

Nivelamento de Lógica de Programação e OO – Aula 7

Samira Antunes

Tópicos de hoje:

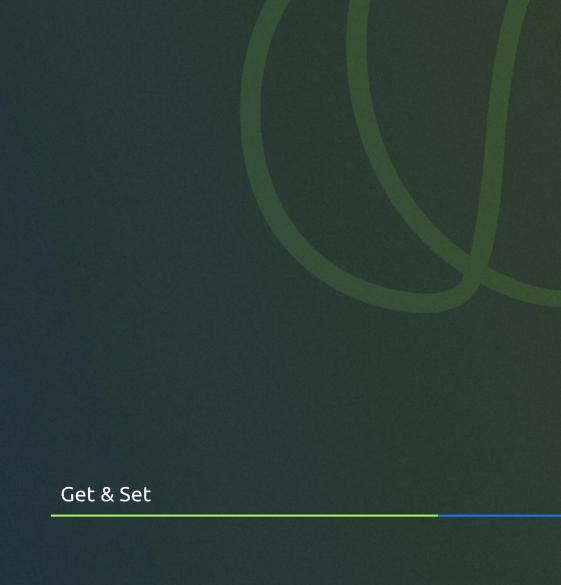
- Get & Set
- Sistema de Pedidos de Lanchonete
- Prática

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.





Maiores detalhes

- Propriedades em C# permitem controlar o acesso a atributos de uma classe.
- Usam as palavras-chave get (para ler) e set (para atribuir valor).
- São uma forma de aplicar encapsulamento (esconder os detalhes internos e expor só o necessário).



```
public class Pessoa
{
    // Campo privado
    private string nome;

    // Propriedade com get/set
    public string Nome
    {
        get { return nome; } // permite ler
        set { nome = value; } // permite alterar
    }
}
```



- private string nome; → variável interna (só a classe pode acessar).
- public string Nome { ... } → propriedade pública que expõe esse valor.
- get { return nome; } → quando alguém faz p.Nome, esse código roda e devolve o valor.
- set { nome = value; } → quando alguém faz p.Nome = "Ana";, o valor "Ana" é
 passado automaticamente na palavra-chave value.
- Exemplo de uso:

```
Pessoa p = new Pessoa();
p.Nome = "Carlos";    // chama o set
Console.WriteLine(p.Nome); // chama o get- Resultado: Carlos
```



- Pessoa p = new Pessoa();
 - Aqui estamos criando um objeto da classe Pessoa.
 - Pessoa é o molde (classe).
 - p é a variável que guarda o objeto (uma "instância" da classe).
 - · Com isso, agora temos um espaço na memória que representa uma pessoa.
- p.Nome = "Carlos";
 - Quando usamos p.Nome = "Carlos", o set da propriedade Nome é chamado.
 - O valor "Carlos" é passado automaticamente para o value dentro do set.
 - Isso faz com que a variável interna (nome, que é privada) receba esse valor.
 - É como se disséssemos: "Pessoa p, guarde dentro do seu atributo nome o valor Carlos".
- Console.WriteLine(p.Nome);
 - Quando usamos p.Nome, o get da propriedade é chamado.
 - Ele devolve o valor que está guardado no campo privado (nome).
 - O Console.WriteLine imprime esse valor na tela.



- set é usado quando atribuímos um valor → p.Nome = "Carlos";
- get é usado quando lemos um valor → Console.WriteLine(p.Nome);
- Propriedade = "porta de entrada controlada" para os atributos da classe.



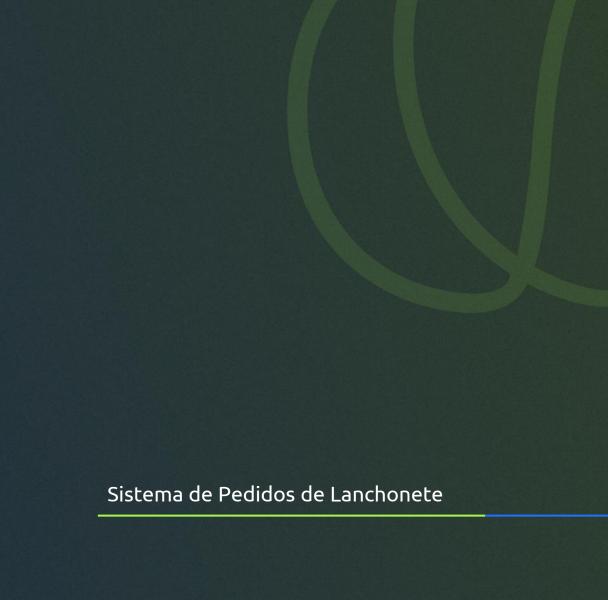
Exemplo com regra no set

```
private decimal saldo;
public decimal Saldo
   get { return saldo; }
    set
        if (value < ♥)
            Console.WriteLine("Saldo não pode ser negativo!");
        else
            saldo = value;
```



- Aqui temos uma regra de negócio dentro do set.
- Se alguém tentar colocar um valor negativo (conta.Saldo = -100), o programa não atualiza e mostra a mensagem.
- Caso contrário, o saldo é atualizado normalmente.
- Exemplo de uso:





Entendendo o sistema

Objetivo

- Criar um programa de console que permita:
 - Cadastrar produtos (lanche, bebida).
 - Registrar pedidos de clientes.
 - Calcular o total da conta.
- Conceitos Usados:
 - Classes, propriedades e construtores (Produto, Bebida).
 - Herança e polimorfismo: Bebida herda de Produto e sobrescreve CalcularPreco.
 - Coleções genéricas (List<Produto>).
 - Loops e condicionais para o menu interativo.



Classe Base

Produto.cs

```
public string Nome { get; set; }
public double Preco { get; set; }
public Produto(string nome, double preco)
    Nome = nome;
    Preco = preco;
// Método virtual para permitir que subclasses alterem o cálculo
public virtual double CalcularPreco()
    return Preco;
```



Classe Base

Produto.cs

- namespace MiniHackathon
 - "Sobrenome" que agrupa todas as classes do projeto. Facilita a organização e evita conflitos de nomes.
- public class Produto
 - É a classe base. Representa qualquer item do cardápio.
- Propriedades Nome e Preco
 - public para poder acessar de fora. get; set; cria automaticamente o campo interno (propriedade automática).
- Construtor Produto(string nome, double preco)
 - Obriga quem criar um produto a informar nome e preço.
- Método virtual CalcularPreco()
 - Retorna o preço normal, mas é virtual para que classes filhas (como Bebida) possam sobrescrever e mudar o cálculo.



Classe Derivada

Bebida.cs

```
// Bebida herda de Produto
    public bool Gelada { get; set; }
    public Bebida(string nome, double preco, bool gelada)
        : base(nome, preco)
       Gelada = gelada;
    // Se a bebida estiver gelada, acrescenta 10% no preço
    public override double CalcularPreco()
        return Gelada ? Preco * 1.1 : Preco;
```



Classe Base

Produto.cs

- public class Bebida : Produto
 - Bebida herda de Produto (herança). Isso significa que Bebida já tem Nome, Preco e CalcularPreco.
- Propriedade Gelada
 - Específica de bebida, indica se é servida gelada.
- Construtor com : base(nome, preco)
 - Chama o construtor da classe pai (Produto) para inicializar Nome e Preço.
- override CalcularPreco()
 - Muda o comportamento: se Gelada for true, acrescenta 10% no preço.



```
static void Main()
    // Cardápio
    var menu = new List<Produto>
        new Produto("Hambúrguer", 15.0),
        new Produto("Batata Frita", 8.0),
        new Bebida("Refrigerante", 6.0, true),
        new Bebida("Suco Natural", 7.0, false)
    var pedido = new List<Produto>();
    int opcao;
        Console.WriteLine("\n--- Menu ---");
        for (int i = 0; i < menu.Count; i++)</pre>
            Console.WriteLine($"{i + 1} - {menu[i].Nome} (R$ {menu[i].CalcularPreco():0.00})");
        Console.WriteLine("0 - Finalizar Pedido");
        Console.Write("Escolha: ");
        opcao = int.Parse(Console.ReadLine() ?? "0");
        if (opcao > 0 && opcao <= menu.Count)</pre>
            pedido.Add(menu[opcao - 1]);
            Console.WriteLine($"{menu[opcao - 1].Nome} adicionado!");
    while (opcao != 0);
    double total = 0;
    Console.WriteLine("\n--- Pedido ---");
    foreach (var item in pedido)
        Console.WriteLine($"{item.Nome} - R$ {item.CalcularPreco():0.00}");
        total += item.CalcularPreco();
    Console.WriteLine($"Total: R$ {total:0.00}");
}}}
```

Ponto de partida

```
Program.cs
using System;
using System.Collections.Generic;
using MiniHackathon;
```



Ponto de partida

Program.cs

- using System / System.Collections.Generic
 - Importa bibliotecas para usar Console e List<T>.
- List<Produto> menu
 - Cria uma lista de produtos e bebidas. Note que Bebida também é Produto (herança), então a mesma lista armazena os dois tipos.
- Loop do...while
 - Mostra o menu repetidamente até o usuário digitar 0.
- Console.ReadLine / int.Parse
 - Lê a escolha do usuário e converte para número.
- pedido.Add(menu[opcao 1])
 - Adiciona o item escolhido à lista de pedido.



Ponto de partida

Program.cs

- foreach
 - Percorre a lista de pedidos para exibir cada item e somar o total.
 - Chama item.CalcularPreco():
 - Para Produto, retorna o preço original.
 - Para Bebida, retorna preço com acréscimo se gelada.
 - Isso demonstra polimorfismo em tempo de execução.



Resumo

Visão geral

- Produto.cs Classe base: define nome, preço e um método virtual.
- Bebida.cs Classe filha: herda Produto e muda o cálculo de preço.
- Program.cs Ponto de entrada: mostra o menu, recebe pedidos e calcula total, usando herança + polimorfismo para tratar produtos e bebidas de forma unificada.





Sistema de Cadastro e Relatórios Objetivo

- Cadastrar alunos (nome, curso, nota).
- Listar, buscar por curso e gerar relatório de médias por curso.

CadastroAlunos/

├ Program.cs

├ Aluno.cs

└ Cadastro.cs



Mini Biblioteca

Objetivo

- Cadastrar livros.
- Registrar empréstimos e devoluções.
- Listar livros disponíveis e empréstimos ativos.

```
MiniBiblioteca/

├─ Program.cs

├─ Livro.cs

├─ LivroDigital.cs

├─ Usuario.cs

└─ Biblioteca.cs
```



Mini Folha de Pagamento Objetivo

- Cadastrar funcionários de tipos diferentes (Horista, Assalariado, Comissionado).
- Listar funcionários e seus salários calculados.
- Exibir o total da folha do mês.

```
FolhaPagamento/

Program.cs

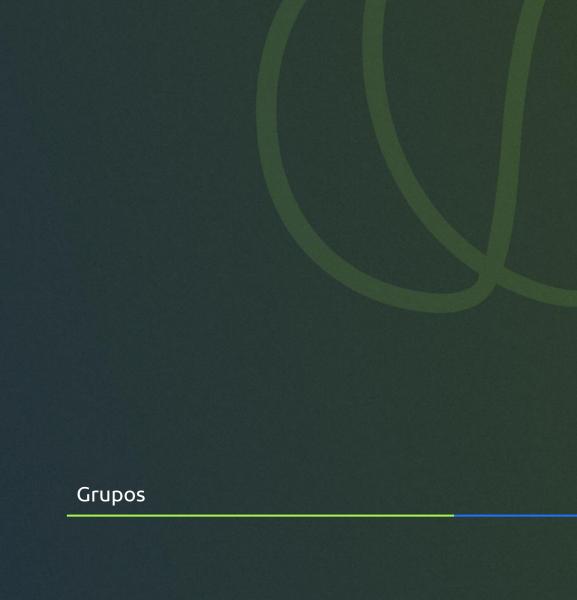
Funcionario.cs

Horista.cs

Assalariado.cs

Comissionado.cs
```





Mini Hackathon

Grupo 1 - https://meet.google.com/bem-xjga-jjf

- Allan Faria Rocha De Oliveira
- Breno Oliveira Arantes
- Diego Eduardo Lima Santos Dos Santos
- Dorival Dalbelo Junior
- Edemar Adacheski Junior
- Felipe Martins Rodrigues

Tema: Sistema de Cadastro e Relatórios



Mini Hackathon

Grupo 2 - https://meet.google.com/emd-goxe-tnv

- Felipe Pereira Ramos
- Filipi Martins Fonseca
- Francisco Halleys De Queiroz Bezerra
- Geraldo Fagner Rodrigues Gaia
- Guilherme Lucas Krom Paccola
- Henrique Araújo Porto



Mini Hackathon

Grupo 3 - https://meet.google.com/qfo-nzks-kwo

- Jefferson Siqueira Costa
- Joelson Costa Batista
- Jose Vicente Pereira
- Joziane De Freitas Da Costa Rodrigues
- Jucemar Jose De Oliveira Reis
- Lidiane Jose De Moura



Mini Hackathon

Grupo 4 - https://meet.google.com/jhy-jpqh-mer

- Marcio Henrique Barranco
- Marco Aurélio Tridico
- Marcos Da Silva Ferreira
- Marcos Rogelio Franco Da Silva
- Michelle Rosler Kley
- Michelle Sant Ana Oliveira



Mini Hackathon

Grupo 5 - https://meet.google.com/eqp-zouq-gvn

- Márcio Caetano Dos Santos
- Paulo Eduardo Lopes Bezerra
- Rafael Fernandes De Melo Lopes
- Rafael Szmit
- Ricardo Fazoli Da Silva
- Robson Ferreira De Souza

Tema: Mini Biblioteca



Mini Hackathon

Grupo 6 - https://meet.google.com/kcy-ppuf-mcu

- Thais Gadiole Schontag
- Thiago Formigoni Dias
- Yann Gabriel Lopes Vasques
- Yuri Cardoso Marques

