# Properties – Propriedades 11/11/2020

São definições de métodos encapsulados, porém expondo uma sintaxe similar à de atributos e não de métodos.

Uma propriedade é um membro que oferece um mecanismo flexível para ler, gravar ou calcular o valor de um campo particular. As propriedades podem ser usadas como se fossem atributos públicos, mas na verdade elas são métodos especiais chamados "acessadores". Isso permite que os dados sejam acessados facilmente e ainda ajuda a promover a segurança e a flexibilidade dos métodos.

Fonte: <https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/properties>

Ainda usando o exemplo dos produtos. Como estava.

Código da classe Produto:

using System.Globalization;

namespace ControleDeEstoque

{

class Produto

{

private string \_nome;

private double \_preco;

private int \_quantidade;

public Produto()

{

}

public Produto(string nome, double preco, int quantidade)

{

\_nome = nome;

\_preco = preco;

\_quantidade = quantidade;

}

public string GetNome()

{

return \_nome;

}

public void SetNome(string nome)

{

if (nome != null && nome.Length > 1) {

\_nome = nome;

}

}

public double GetPreco()

{

return \_preco;

}

public int GetQuantidade()

{

return \_quantidade;

}

public double ValorTotalEmEstoque()

{

return \_preco \* \_quantidade;

}

public void AdicionarProdutos( int quantidade )

{

\_quantidade += quantidade;

}

public void RemoverProdutos (int quantidade)

{

\_quantidade -= quantidade;

}

public override string ToString()

{

return \_nome +

", $ " +

\_preco.ToString("F2", CultureInfo.InvariantCulture) +

", " +

\_quantidade +

" unidades, Total: $ " +

ValorTotalEmEstoque().ToString("F2", CultureInfo.InvariantCulture);

}

}

}

Como estava o código do programa principal:

using System;

using System.Globalization;

namespace ControleDeEstoque

{

class Program

{

static void Main(string[] args)

{

Produto p = new Produto("TV", 500.00, 10);

p.SetNome("TV 4K");

Console.WriteLine(p.GetNome());

Console.WriteLine(p.GetPreco());

}

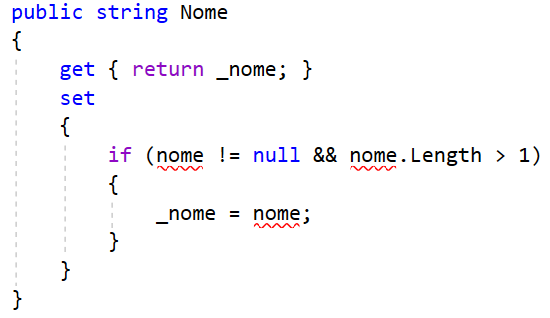
}

}

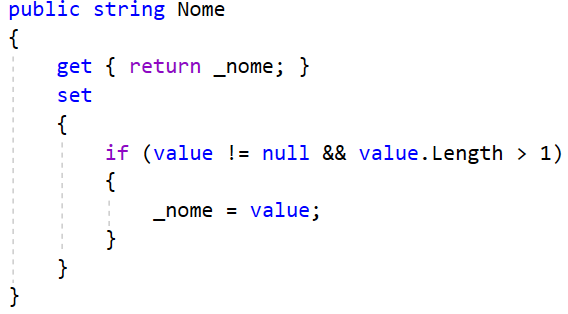
# **Criando Properties**

Properties do atributo Nome:

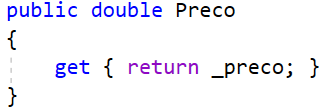
Pegue a lógica elaborada no método set antes e coloco dentro do método set.



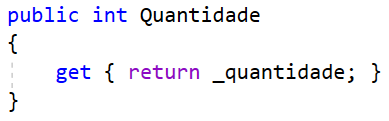
Porém, como não estamos recebendo parâmetros, precisamos substituir o nome (parâmetro) por value (valor).



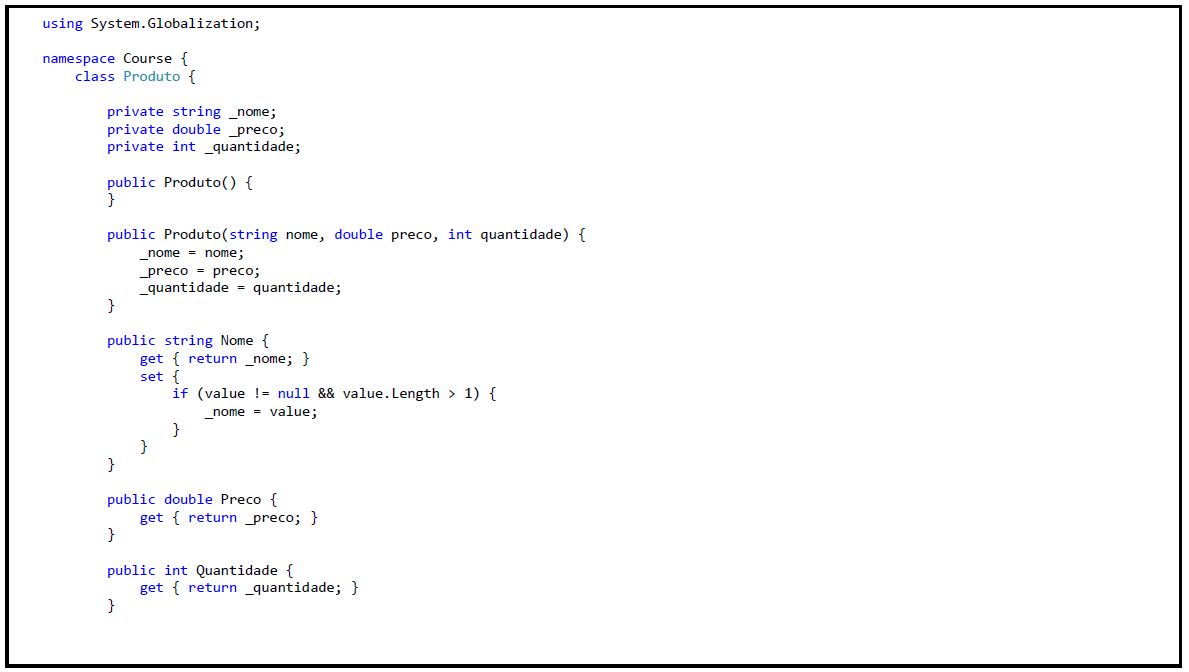
Properties do atributo Preco:

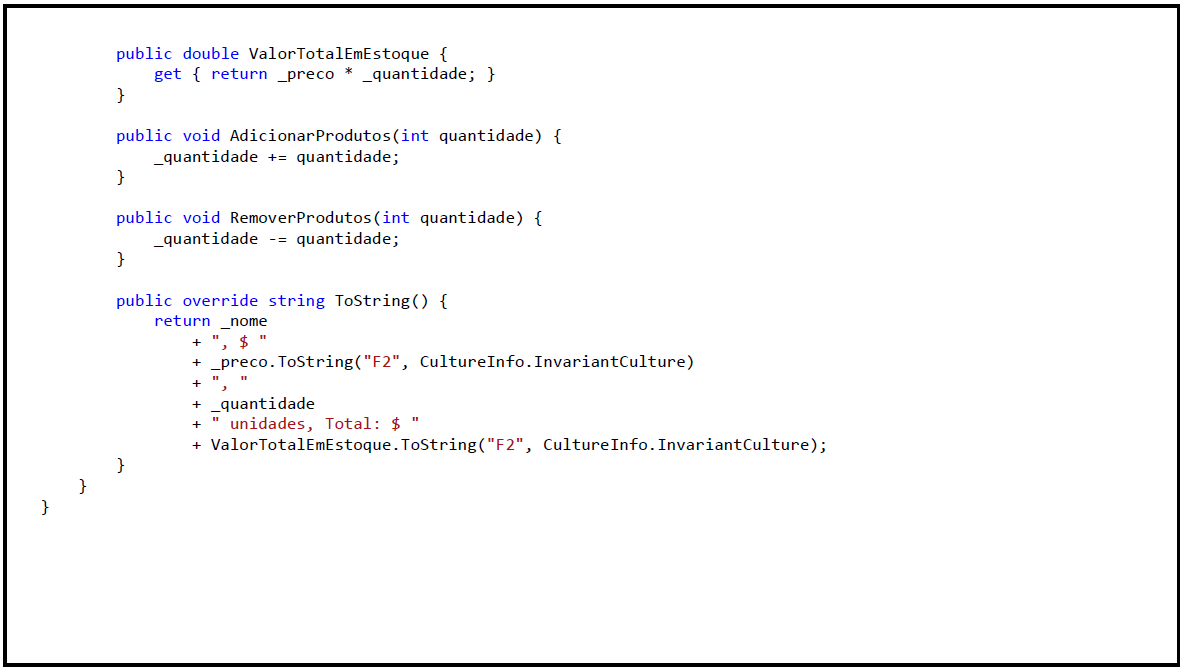


Properties do atributo Quantidade:

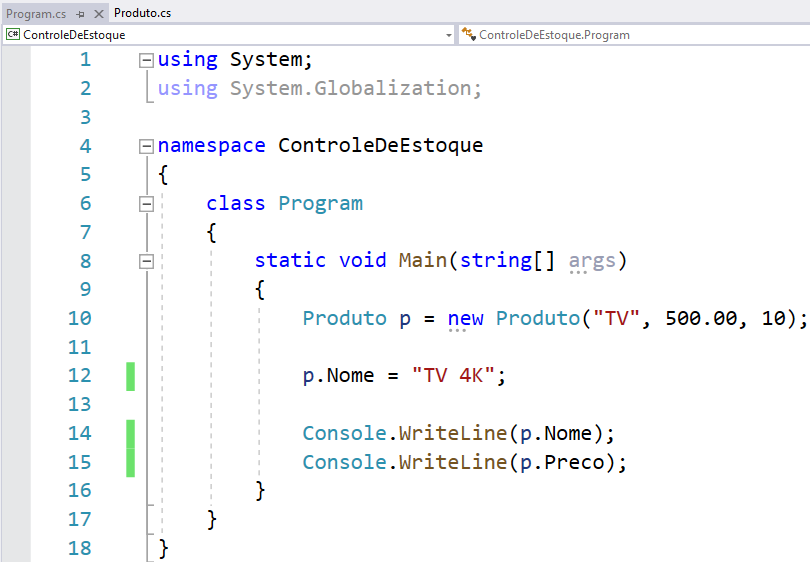


A classe irá ficar parecido com isso





A classe principal também precisa ser atualizada.



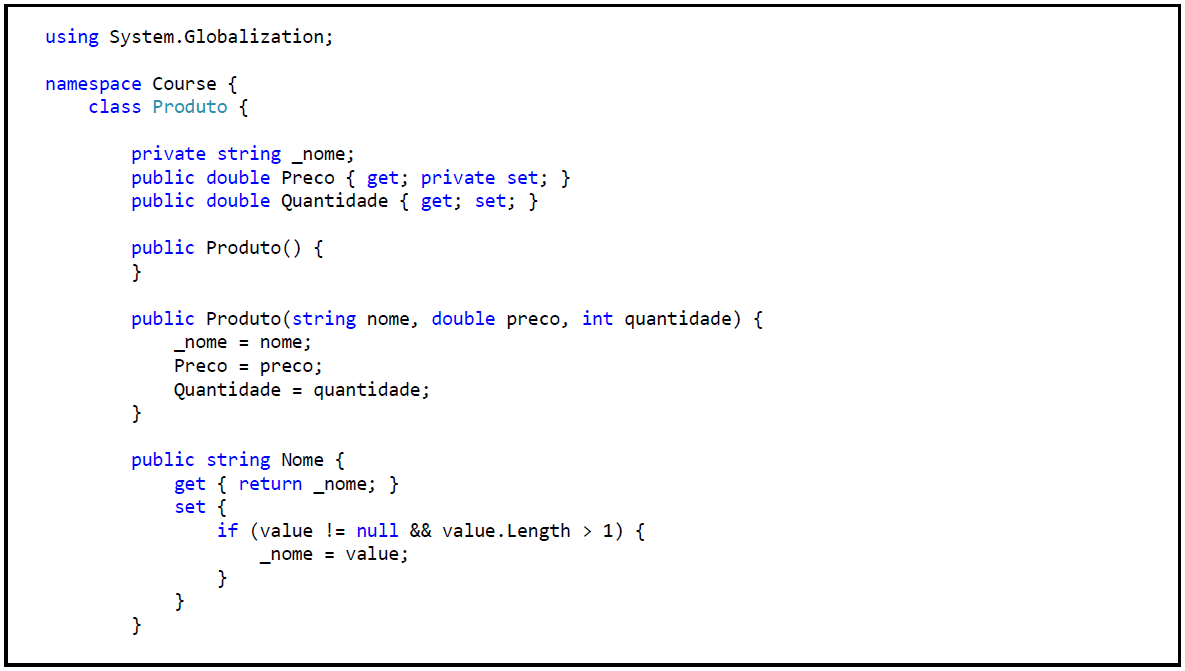
# Auto Properties

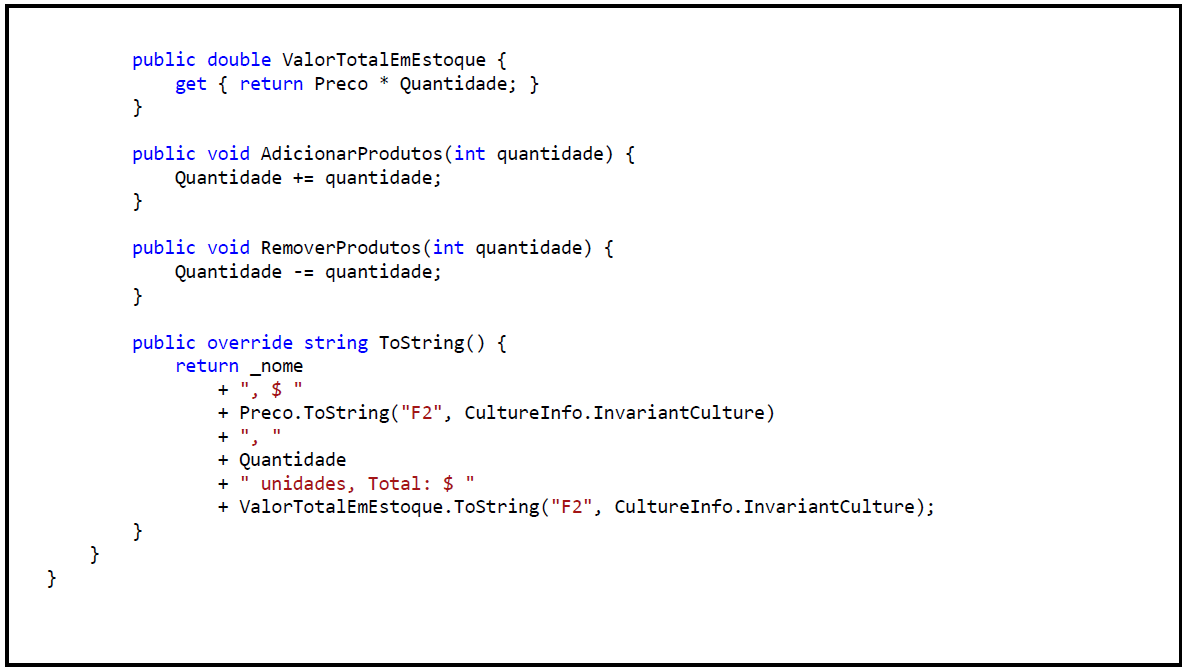
## Propriedades autoimplementadas

É uma forma simplificada de se declarar propriedades que não necessitam lógicas particulares para as operações get e set.

Exemplo:







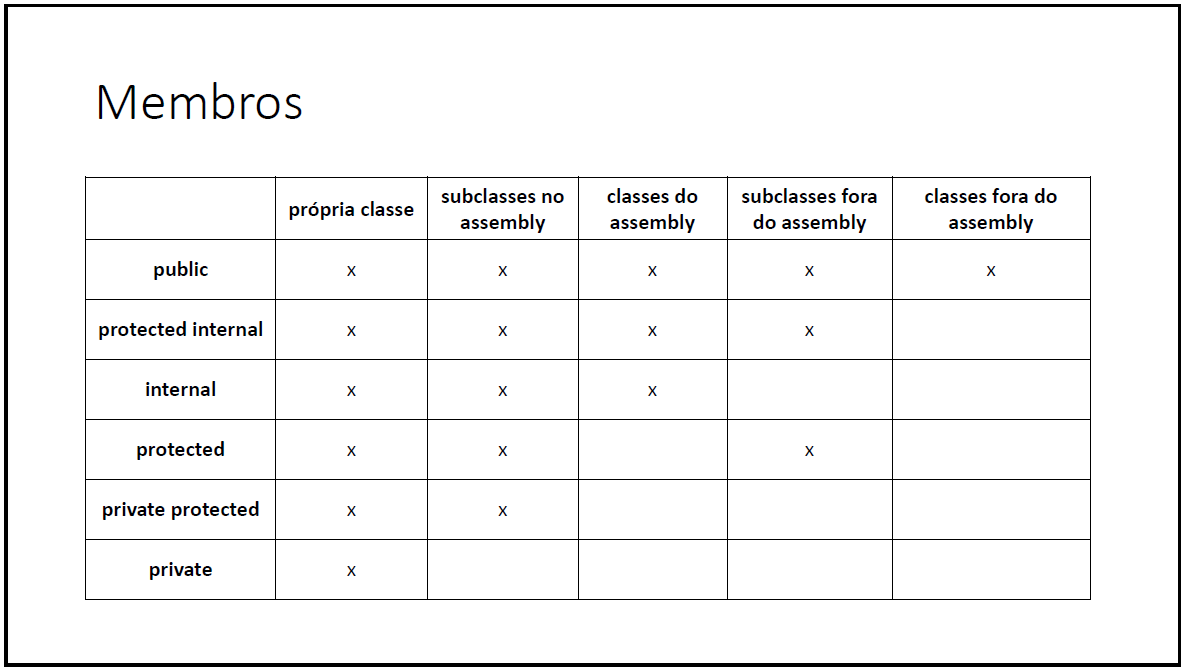
Referência:

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/auto-implemented-properties>

# Ordem sugerida para implementação de membros

1. Atributos privados
2. Propriedades autoimplementadas
3. Construtores
4. Propriedades customizadas
5. Outros métodos da classe

# Modificadores e acesso



## Classes

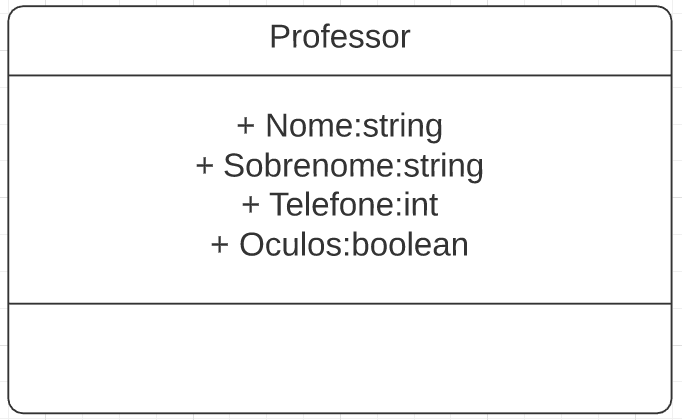
* Acesso por qualquer classe
  + public class Product
* Acesso somente dentro do assembly
  + internal class Product
  + class Product
* Acesso somente pela classe-mãe
  + private class Product
  + Nota: classe aninhada, por padrão, é private

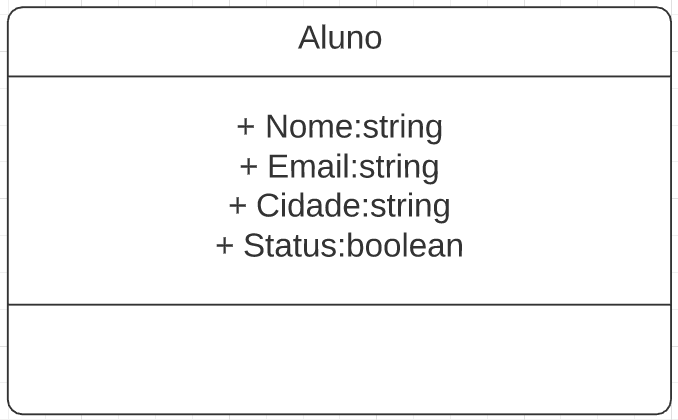
Referência:

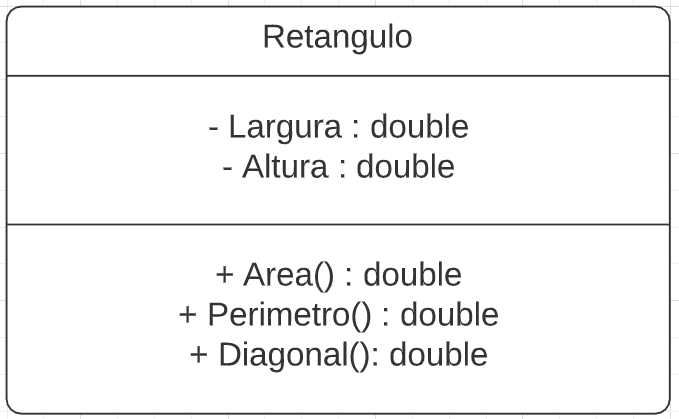
<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/access-modifiers>

# Atividade de fixação Encapsulamento com Get Set

Utilizando encapsulamento. Faça uma classe para os diagramas abaixo. Realize testes.

a) 

b) 

c) 

d) 