

- Using
 - O que é: É como importar uma biblioteca ou caixinha de ferramentas para usar no seu código.
 - Exemplo: using System;
 - O System tem várias classes prontas, como Console, que você usa para Console.WriteLine().
 - Resumo: using é como dizer: quero ter acesso às ferramentas que estão nesse pacote."



Estrutura

Visão geral

- Namespace
 - O que é: Um sobrenome ou pasta lógica para agrupar classes, métodos e outros arquivos.
 - Por que serve: Organiza o projeto. & Evita que duas classes com o mesmo nome entrem em conflito.
 - Exemplo:

```
namespace Aula_4
{
    public class Conversor
    {
        // ...
}
```

 Resumo: Pense no namespace como o endereço ou sobrenome da sua classe dentro do projeto



Estrutura Main e Classes

- Um projeto C# costuma ter:
 - class Program: É onde normalmente fica o ponto de entrada da aplicação: o método Main.
 - static void Main(): É o primeiro método que o C# executa quando você roda o programa — como a "porta de entrada" da aplicação.
 - Outras classes "peças" ou "modelos" que você cria para organizar o código. Ex.: a classe Conversor.



Estrutura Main e Classes

```
static void Main()
    // ponto de entrada
// outras funcionalidades
```



Comentar linha

- Comentar: Ctrl + K seguido de Ctrl + C
- Descomentar: Ctrl + K seguido de Ctrl + U
- Passo a passo:
 - Selecione as linhas de código que quer comentar.
 - Pressione Ctrl+K e logo depois Ctrl+C para comentar(o Visual Studio adiciona // em cada linha).
 - Para remover, selecione novamente e pressione Ctrl+K e Ctrl+U.



Métodos e Construtores

Métodos

- Definição: um método é um bloco de código nomeado que executa uma ação ou calcula um valor.
- Estrutura básica:

```
[modificador] [tipoRetorno] NomeDoMetodo([parâmetros])
{
    // corpo do método
}
```

- modificador: public, private, etc.
- tipoRetorno: tipo do valor devolvido (int, string, void se não retorna nada).



Métodos

- Por que usar métodos?
- Para evitar repetir código, deixar a lógica organizada e facilitar manutenção.



Métodos

```
public class Conversor
    public double CelsiusParaFahrenheit(double c) => (c * 9/5) + 32;
    public double FahrenheitParaCelsius(double f) => (f - 32) * 5/9;
 public class Tabuada
     public void Imprimir(int numero)
         for (int i = 1; i <= 10; i++)
             Console.WriteLine(\$"{numero} x {i} = {numero * i}");
```

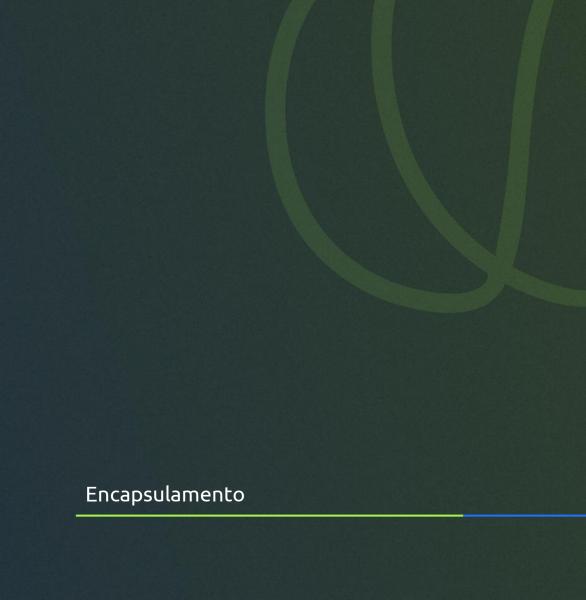


Construtor

- Método especial que tem o mesmo nome da classe e roda automaticamente quando um objeto é criado.
- Exemplo:

```
public class Pessoa
{
    public string Nome { get; set; }
    public Pessoa(string nome) // Construtor
    {
        Nome = nome;
    }
}
```





- Ideia central: esconder os detalhes internos e expor apenas o necessário.
- Benefícios: segurança, clareza e menor chance de erros.
- Como aplicar em C#: usando modificadores de acesso:
 - public: pode ser acessado por qualquer código.
 - private: só dentro da própria classe.
 - protected: dentro da classe e de suas herdeiras.
 - internal: acessível apenas dentro do mesmo assembly (projeto/compilação).



- Por que encapsular?
- Para proteger dados, evitar alterações acidentais e manter a classe como uma "caixa preta" com regras.





```
private string senha = "1234";
private double valorGuardado;
public bool Depositar(double valor, string tentativaSenha)
   if (tentativaSenha == senha)
       valorGuardado += valor;
       return true;
   return false;
```





Propriedades

- Propriedade é um "acesso controlado" a um dado interno da classe.
- Ela parece uma variável quando usamos, mas é, na verdade, um método disfarçado (com get e set automáticos).
- É como a campainha de uma casa: você não entra direto (campo/field), você toca a campainha (propriedade), e a casa decide se abre ou não.
- Campo é o dado cru, escondido.
- Propriedade é o jeito seguro de acessar ou alterar esse dado, podendo validar, calcular ou impedir mudanças.



Propriedades

- Por que usar propriedades em vez de campos públicos?
- Para ter controle (validação, acesso somente leitura) sem mudar a forma de usar a classe.



Propriedades

```
public class Pessoa
   private string nome; // Campo interno
   public string Nome // Propriedade
      get { return nome; } // retorna o valor
      set { nome = value; } // define o valor
   Uso:
   Pessoa p = new Pessoa();
   Console.WriteLine(p.Nome); // usa o get
```



Propriedade Automática

- O C# permite simplificar.
- O compilador cria o campo privado automaticamente.
- É como dizer: "C# cuide do campo para mim".

```
public class Pessoa
{
    public string Nome { get; set; }
}
```



Propriedade

```
public class Carro
{
    public string Modelo { get; set; }
    public int Ano { get; private set; }

    public Carro(string modelo, int ano)
    {
        Modelo = modelo;
        Ano = ano;
    }
}
```



Propriedade

```
private double celsius;
public double Celsius
   get => celsius;
   set
       if (value > -273.15) // acima do zero absoluto
            celsius = value;
public double Fahrenheit => (celsius * 9 / 5) + 32;
```





Integração dos conceitos

Lista de exercícios:

Métodos e Construtores

- Crie uma classe Conversor com dois métodos:
 - double CelsiusParaFahrenheit(double c)
 - double FahrenheitParaCelsius(double f)
- No Main, peça ao usuário uma temperatura e converta para a outra escala.



Integração dos conceitos

Lista de exercícios:

Propriedades

- Crie a classe Produto com:
 - Propriedade Nome (string)
 - Propriedade Preco (double) que não permite valor negativo.
- No Main, teste atribuir um preço negativo e mostre a mensagem de erro no set.



Integração dos conceitos

Lista de exercícios:

Encapsulamento

- Crie a classe ContaBancaria com:
 - Campo privado _saldo
 - Método Depositar(double valor)
 - Método Sacar(double valor) que não permita saldo negativo
 - Método ExibirSaldo().
- No Main, peça ao usuário depósitos e saques e exiba o saldo a cada operação.



Integração dos conceitos

Lista de exercícios:

Entrega: até 28/09/2025.

Envio para o e-mail da professora.



Obrigada