DESENVOLVIMENTO DE SISTEMAS COM C#

Cleverson Lopes Ledur

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Compreender e utilizar o conceito de orientação a objetos em C# – II

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Criar construtores em classes e utilizá-los em objetos.
- Instanciar e manipular objetos.
- Atribuir e receber valores do objeto.

Introdução

Atualmente, é possível afirmar que a maioria dos sistemas produzidos é orientada a objetos. A orientação a objetos permitiu um grande avanço no desenvolvimento de sistemas em todo o mundo. Nesse sentido, ela possibilitou a criação de um código de fácil manutenção e com abstrações importantes para a produtividade.

O conhecimento de como manipular os objetos e desenvolver de forma adequada utilizando esses artefatos é essencial, principalmente quando se usa a linguagem de programação C#. O C# possui um conjunto muito interessante de possibilidades de instanciação e manipulação de objetos que foram desenvolvidos nos últimos anos.

Neste texto, você irá aprender a elaborar construtores em classes para possibilitar a criação e a manipulação de objetos. Também irá estudar a atribuição/leitura de dados de objetos.

Construtores

Sempre que uma classe ou estrutura é criada, seu construtor é chamado. Uma classe ou estrutura pode ter vários construtores que usam diferentes argumentos. Os construtores permitem que o programador defina valores padrões e

verificações de regras de instanciação. Também possibilitam que ele produza um código flexível e de fácil leitura (FAGERBERG, 2015).

Saiba mais

Quando você cria uma classe sem definir um construtor, o próprio C# cria um por padrão. Esse construtor padrão instancia o objeto e define variáveis de membros para os valores padrões. Se você não fornecer um construtor para sua estrutura, o C# depende de um construtor padrão implícito para inicializar automaticamente cada campo de um tipo de valor para seu valor padrão. Esse construtor padrão é definido sem a necessidade de parâmetros, então geralmente se utiliza apenas a expressão *new NomeDaClasse()* para a instanciação (MICROSOFT, 2015).

No Visual Studio, você pode criar um construtor muito facilmente por meio dos seguintes passos:

1. Clique com o botão direito do mouse em uma área em branco dentro da classe e depois clique em **Ações Rápidas e Refatorações** ou digite **Ctrl + .** (ponto-final), como mostra a Figura 1.

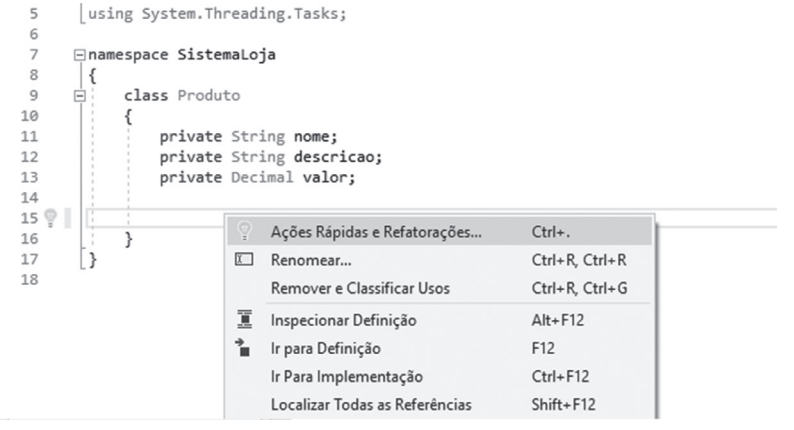

Figura 1. O primeiro passo para criar um construtor no Visual Studio.

2. Selecione **Gerar construtor** (Figura 2).

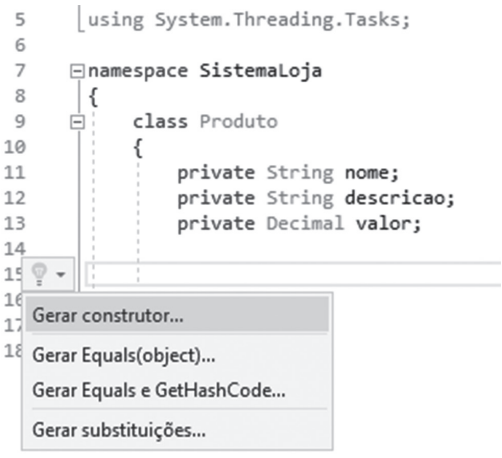

Figura 2. O segundo passo para criar um construtor no Visual Studio.

3. Na janela **Escolher membros**, selecione os membros que deseja inserir no construtor. Clique em **OK** (Figura 3).

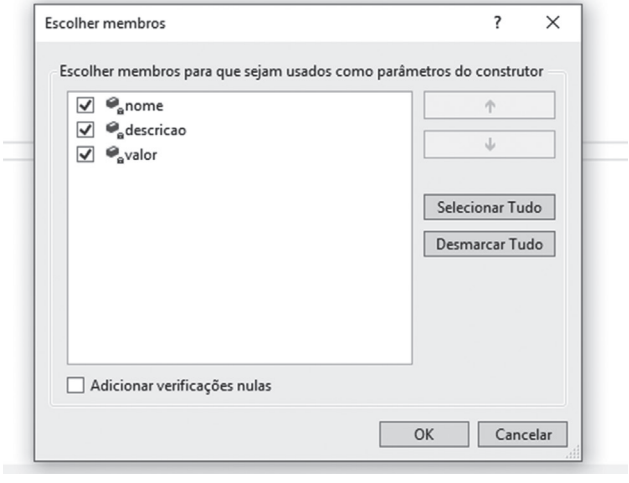

Figura 3. O terceiro passo para criar um construtor no Visual Studio.

4. Pronto! Seu construtor foi criado (Figura 4). Agora você pode modificá-lo de acordo com a lógica do seu sistema.

```
namespace SistemaLoja
{
    class Produto
    {
        private String nome;
        private String descricao;
        private Decimal valor;

        public Produto(string nome, string descricao, decimal valor)
        {
            this.nome = nome;
            this.descricao = descricao;
            this.valor = valor;
        }
    }
}
```

Figura 4. Em três passos, seu construtor fica pronto.

Sintaxe do construtor

Um construtor é um método que tem o mesmo nome que o seu tipo. Sua assinatura de método inclui apenas o nome do método e a sua lista de parâmetros.

Fique atento

Lembre-se sempre de que um construtor não inclui um tipo de retorno.

O exemplo a seguir mostra o construtor de uma classe chamada `Pessoa`.

```
public class Pessoa
{
    private string ultimoNome;
    private string primeiroNome;

    public Pessoa(string primeiroNome, string ultimoNome)
    {
        this.ultimoNome = ultimoNome;
        this.primeiroNome = primeiroNome;
    }

}
```

Se um construtor pode ser implementado como uma única declaração, você pode usar uma definição de corpo de expressão. O próximo exemplo define uma classe de localização cujo construtor possui um único parâmetro de string com o nome. A expressão atribui o argumento ao campo `nome`.

```
public class Localizacao
{
   private string nome;

   public Localizacao(string name) => this.nome = nome;

   public string Nome
   {
      get => nome;
      set => nome = value;
   }
}
```

Construtores estáticos

Até o momento, você viu todos os construtores de instâncias, que criam um novo objeto. Uma classe ou estrutura também pode ter um construtor estático, que inicializa membros estáticos do tipo. Os construtores estáticos são sem parâmetros. Se você não fornecer um construtor estático para inicializar campos estáticos, o compilador C# fornecerá um construtor estático padrão que inicializa os campos estáticos ao seu valor padrão. O exemplo a seguir usa um construtor estático para inicializar um campo estático.

```
public class Adulto : Pessoa
{
   private static int idadeMinima;

   public Adulto(string ultimoNome, string primeiroNome) : base(ultimoNome,
primeiroNome)
   { }

   static Adulto()
   {
      idadeMinima= 18;
   }
}
```

Note que, no código do último exemplo, há também a chamada do construtor da classe base, já que a classe `Adulto` estende a classe `Pessoa`. Logo, o construtor da classe pai é adicionado à classe filha. A classe filha pode reescrever esse construtor, se necessário.

Instanciando objetos

Uma definição de classe ou estrutura é como um modelo que especifica o que o tipo pode fazer. Um objeto é basicamente um bloco de memória que foi alocado e configurado de acordo com o plano. Um programa pode criar muitos objetos da mesma classe. Os objetos também são chamados de instâncias e podem ser armazenados em uma variável ou em uma matriz ou **collection**. Em uma linguagem orientada a objetos, como C #, um programa típico consiste em múltiplos objetos que interagem dinamicamente.

No exemplo a seguir, se tivesse sido criado apenas o objeto pessoa1, sem a inicialização com o operador `new`, haveria um objeto `null` (vazio).

```
public class Pessoa
{
    public string Nome{ get; set; }
    public int Idade { get; set; }
    public Pessoa(string nome, int idade)
    {
        Nome= nome;
        Idade = idade;
    }
}

class Program
{
    static void Main()
    {
        Pessoa pessoa1 = new Pessoa("Joao", 30); //Usando o construtor.

        // Outras implementações...
    }
}
```

Manipulando objetos

Como as classes são tipos de referência, uma variável de um objeto de classe contém uma referência ao endereço do objeto no heap gerenciado (MICROSOFT, 2015). Se um segundo objeto do mesmo tipo é atribuído ao primeiro objeto, então ambas as variáveis se referem ao objeto nesse endereço. Seria algo similar ao uso de ponteiros em C/C++.

As instâncias das classes são criadas usando o novo operador. Veja o exemplo a seguir de instanciação, manipulação dos objetos e atribuição/leitura de valores, onde `Pessoa` é o tipo. Já a `pessoa1` e a `pessoa2` são instâncias ou objetos desse tipo.

```
public class Pessoa
{
    public string Nome{ get; set; }
    public int Idade { get; set; }
    public Pessoa(string nome, int idade)
    {
        Nome= nome;
        Idade = idade;
    }
}

class Program
{
    static void Main ()
    {
        Pessoa pessoa1 = new Pessoa("Joao", 30); //Usando o construtor.

        Person pessoa2 = person1;
        //pessoa1 e pessoa2 apontam para o mesmo objeto na memória.

        pessoa2.Nome  = "Pedro"; //alteramos o nome
        pessoa2.Idade = 16; //alteramos a idade

        Console.WriteLine("P2 Nome: {0} Idade: {1}", pessoa2.Nome, pessoa2.Idade);
        Console.WriteLine("P1 Nome: {0} Idade: {1}", pessoa1.Nome, pessoa1.Idade);
        Console.ReadKey();
    }
}
/*
    Saída:
        P2 Nome: Pedro Idade: 16
        P1 Nome: Pedro Idade: 16
*/
```

No exemplo mostrado, você pode ver claramente o uso de objetos em C#. Nesse exemplo, ocorreu a criação de dois objetos e também foram manipulados os valores por meio de *getters* e *setters*.

Referências

FAGERBERG, J. *C# for beginners*: the tactical guidebook-learn CSharp by coding. [S.l.]: CreateSpace Independent, 2015.

MICROSOFT. *Object and collection initializers (C# programming guide)*. Redmond, 2015. Disponível em: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/object-and-collection-initializers>. Acesso em: 02 nov. 2017.

Leitura recomendada

SHARP, J. *Microsoft Visual C# 2013*: Passo a passo. Porto Alegre: Bookman, 2014.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS