# Construtores – new Aula dia 26/10/2020

Construtor é uma operação especial da classe, que executa no momento da instanciação do objeto.

Usos comuns:

• Iniciar valores dos atributos

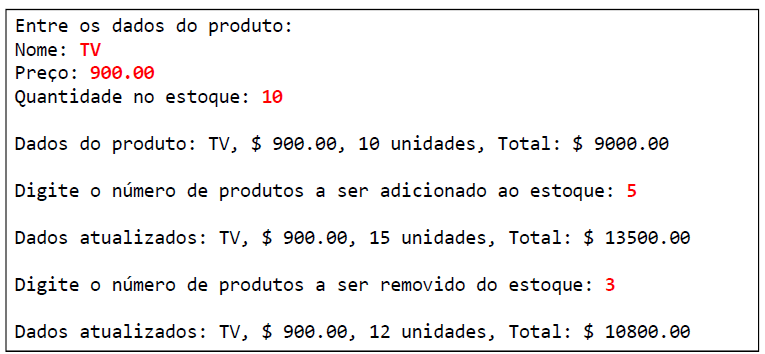
• Permitir ou obrigar que o objeto receba dados / dependências no momento de sua instanciação (injeção de dependência)

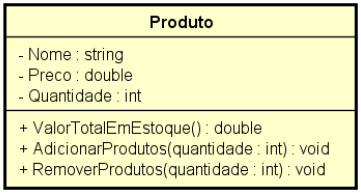
Se um construtor customizado não for especificado, a classe disponibiliza o construtor padrão:

• Produto p = new Produto();

É possível especificar mais de um construtor na mesma classe (sobrecarga)

Exemplo:





Nossa classe produto estava assim:

using System.Globalization;

namespace ControleDeEstoque

{

class Produto

{

public string Nome;

public double Preco;

public int 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);

}

}

}

Nosso programa principal estava assim:

using System;

using System.Globalization;

namespace ControleDeEstoque

{

class Program

{

static void Main(string[] args)

{

Produto p = new Produto();

Console.WriteLine("Entre os dados do produto:");

Console.Write("Nome: ");

p.Nome = Console.ReadLine();

Console.Write("Preço: ");

p.Preco = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

Console.Write("Quantidade: ");

p.Quantidade = int.Parse(Console.ReadLine());

Console.WriteLine();

Console.WriteLine();

double total = p.ValorTotalEmEstoque();

Console.WriteLine("Dados do produto: " + p);

Console.WriteLine();

Console.Write("Digite o número de produtos a ser adicionado ao estoque: ");

Console.WriteLine();

int qte = int.Parse(Console.ReadLine());

p.AdicionarProdutos(qte);

Console.WriteLine("Dados do produto: " + p);

Console.WriteLine();

Console.Write("Digite o número de produtos a ser removido do estoque: ");

qte = int.Parse(Console.ReadLine());

p.RemoverProdutos(qte);

Console.WriteLine();

Console.WriteLine("Dados do produto: " + p);

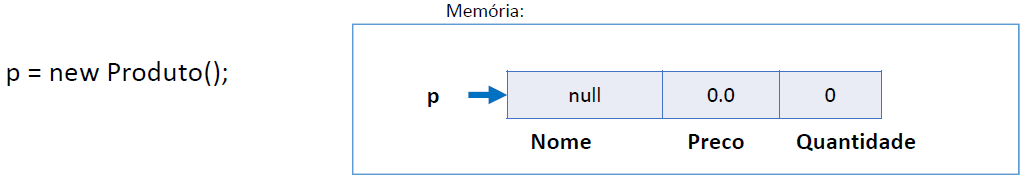
Console.ReadKey();

}

}

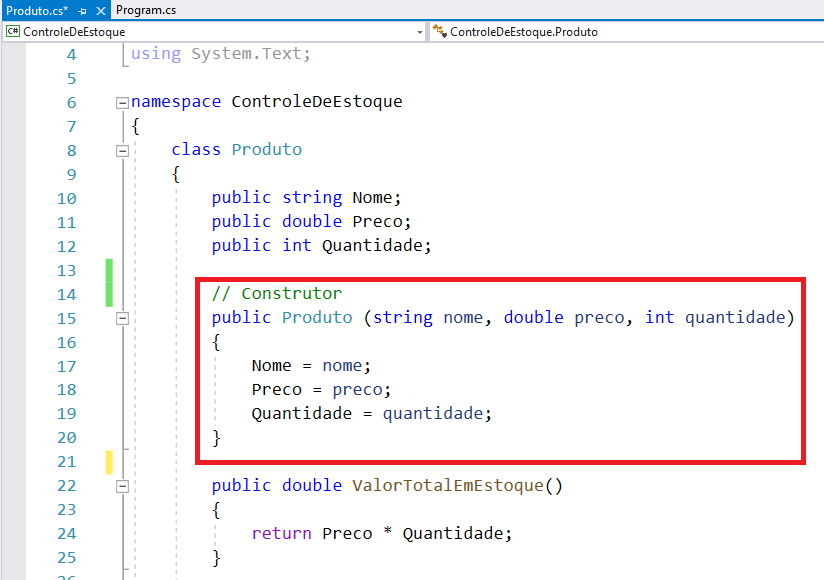
}

Quando instanciamos, o construtor inicia com valores nulos ou vazios.

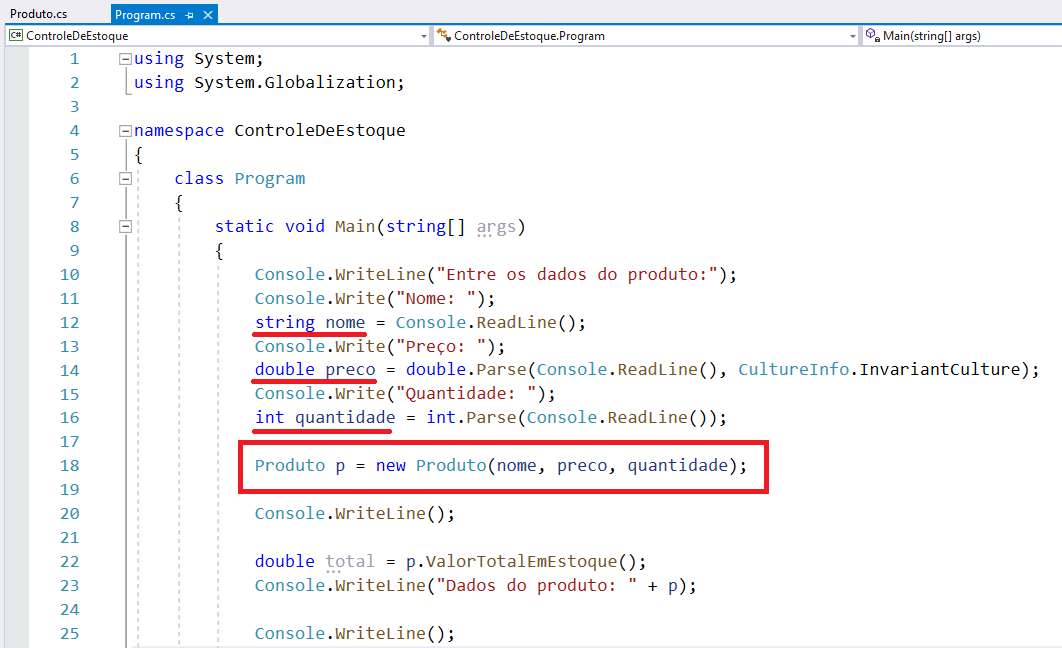


Com o intuito de evitar a existência de produtos sem nome e sem preço, é possível fazer com que seja “obrigatória” a iniciação desses valores.

Observem:



Ao alterar a classe incluindo o construtor. O código principal precisa ser alterado.



# Sobrecarga – Aula dia 30/10/2020

É um recurso que uma classe possui de oferecer mais de uma operação com o mesmo nome, porém com diferentes listas de parâmetros.

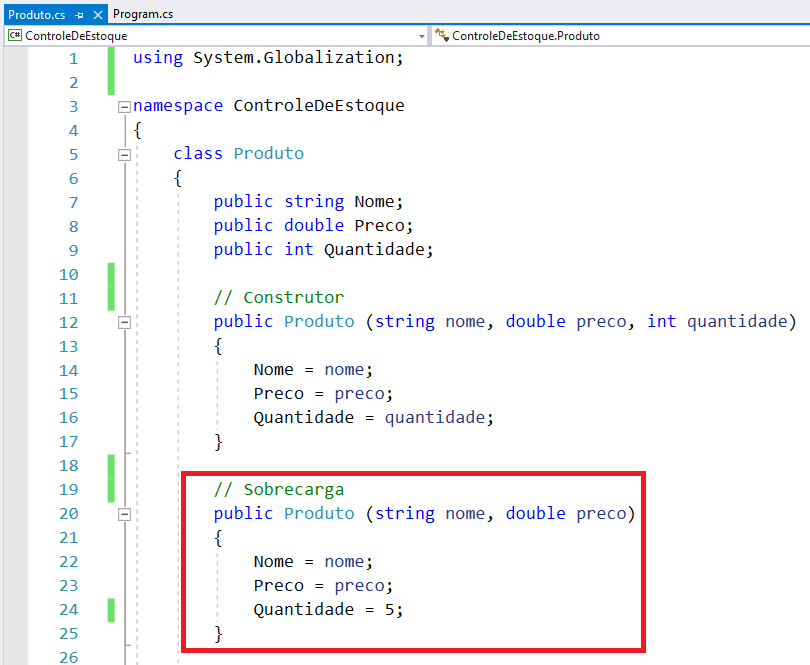
É uma forma de inicializar uma classe com valores específicos.

## Proposta de melhoria

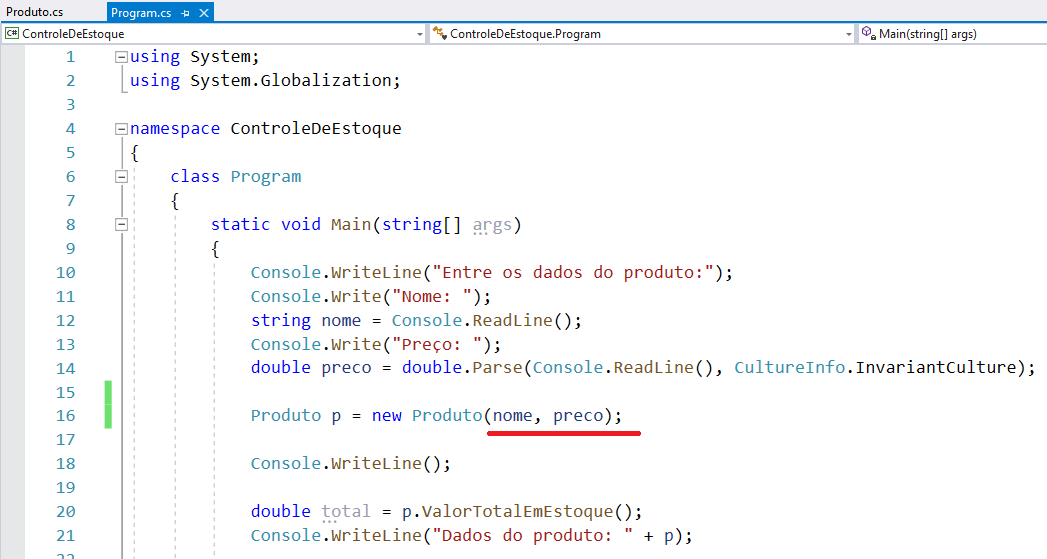
• Vamos criar um construtor opcional, o qual recebe apenas nome e preço do produto. A quantidade em estoque deste novo produto, por padrão, deverá então ser iniciada com o valor zero.

• Nota: é possível também incluir um **construtor padrão** (sem parâmetros)

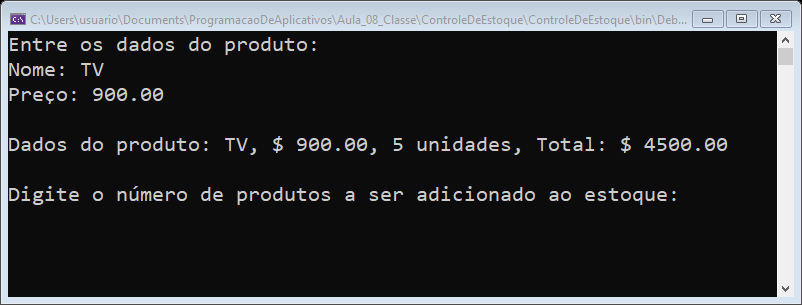
Vamos supor que uma das regras de negócio da empresa é sempre começar um novo produto com 5 unidades.



Agora retire a parte que solicita do usuário a quantidade no programa principal:

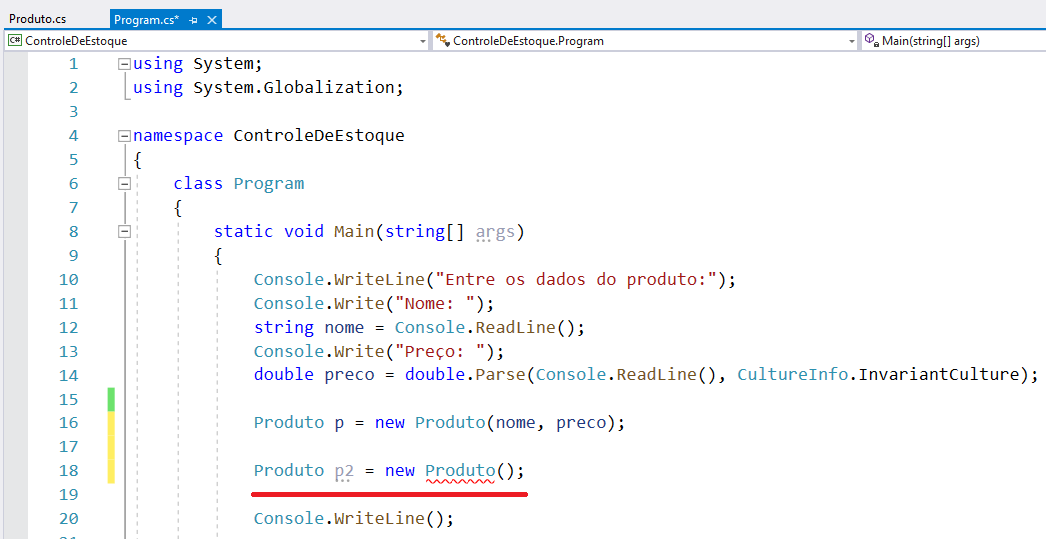


Execute com Ctrl + F5 e veja que mesmo sem digitar, a quantidade já veio como 5 unidades. Como se fosse um valor padrão.

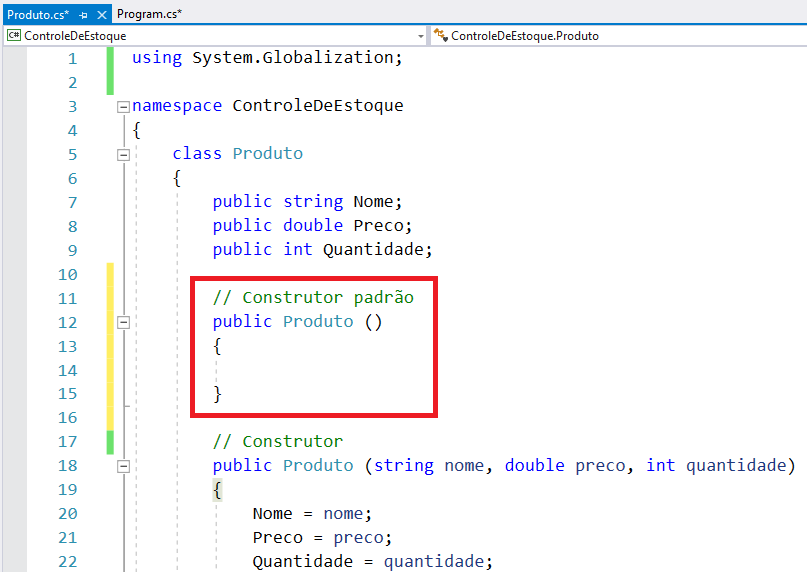


Lembre-se que o construtor padrão não pode ser chamado a não ser que seja construído na classe.

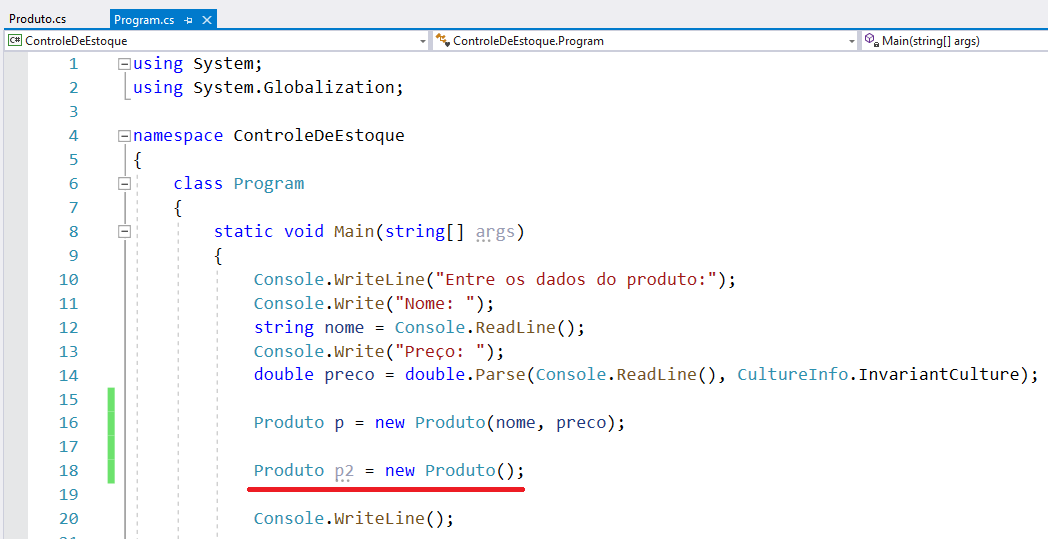
Exemplo:



Fazendo funcionar a sobrecarga, o construtor padrão precisa ser criado na classe:

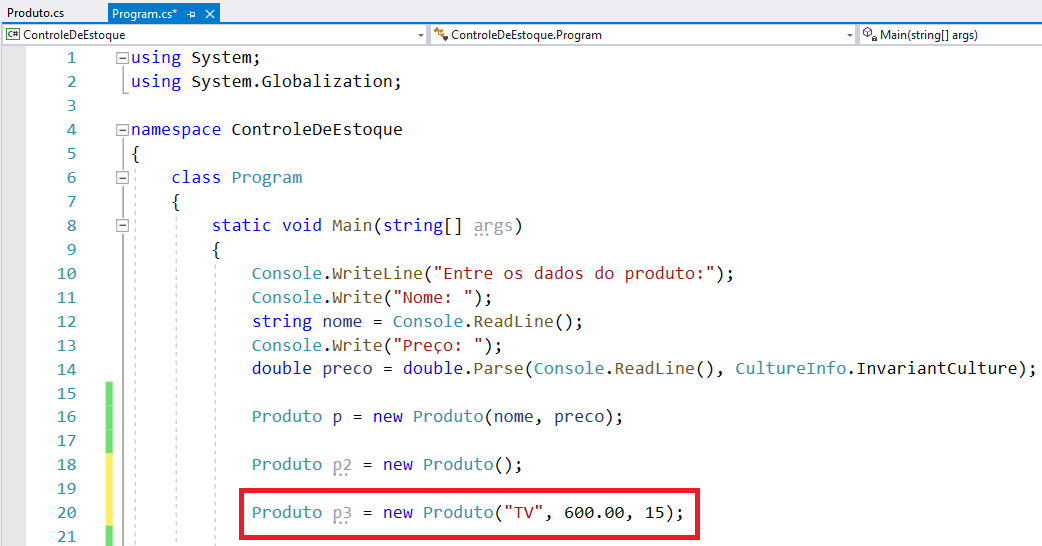


Agora o erro na classe principal sumiu.

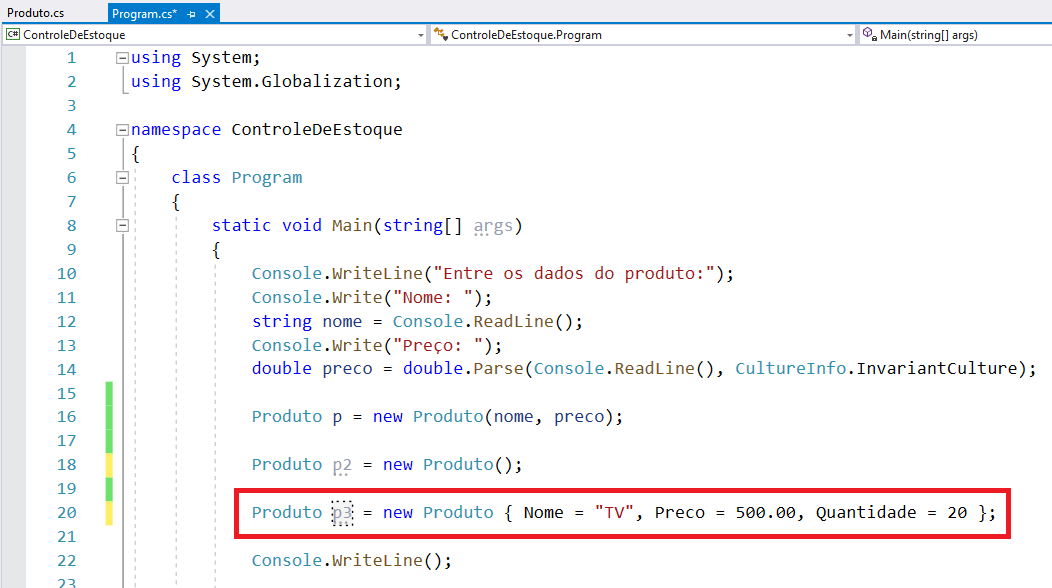


## Sintaxe alternativa para inicializar valores

Se você quiser já inserir os valores diretamente. Podemos fazer

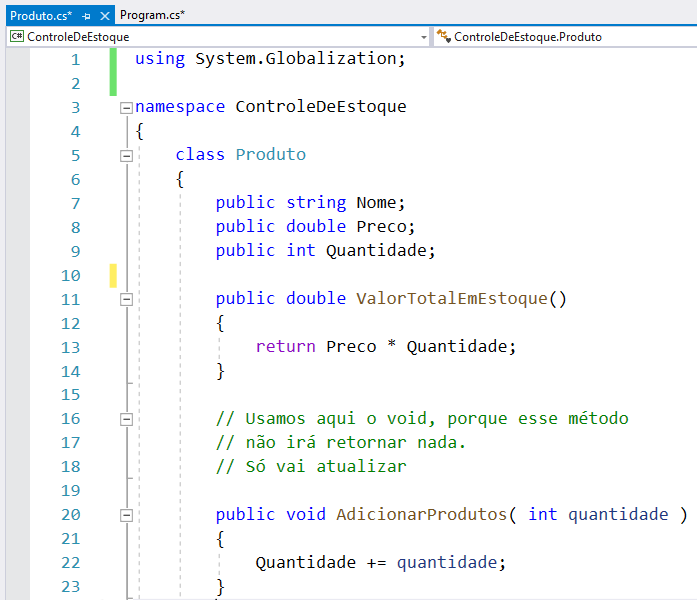


Ou assim

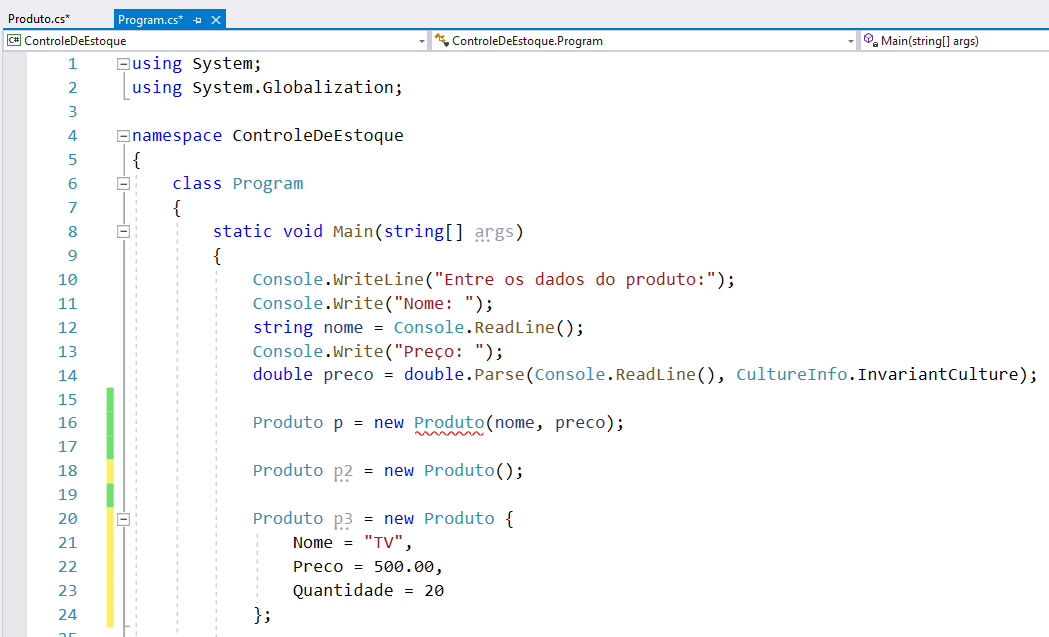


Mesmo quando a classe não tem construtores podemos utilizar essa sintaxe.

Vamos excluir os construtores da classe Produto como teste.

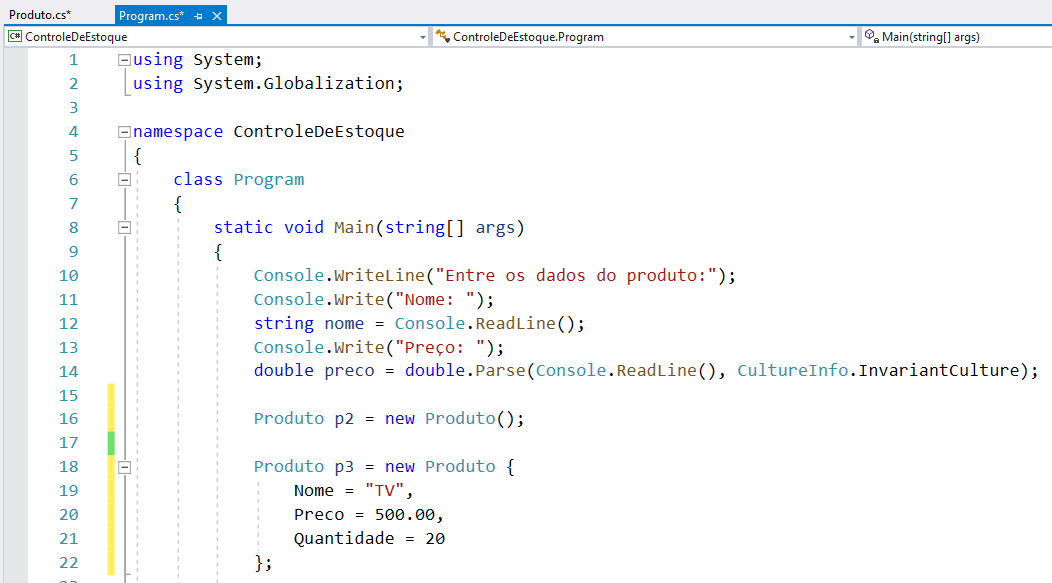


E observe a classe principal

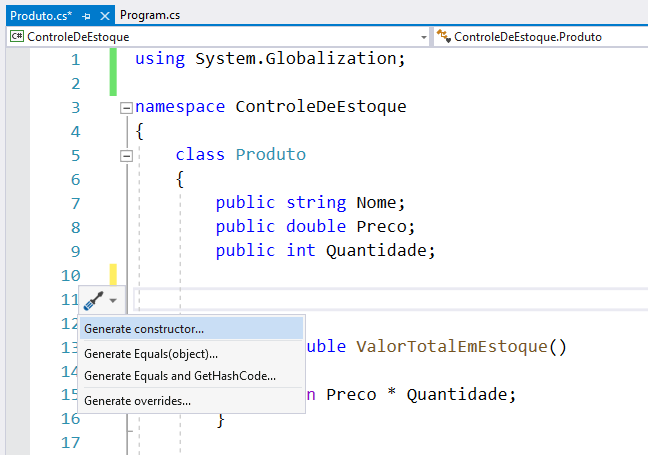


O construtor de dois argumentos da linha 16 não funciona, mas o construtor padrão funciona, porque não tem construtor, a instanciação padrão funciona. Mas essa a sintaxe (linha 20 até 24) de já informar os atributos, funciona perfeitamente.

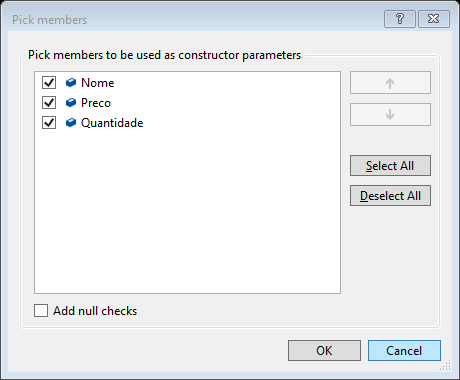
Podemos apagar a linha 16.



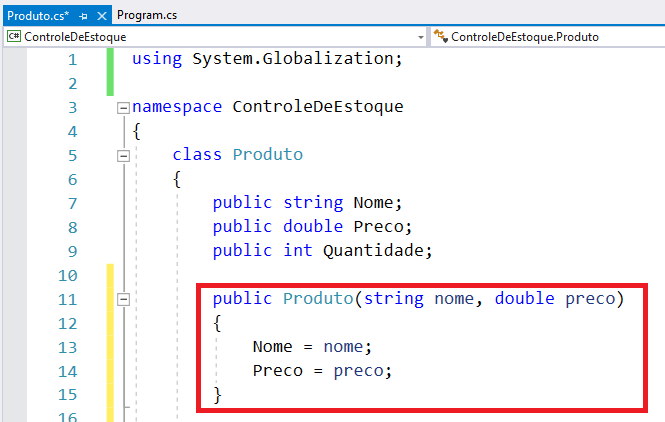
Produzindo um erro para e mostrando como gera o construtor de maneira gráfica.



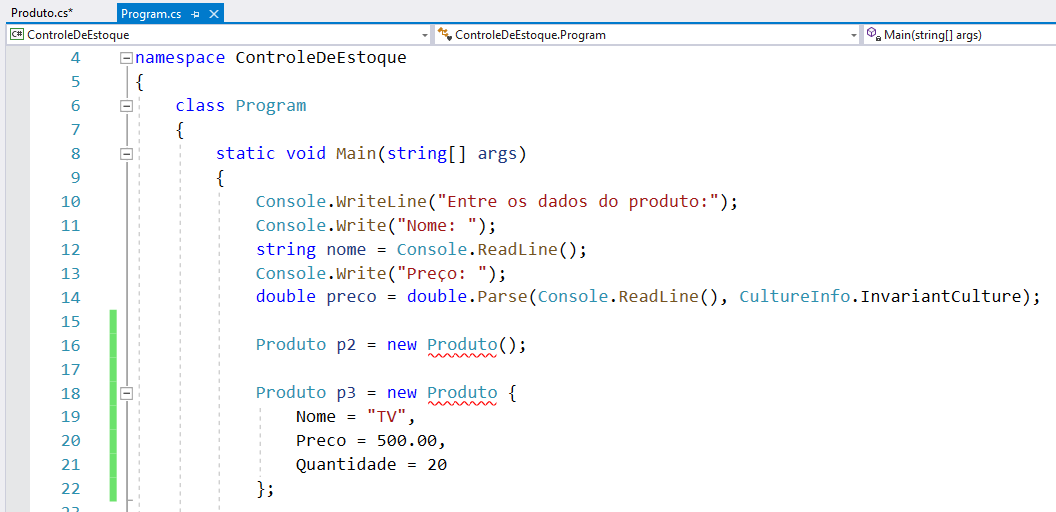
Podemos selecionar os atributos desejados.



Construtor gerado automaticamente.

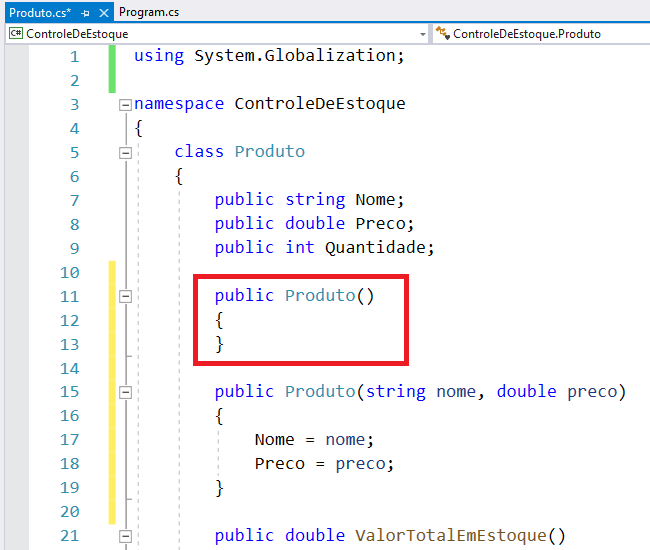


Observe na classe Program.cs que aparece erro nas duas formas



Por isso o construtor padrão é necessário.

Construindo novamente o construtor padrão na classe Produto



Exercício de fixação:

Faça um programa para cadastro de alunos. Crie uma classe Aluno com os atributos Nome, Telefone e Cidade. O programa principal deverá solicitar do usuário apenas o nome e telefone. Em seguida exiba na tela o nome, telefone e a cidade. Utilize o que aprendemos sobre construtor e sobrecarga para que a Naturalidade receba “Belo Horizonte”. Assim, por padrão, todos os alunos estarão morando em Belo Horizonte.

Exemplo:

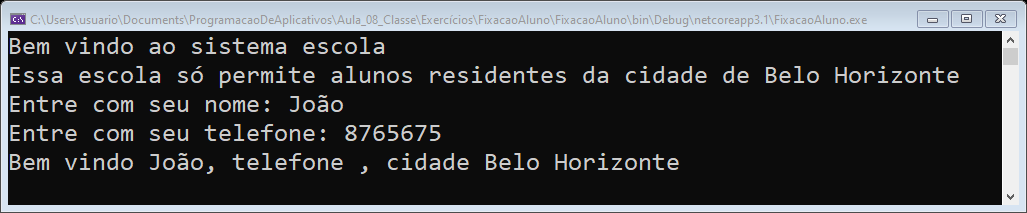
Bem vindo ao sistema escola

Essa escola só permite alunos residentes na cidade de Belo Horizonte

Entre com seu nome: **João**

Entre com seu telefone: **8765675**

Bem vindo **João**, telefone **8765675** cidade **Belo Horizonte**



Resolução

Classe:

namespace FixAluno

{

class Aluno

{

public string Nome;

public int Telefone;

public string Cidade;

public Aluno (string nome, int telefone)

{

Nome = nome;

Telefone = telefone;

Cidade = "Belo Horizonte";

}

public override string ToString()

{

return "Bem vindo "

+ Nome +

", telefone "

+ Telefone +

" cidade " +

Cidade;

}

}

}

Programa principal:

using System;

namespace FixAluno

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Bem vindo ao sistema escola!");

Console.WriteLine("Essa escola só permite alunos residentes na cidade de Belo Horizonte");

Console.Write("Entre com seu nome: ");

string nome = Console.ReadLine();

Console.WriteLine("Entre com seu telefone: ");

int telefone = int.Parse(Console.ReadLine());

Aluno aluno = new Aluno(nome, telefone);

Console.WriteLine(aluno);

}

}

}

# Palavra this 06/11/2020

A palavra this é uma referência para o próprio objeto

Usos comuns:

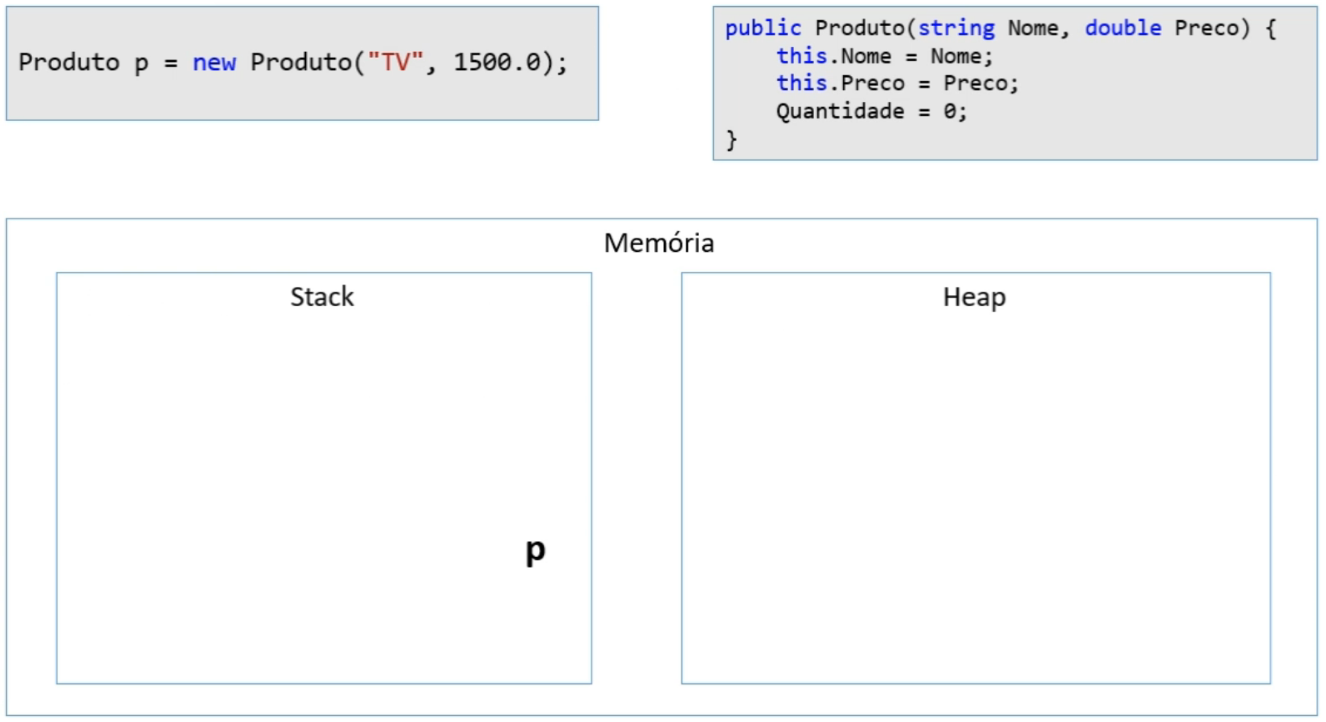
• Diferenciar atributos de variáveis locais (Java)

• Referenciar outro construtor em um construtor

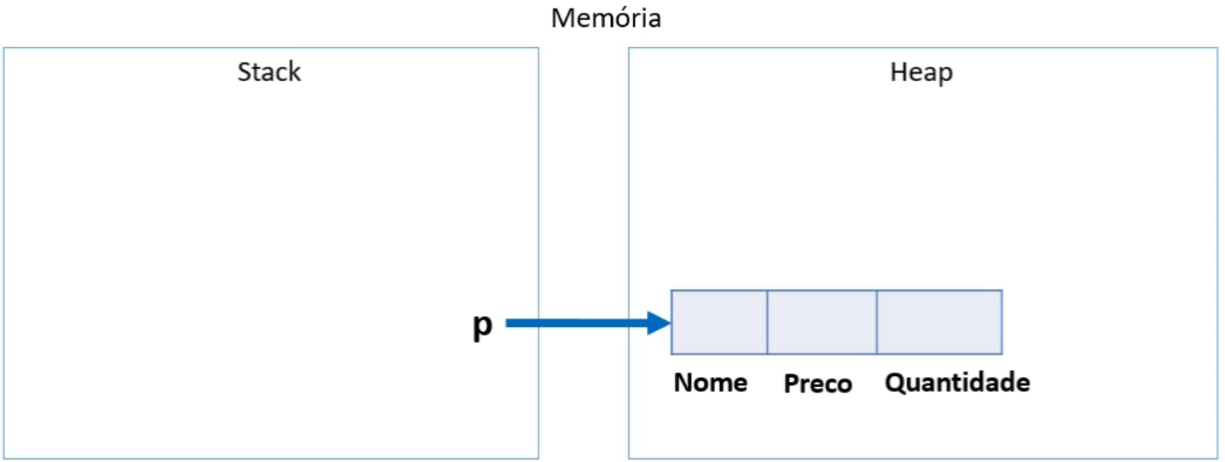
• Passar o próprio objeto como argumento na chamada de um método ou construtor

Primeiro exemplo:

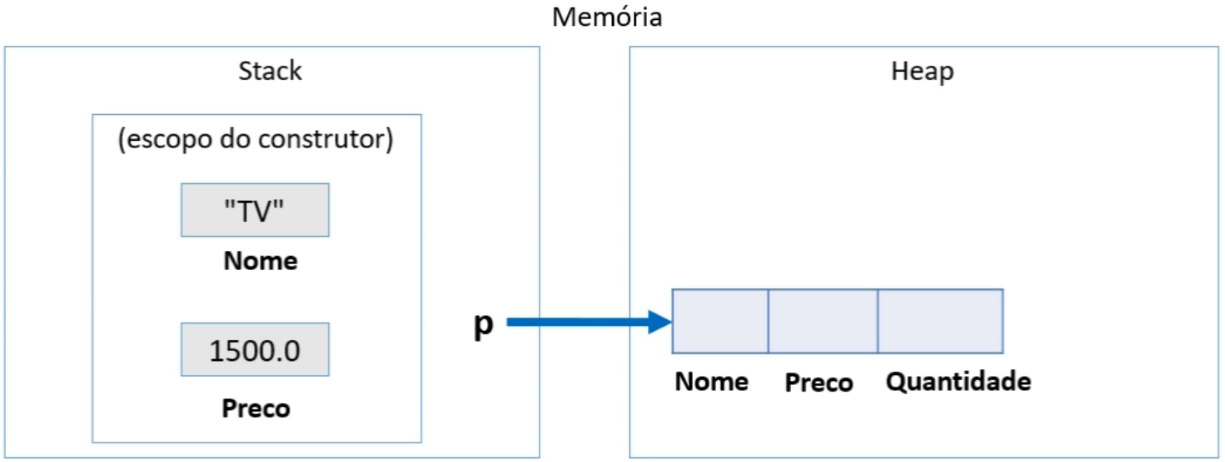
Diferenciar atributos de variáveis locais



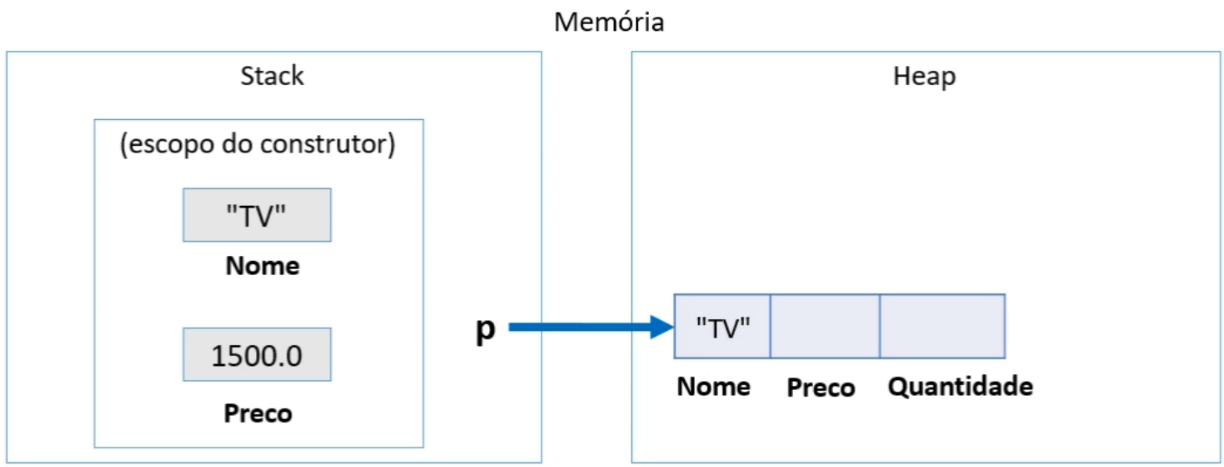
O produto p sendo isntanciado



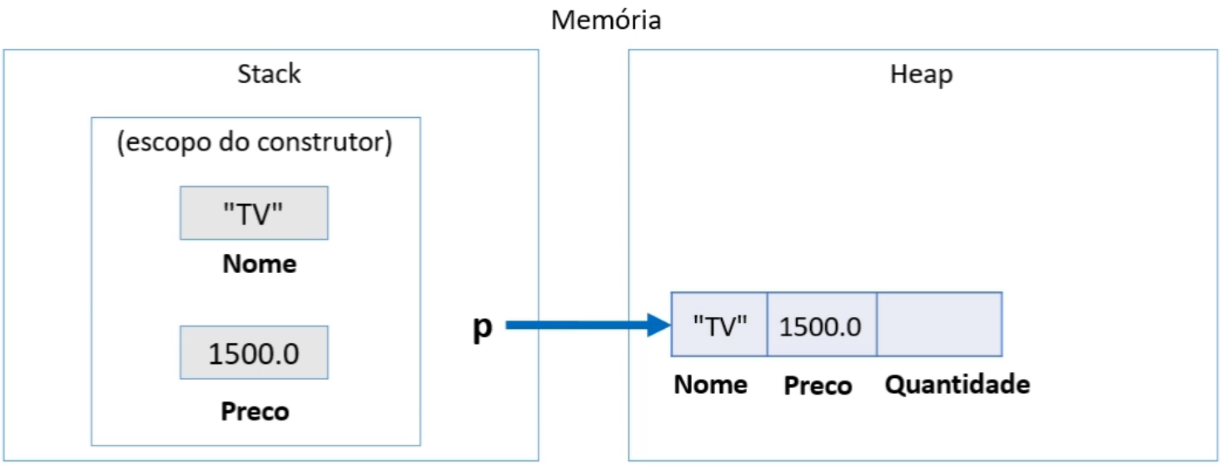
Alocação temporária na memória antes de passar para os atributos da classe.



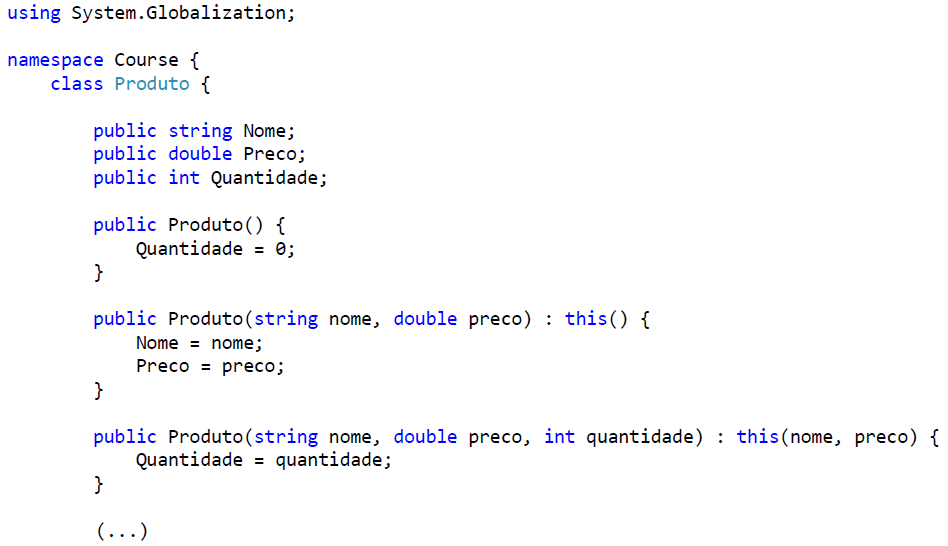
Quando passa pelo this.Nome = Nome



Quando passar pelo this.Preco = Preco

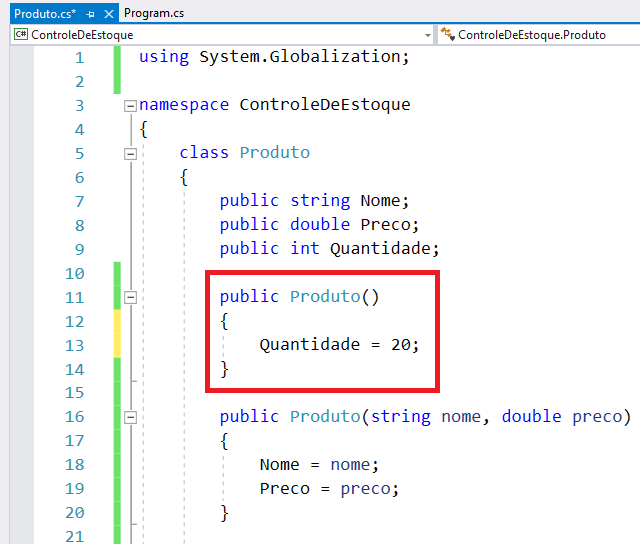


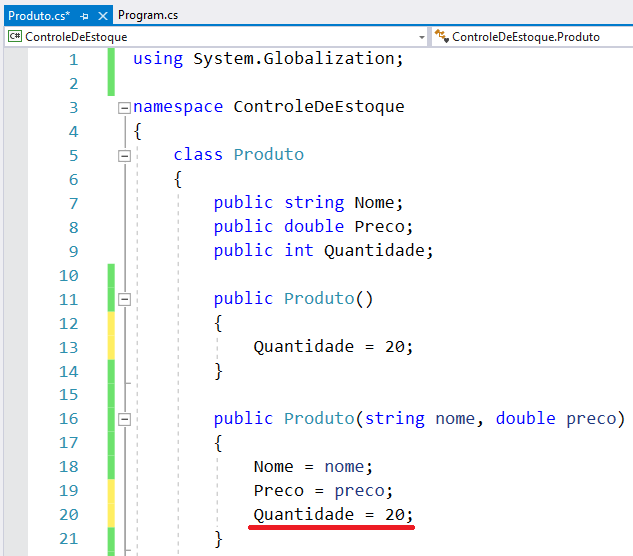
Referenciar outro construtor em um construtor



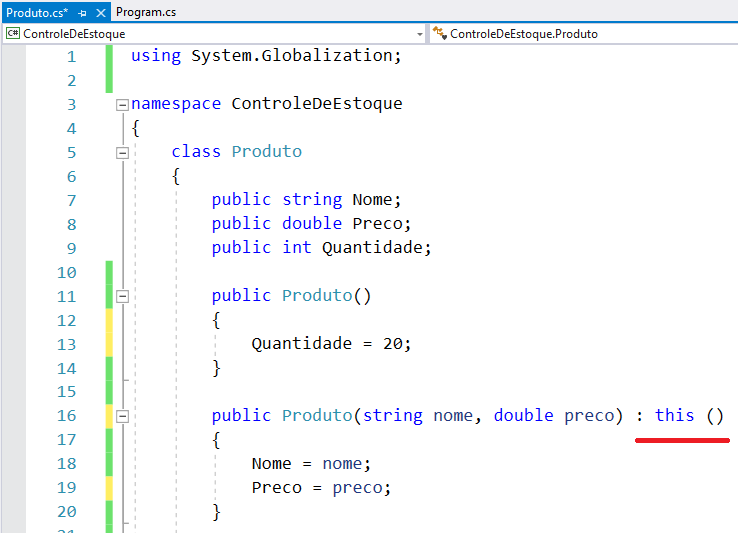
No exemplo acima foi feito a referência da Quantidade recebendo 0 (zero). Para não repetir o código, utilizamos a palavra this()

Exemplo:

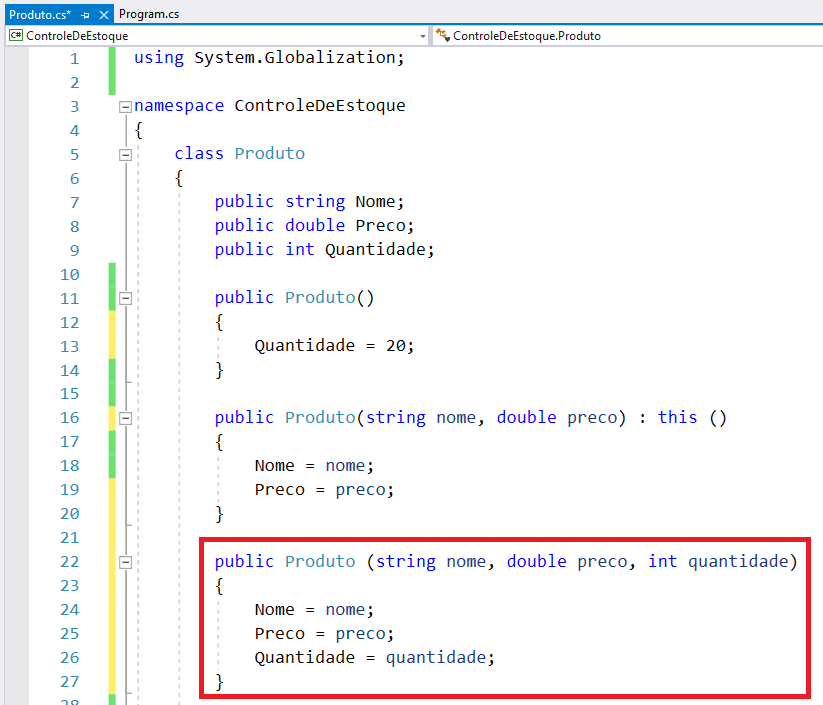


Para eu não precisar repetir a linha de código, e replicar a linha 20. 

Podemos simplesmente utilizar a palavra this assim:



Ultimo exemplo. Ao invés de ficar repetindo código:



Podemos reaproveitar o construtor

