Curso Técnico

de Programação de Jogos Digitais

Aula 06

índice

[1. Orientação a Objetos em C 3](#_Toc169074971)

[2. Classes e Objetos 3](#_Toc169074972)

[3. Referencias 6](#_Toc169074973)

# Orientação a Objetos em C

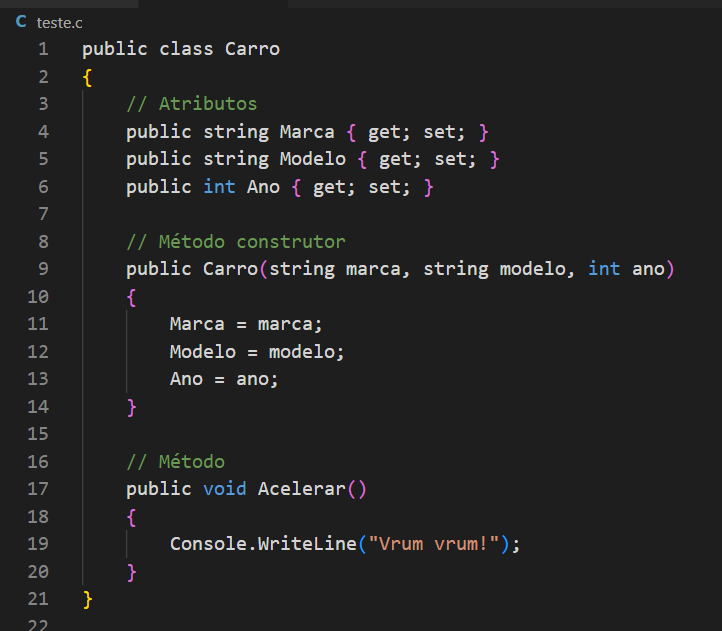
Na programação C#, a Orientação a Objetos (OO) é uma abordagem crucial para o desenvolvimento de sistemas robustos e escaláveis. Esta metodologia permite a modelagem do mundo real de forma mais eficiente, promovendo a reutilização de código e facilitando a manutenção. Vamos explorar os principais conceitos da Orientação a Objetos em C#:

# Classes e Objetos

Em C#, as classes e objetos são os pilares fundamentais da programação orientada a objetos. Vamos entender o significado de cada um:

Uma classe em C# é uma estrutura que define as características e comportamentos de um objeto. Ela serve como um modelo para criar objetos. As classes são compostas por atributos (também chamados de campos ou propriedades) e métodos.

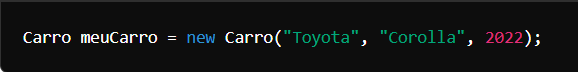
Exemplo de uma classe em C#:



Neste exemplo, a classe `Carro` possui atributos como `Marca`, `Modelo` e `Ano`, além de um método `Acelerar()`.

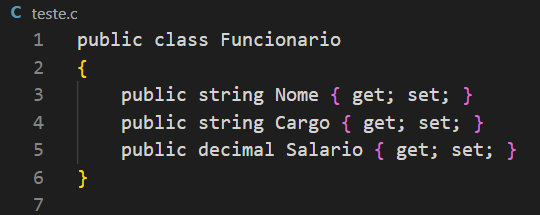
Um objeto é uma instância de uma classe. Em outras palavras, é uma representação específica de uma classe com valores definidos para seus atributos. Quando você cria um objeto, está reservando espaço na memória para armazenar seus dados.

Exemplo de criação de objetos em C#:

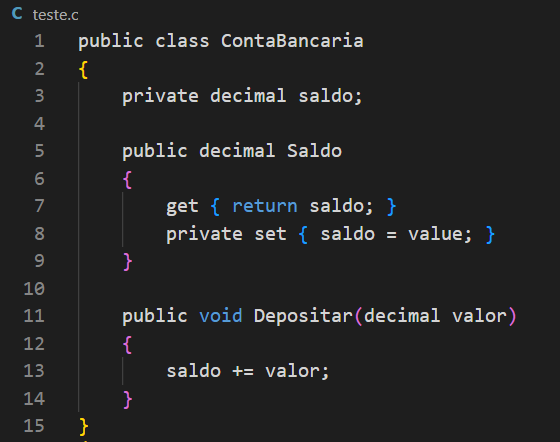


Neste exemplo, `meuCarro` é um objeto da classe `Carro` com a marca "Toyota", modelo "Corolla" e ano "2022".

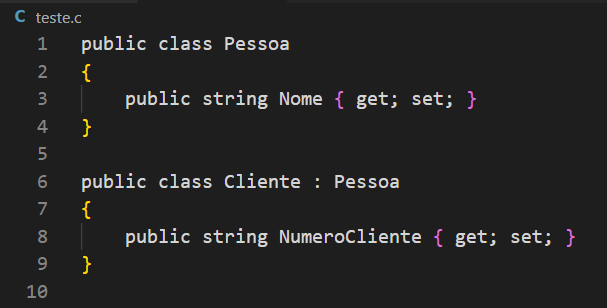
A abstração é a capacidade de representar características essenciais sem incluir os detalhes complexos. Em C#, isso é alcançado através da definição de classes, que são modelos para objetos. Por exemplo, ao modelar um sistema de gerenciamento de funcionários, podemos criar uma classe `Funcionario` com propriedades como nome, cargo e salário.



O encapsulamento é o conceito de esconder os detalhes de implementação de uma classe e exibir apenas os métodos públicos necessários para interagir com ela. Isso promove a segurança e a modularidade do código. Por exemplo, em uma classe `ContaBancaria`, podemos encapsular o saldo para evitar acesso direto.

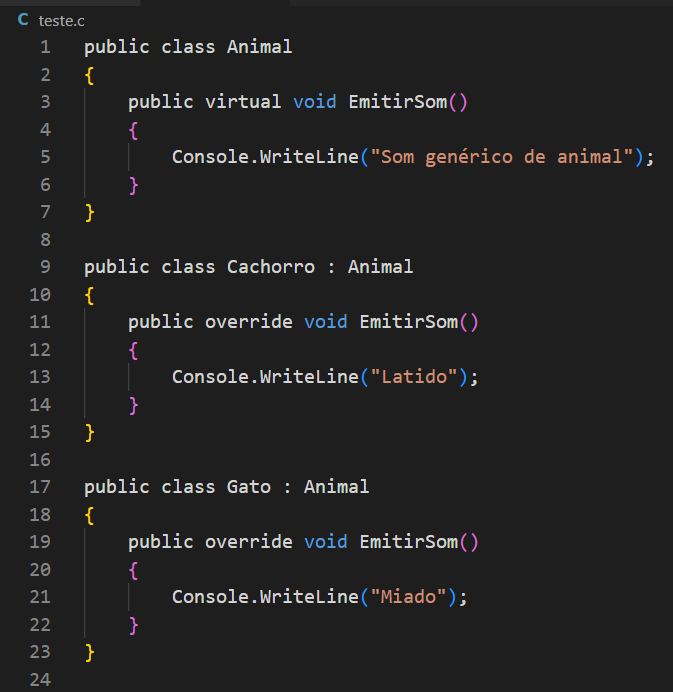


A herança permite que uma classe herde propriedades e comportamentos de outra classe. Em C#, uma classe pode herdar de apenas uma classe base, mas a herança pode ser feita em várias camadas.



No exemplo acima, a classe `Cliente` herda a propriedade `Nome` da classe `Pessoa`.

O polimorfismo permite que objetos de diferentes classes sejam tratados de maneira uniforme, simplificando o código e tornando-o mais flexível. Em C#, o polimorfismo é alcançado através de métodos virtuais e sobrescrita de métodos.



Aqui, as classes `Cachorro` e `Gato` herdam da classe `Animal` e sobrescrevem o método `EmitirSom` para emitir os sons específicos de cada animal.

# Referencias

1. Unity Technologies. (2020). **Unity User Manual.** Unity Technologies.
2. [C#, 2020] **Visual C# Developer Center**, Microsoft Docs.

Microsoft Corporation. (2020).

1. Geig, M. (2018). **Unity 2018 Game Development in 24 Hours**, Sams Teach Yourself. Sams Publishing.
2. Hocking, J. (2015). **Unity in Action: Multiplatform Game Development in C#.** Manning Publications.