

Nivelamento de Lógica de Programação e OO – Aula 5

Samira Antunes

Tópicos de hoje:

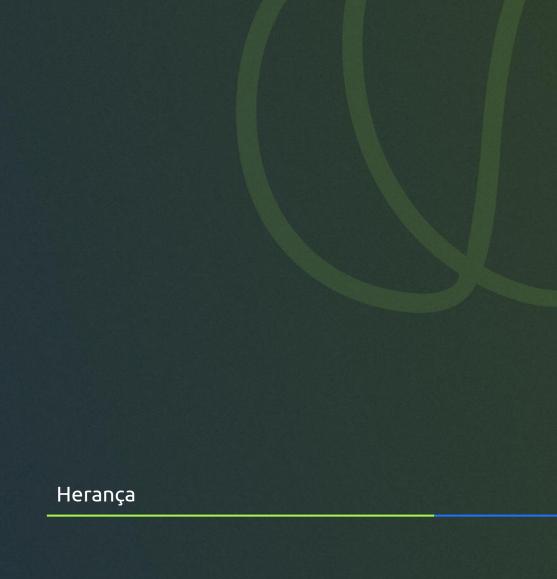
Herança

#### 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.





- Definição: mecanismo que permite que uma classe (derivada ou filha) receba automaticamente propriedades e métodos de outra classe (base ou pai).
- Analogia:
  - Imagine uma "Classe Mãe" Animal com a característica "Respirar".
  - Toda "Classe Filha" (Cachorro, Gato) já nasce sabendo respirar, sem precisar reescrever.



- Reuso de código: escreve-se uma vez, usa-se em várias classes.
- Organização: deixa clara a relação entre tipos de objetos.
- Manutenção fácil: alteração na classe base propaga para as filhas.



```
class ClasseFilha : ClassePai
{
    // novos atributos ou métodos
}
```

- : indica que a filha herda da pai.
- public: membros visíveis em qualquer lugar.
- protected: visíveis apenas na classe base e nas classes derivadas (ótimo para herança).



### Herança simples

Exemplo

```
public void Comer() => Console.WriteLine("Comendo...");
    public void Latir() => Console.WriteLine("Au au!");
// Uso
var dog = new Cachorro();
dog.Comer(); // herdado
dog.Latir(); // próprio
```



- public class Animal: declara uma classe base (pai).
- public void Comer(): método público (qualquer um pode chamar) que só imprime um texto.
- Cachorro : Animal: esta sintaxe significa "Cachorro herda de Animal". Na prática, todo Cachorro já vem com tudo que Animal tem (no caso, o método Comer()).
- Latir(): comportamento específico do Cachorro, que não existe em Animal.



# Herança simples

Exemplo

- o que acontece em tempo de execução?
  - new Cachorro() aloca um objeto do tipo Cachorro.
  - Como Cachorro herda de Animal, o objeto tem acesso a Comer().
  - Chamar dog.Comer() imprime "Comendo...".
  - Chamar dog.Latir() imprime "Au au!".
  - nota: se Comer() fosse protected (em vez de public), apenas a própria classe e classes filhas veriam o método – você não conseguiria chamar dog.Comer() de fora.



#### Construtores com base

```
Exemplo
```

```
public class Pessoa
    public string Nome { get; set; }
    public Pessoa(string nome) { Nome = nome; }
public class Aluno : Pessoa
    public string Curso { get; set; }
    public Aluno(string nome, string curso) : base(nome)
        Curso = curso;
// Uso
var aluno = new Aluno("Ana", "Computação");
Console.WriteLine($"{aluno.Nome} - {aluno.Curso}");
base(nome) chama o construtor da classe pai.
```



### Acesso protected

```
Exemplo
 public class Veiculo
     protected int Velocidade;
     public void Acelerar(int valor)
         Velocidade += valor;
         Console.WriteLine($"Velocidade: {Velocidade} km/h");
 public class Carro : Veiculo
     public void Turbo()
         Velocidade += 20; // possível por ser protected
         Console.WriteLine($"Turbo! Velocidade: {Velocidade} km/h");
```





### Exercícios Práticos

Integração dos conceitos

### Lista de exercícios:

Exercício 1: Hierarquia simples

- Crie uma classe Veiculo com propriedade Marca e método Ligar().
- Crie as classes Carro e Moto que herdem de Veiculo e adicionem métodos próprios, como AbrirPortaMala() ou Empinar().



#### Exercícios Práticos

Integração dos conceitos

## Lista de exercícios:

# Exercício 2: Pretected

- Crie uma classe Conta com campo protegido saldo.
- Crie uma classe ContaCorrente que permita sacar valores, acessando saldo diretamente dentro da classe filha.



### Exercícios Práticos

Integração dos conceitos

# Lista de exercícios:

Entrega: até 28/09/2025.

Envio para o e-mail da professora.



Obrigada