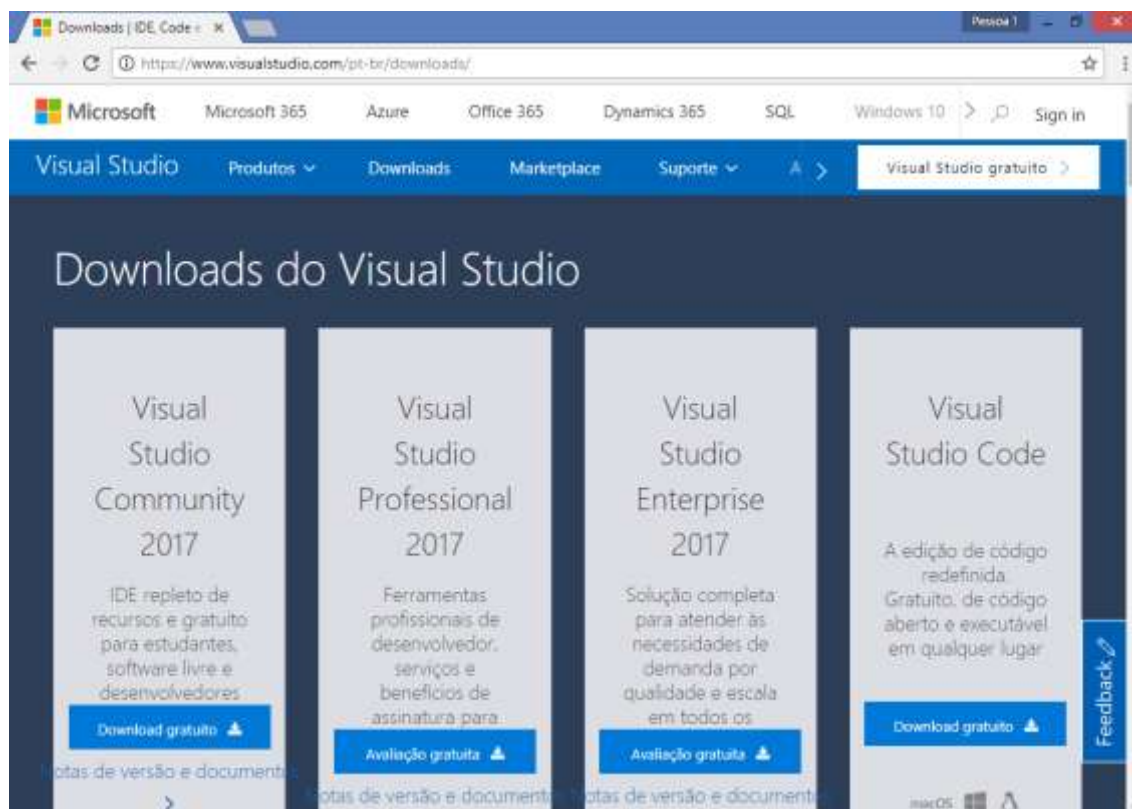


## VisualStudio 2017 Community

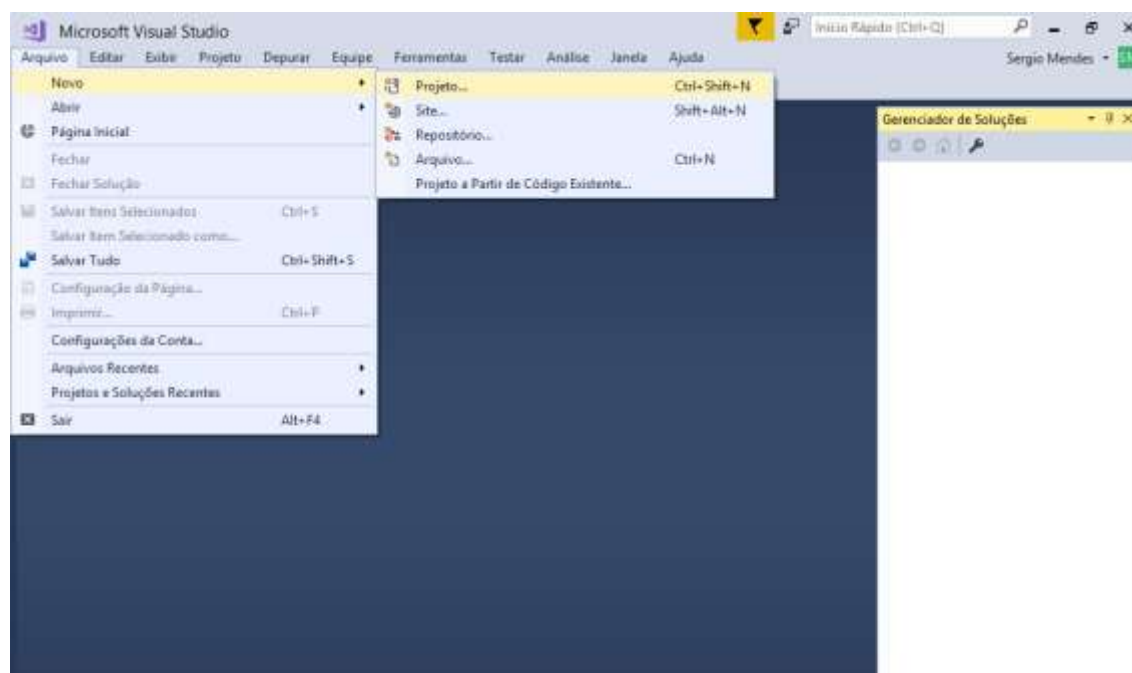
IDE: Ambiente de desenvolvimento

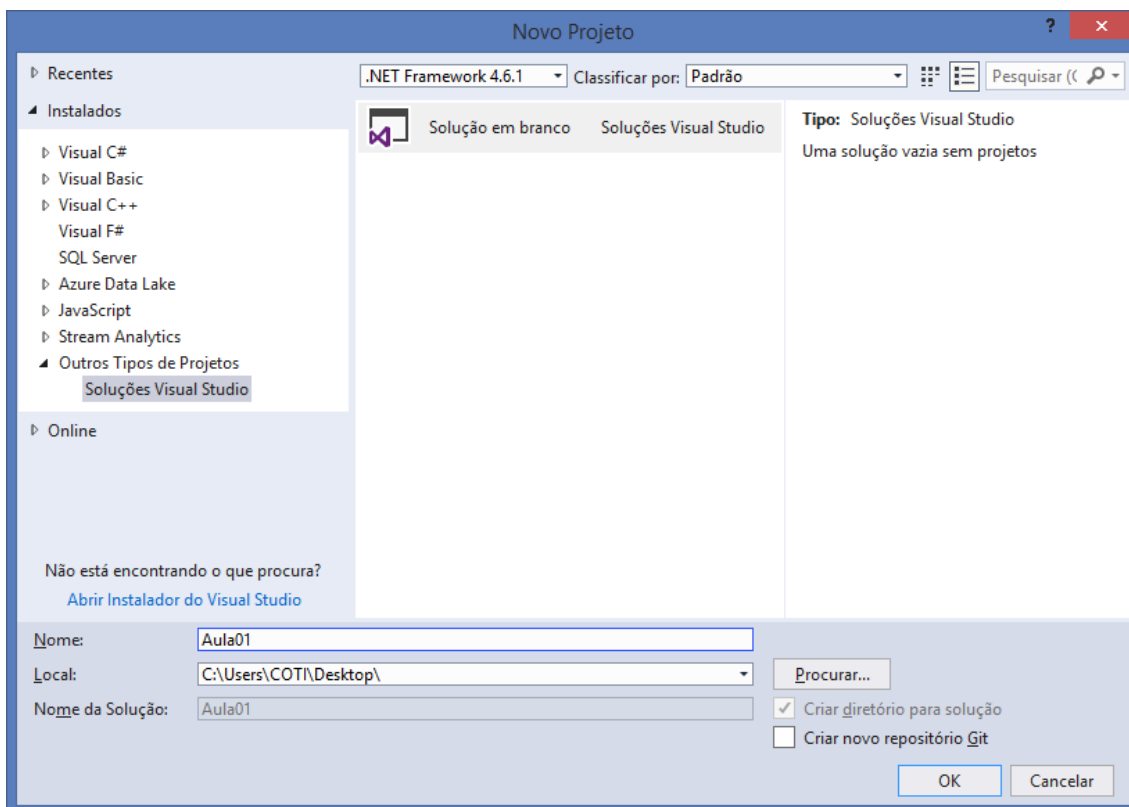
<https://www.visualstudio.com/pt-br/downloads/>



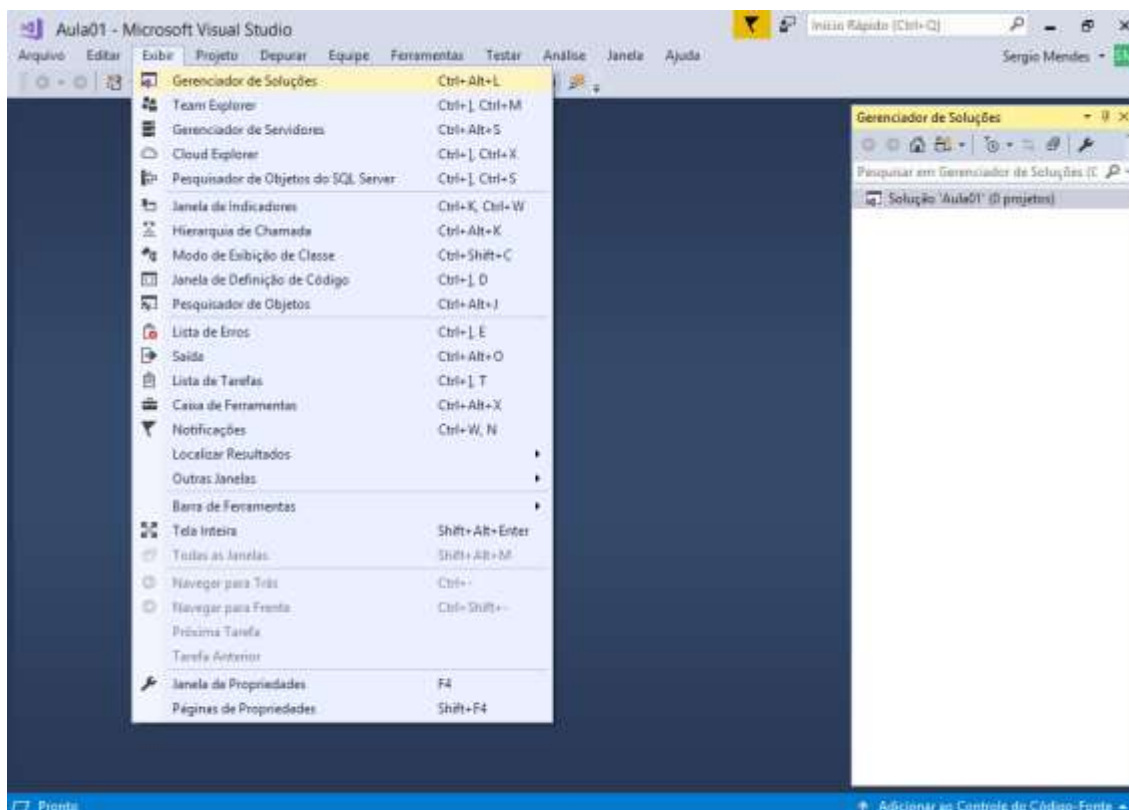
### Primeiro passo:

Criando uma solution (pasta de trabalho)



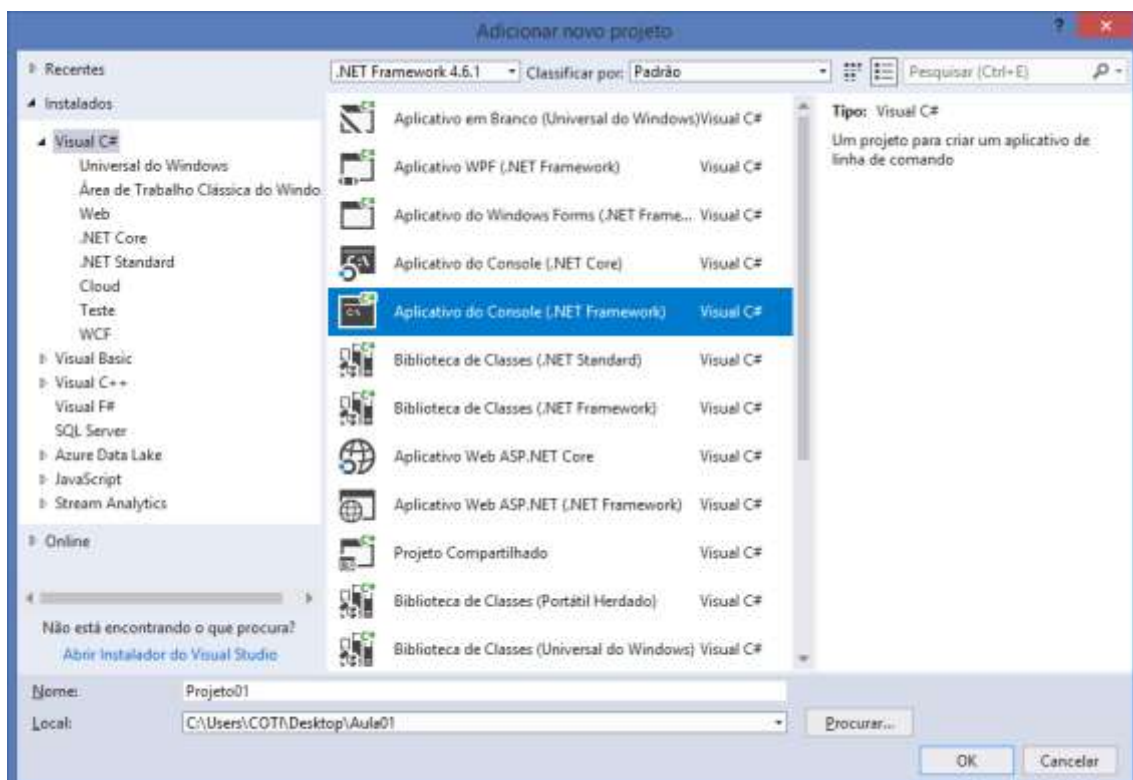
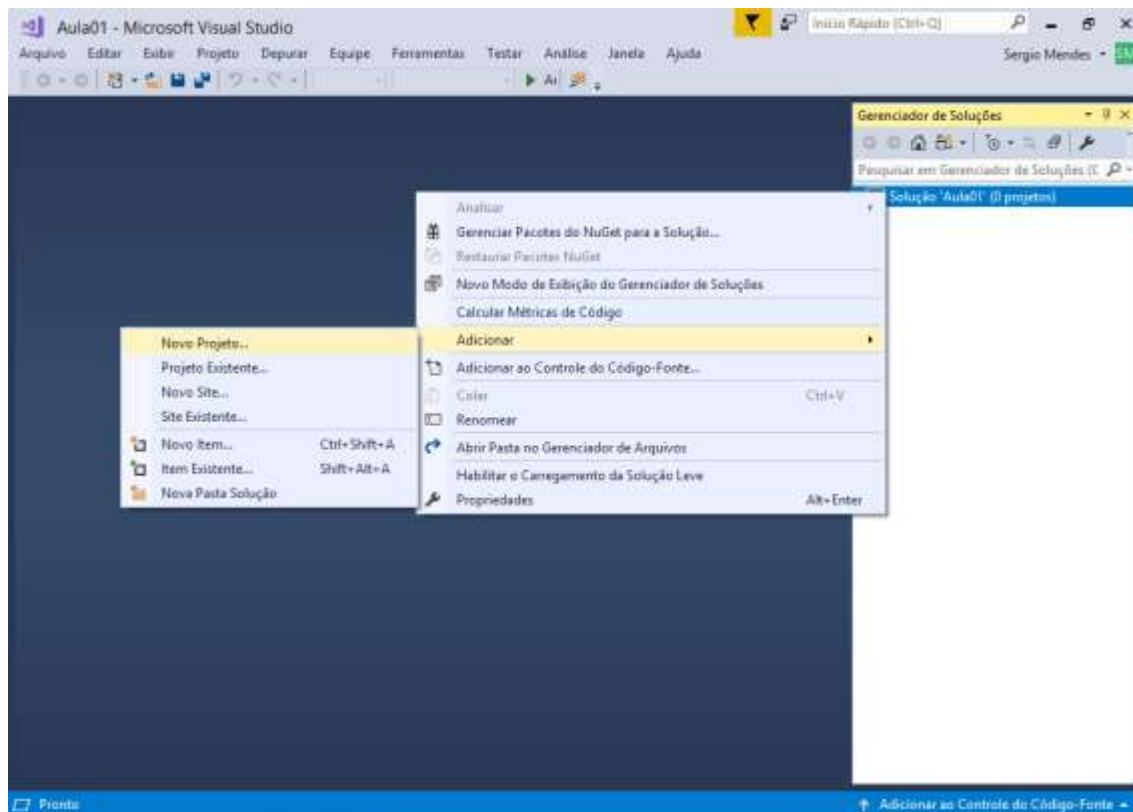


Exibindo a solution criada:  
Exibir / Gerenciador de Soluções



## Console Application

Prompt de comando (DOS)



## Program.cs

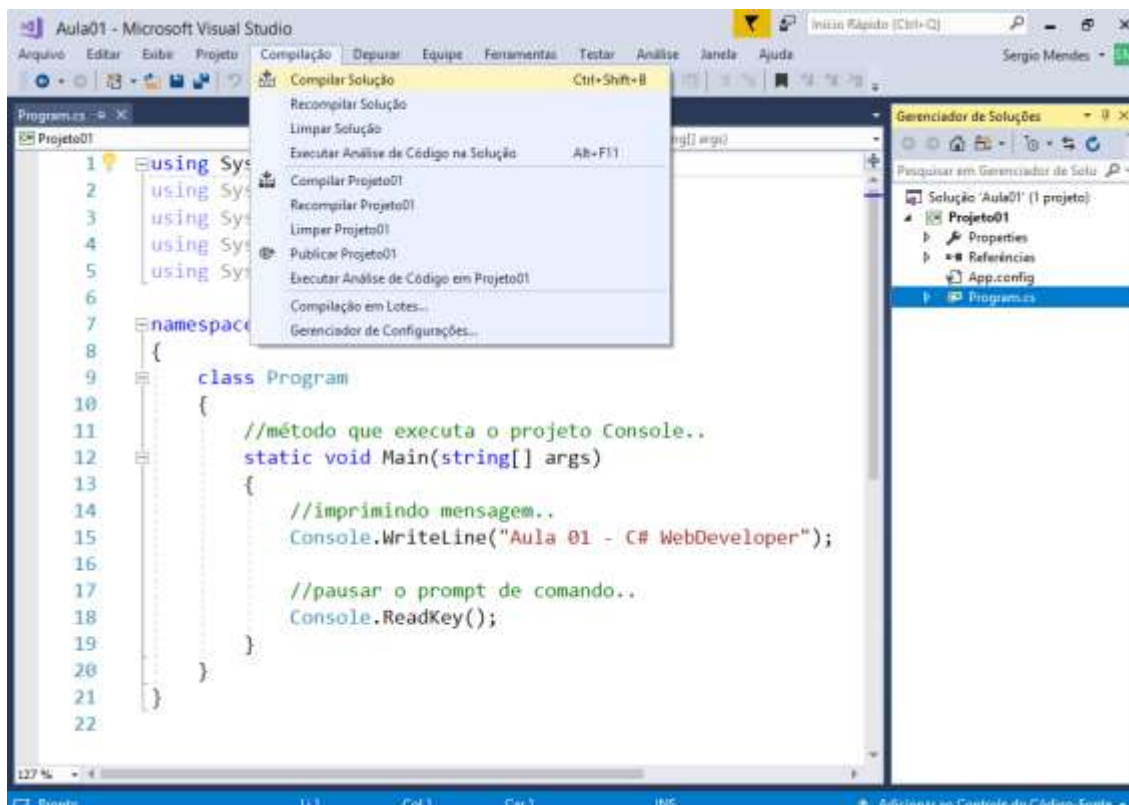
Classe principal do projeto Console, pois é onde está declarado o método Main() do projeto. Este método é disparado quando o projeto for executado.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

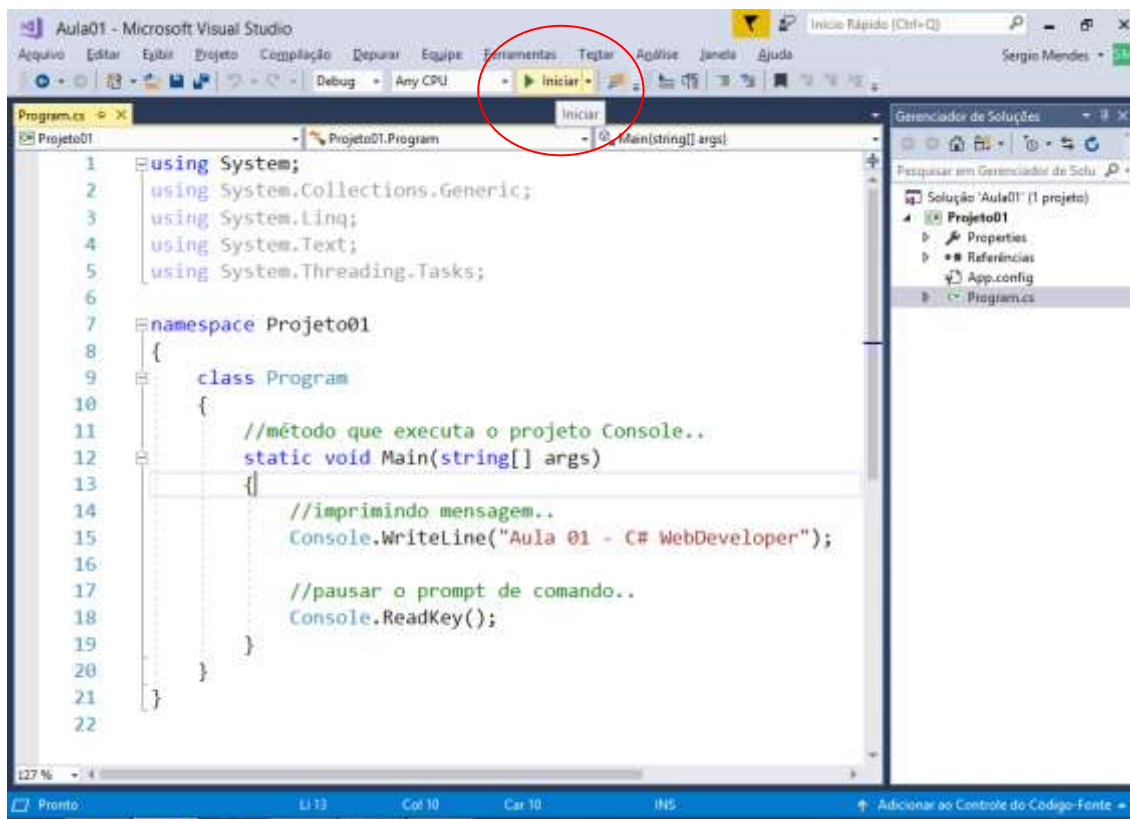
namespace Projeto01
{
    class Program
    {
        //método que executa o projeto Console..
        static void Main(string[] args)
        {
            //imprimindo mensagem..
            Console.WriteLine("Aula 01 - C# WebDeveloper");

            //pausar o prompt de comando..
            Console.ReadKey();
        }
    }
}
```

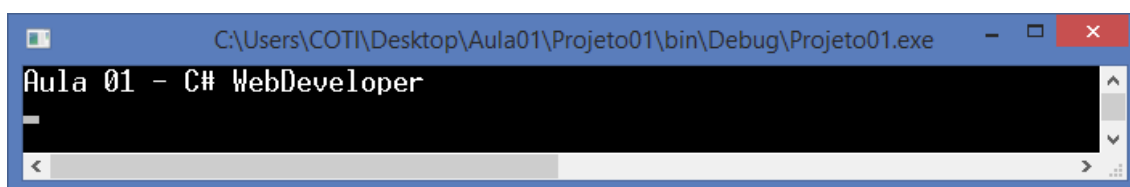
## Compilando o projeto:



**Para executar: F5**



Resultado:



## POO - Programação Orientada a Objetos

### Classe

Estrutura de programação Orientada a Objetos composta basicamente de atributos (dados) e métodos (funções, rotinas) e podem ser utilizadas para resolver qualquer tipo de aspecto do sistema.

Por exemplo, podemos criar classes para:

- Modelar entidades do projeto
- Rotinas de acesso a banco de dados
- Regras de negócio
- Testes
- etc...



## Criando uma classe de entidade

Este tipo de classe é uma das importantes em um sistema e tem como objetivo modelar e representar os objetos que são entendidos como "entidades" (substantivos) que compoem o dominio do projeto.

Exemplo:

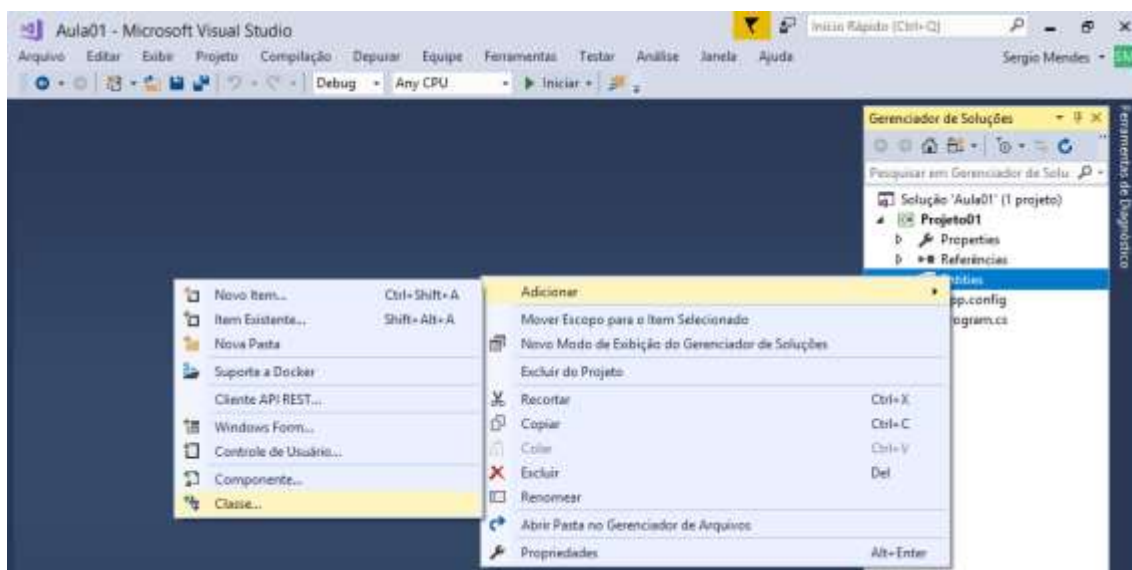
### Sistema de controle de falculdade

Entidades: Aluno, Professor, Turma, Curso, Matricula, etc...

### Sistema de gestão de projetos

Entidades: Funcionario, Projeto, Equipe, Gerente, etc...

Criando uma classe de entidade: Produto

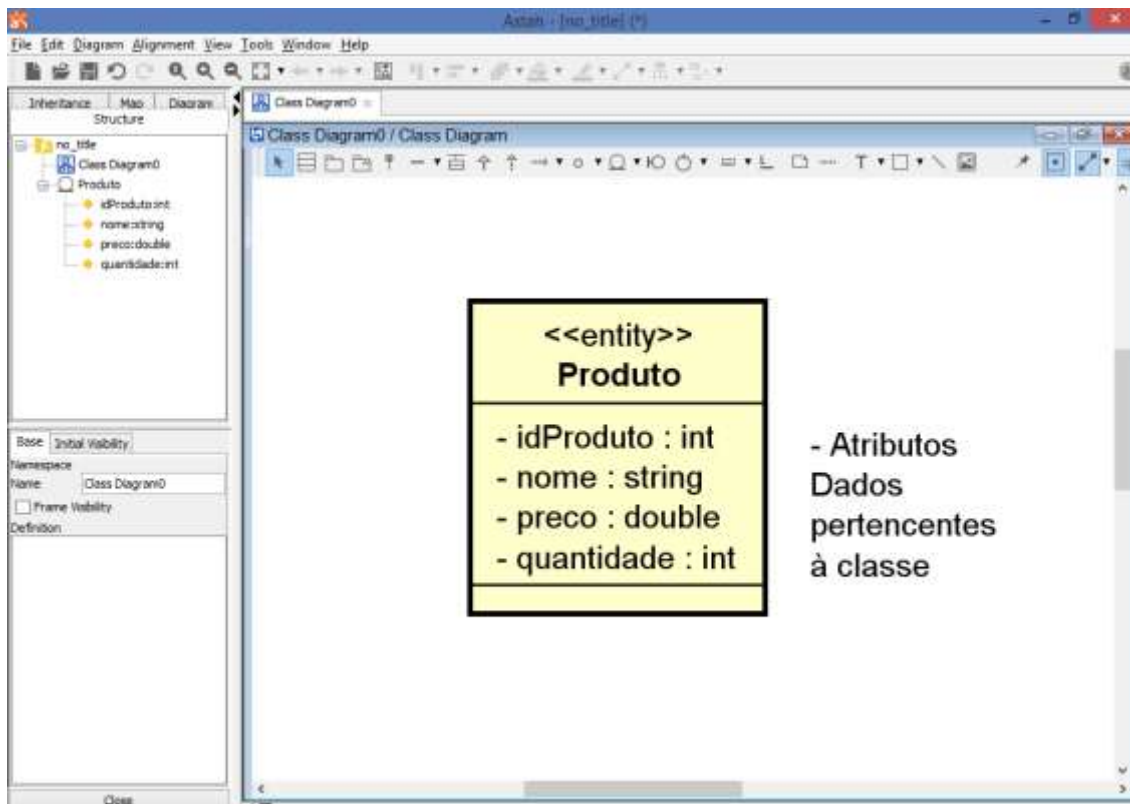


## UML - Unified Modeling Language

Linguagem visual para diagramação, documentação e especificação de sistemas orientados a objetos.

### - Diagrama de Classes

Representa as classes e suas relações em um projeto OO.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

//namespace -> local da classe dentro do projeto..
namespace Projeto01.Entities
{
    //public -> acesso total
    public class Produto
    {
        //atributos..
        //private -> não permite acesso, um elemento
        //do tipo private só pode ser acessado dentro
        //da propria classe onde esta declarado..
        private int idProduto;
        private string nome;
        private double preco;
        private int quantidade;
    }
}
```

## Modificadores de visibilidade:

<b>public</b>	Permite acesso total ao elemento
<b>protected</b>	Permite acesso por meio de herança
<b>internal</b>	Permite acesso dentro do arquivo ou namespace
<b>private</b>	Só permite acesso dentro da propria classe

## Objeto

Consiste de uma variavel criada a partir de uma classe, atraves do objeto temos acesso ao conteudo da classe, ou seja, seus atributos e métodos (desde que estes tenham visibilidade para aser acessado)

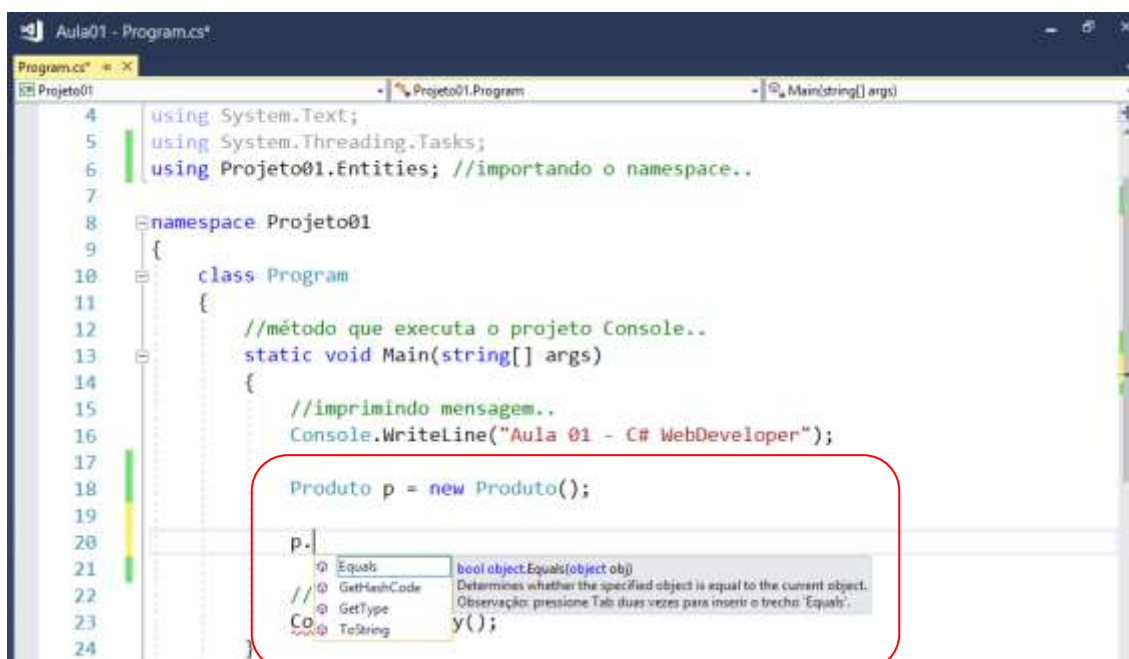
**Produto p = new Produto();**

[Classe]      [Objeto]      [Espaço de memória (Inicializando)]

### Observação:

Toda **classe criada em C# é HERANÇA Object**, portanto qualquer objeto criado para qualquer classe terá por herança os métodos abaixo:

- Equals
- GetHashCode
- GetType
- ToString



```

4  using System.Text;
5  using System.Threading.Tasks;
6  using Projeto01.Entities; //importando o namespace..
7
8  namespace Projeto01
9  {
10     class Program
11     {
12         //método que executa o projeto Console..
13         static void Main(string[] args)
14         {
15             //imprimindo mensagem..
16             Console.WriteLine("Aula 01 - C# WebDeveloper");
17
18             Produto p = new Produto();
19
20             p.
21             //
22             Co
23             ToString
24         }
25     }
26 }

```



## Encapsulamento

Tem como objetivo proteger o conteúdo de uma classe do acesso externo. Um exemplo ocorre quando declaramos os atributos de uma classe como privados e criamos métodos públicos (funções) que realizam a entrada e saída de dados para cada atributo privado.

Estes métodos são universalmente chamados de **set** e **get**

- set (entrada)
- get (saída)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

//namespace -> local da classe dentro do projeto..
namespace Projeto01.Entities
{
    //public -> acesso total
    public class Produto
    {
        //atributos..
        //private -> não permite acesso, um elemento
        //do tipo private só pode ser acessado dentro
        //da propria classe onde esta declarado..
        private int idProduto;
        private string nome;
        private double preco;
        private int quantidade;

        //Métodos..
        //Estes métodos serão para encapsular os atributos,
        //ou seja, permitir entrada (set) / saída (get)
        public int IdProduto
        {
            set { idProduto = value; } //entrada
            get { return idProduto; } //saída
        }

        public string Nome
        {
            set { nome = value; } //entrada
            get { return nome; } //saída
        }

        public double Preco
        {
            set { preco = value; } //entrada
            get { return preco; } //saída
        }

        public int Quantidade
        {
            set { quantidade = value; } //entrada
            get { return quantidade; } //saída
        }
    }
}
```

## Voltando na classe Program:

Implementando entrada de dados para a entidade Produto:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto01.Entities; //importando o namespace..

namespace Projeto01
{
    class Program
    {
        //método que executa o projeto Console..
        static void Main(string[] args)
        {
            //imprimindo mensagem..
            Console.WriteLine("Aula 01 - C# WebDeveloper");

            //declarando um objeto para a classe Produto..
            Produto p = new Produto();

            Console.WriteLine("\n - CONTROLE DE PRODUTOS - \n");

            Console.Write("Id do Produto.....: ");
            p.IdProduto = int.Parse(Console.ReadLine());

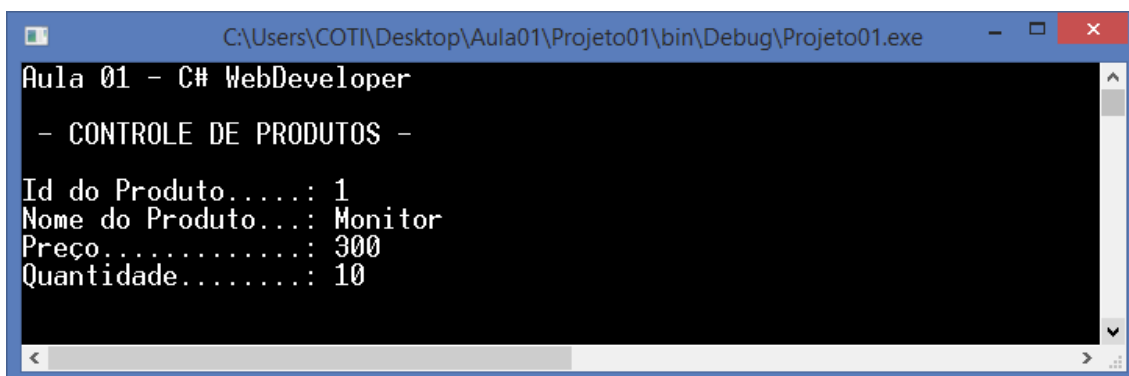
            Console.Write("Nome do Produto....: ");
            p.Nome = Console.ReadLine();

            Console.Write("Preço.....: ");
            p.Preco = double.Parse(Console.ReadLine());

            Console.Write("Quantidade.....: ");
            p.Quantidade = int.Parse(Console.ReadLine());

            //pausar o prompt de comando..
            Console.ReadKey();
        }
    }
}
```

## Executando:



```
C:\Users\COTI\Desktop\Aula01\Projeto01\bin\Debug\Projeto01.exe
Aula 01 - C# WebDeveloper
- CONTROLE DE PRODUTOS -
Id do Produto.....: 1
Nome do Produto....: Monitor
Preço.....: 300
Quantidade.....: 10
```

## Imprimindo:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto01.Entities; //importando o namespace..

namespace Projeto01
{
    class Program
    {
        //método que executa o projeto Console..
        static void Main(string[] args)
        {
            //imprimindo mensagem..
            Console.WriteLine("Aula 01 - C# WebDeveloper");

            //declarando um objeto para a classe Produto..
            Produto p = new Produto();

            Console.WriteLine("\n - CONTROLE DE PRODUTOS - \n");

            Console.Write("Id do Produto.....: ");
            p.IdProduto = int.Parse(Console.ReadLine());

            Console.Write("Nome do Produto...: ");
            p.Nome = Console.ReadLine();

            Console.Write("Preço.....: ");
            p.Preco = double.Parse(Console.ReadLine());

            Console.Write("Quantidade.....: ");
            p.Quantidade = int.Parse(Console.ReadLine());

            //imprimindo..
            Console.WriteLine("\n - DADOS DO PRODUTO - \n");
            Console.WriteLine("Id do Produto.: " + p.IdProduto);
            Console.WriteLine("Nome.....: " + p.Nome);
            Console.WriteLine("Preço.....: " + p.Preco);
            Console.WriteLine("Quantidade.....: " + p.Quantidade);

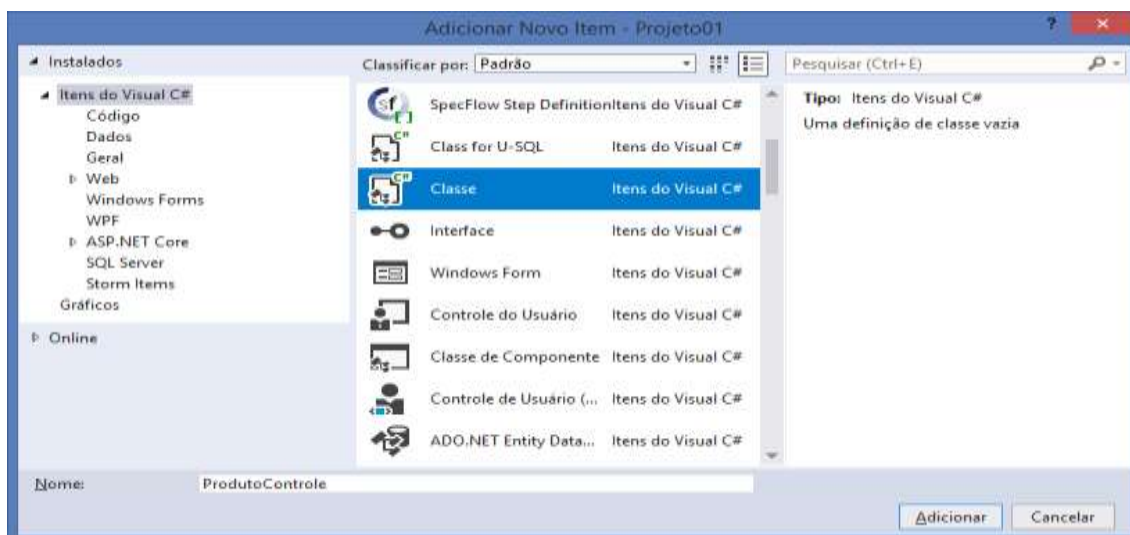
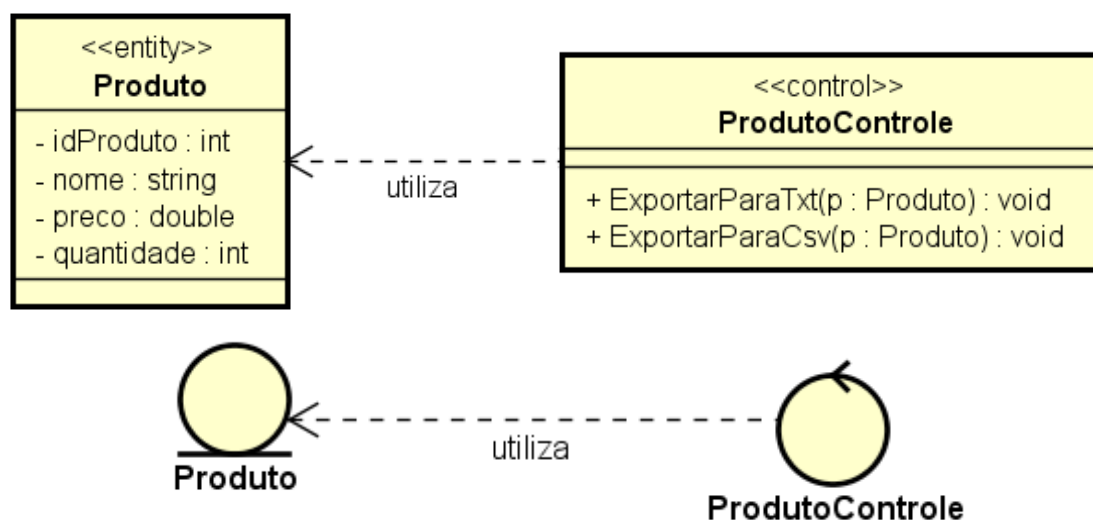
            //pausar o prompt de comando..
            Console.ReadKey();
        }
    }
}
```

Executando:

```

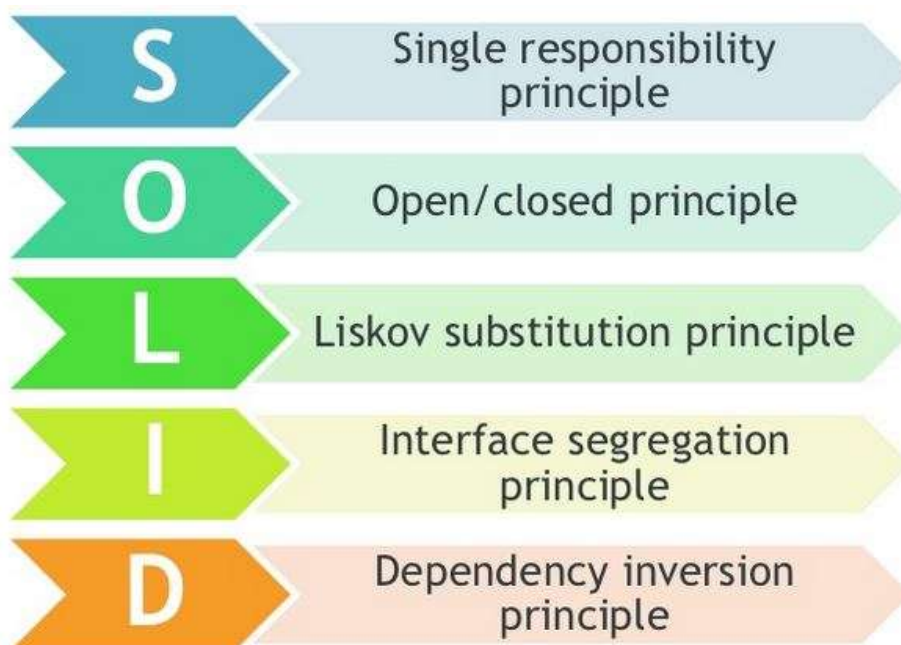
C:\Users\COTI\Desktop\Aula01\Projeto01\bin\Debug\Projeto01.exe
Aula 01 - C# WebDeveloper
- CONTROLE DE PRODUTOS -
Id do Produto.....: 1
Nome do Produto...: Monitor
Preço.....: 500
Quantidade.....: 10
- DADOS DO PRODUTO -
Id do Produto.: 1
Nome.....: Monitor
Preço.....: 500
Quantidade....: 10
  
```

Criando uma classe para gravar os dados do produto em arquivo:



## SOLID

Conjunto de 5 boas princípios para desenvolvimento Orientado a Objetos.



## SRP - Princípio de Responsabilidade Unica

Define que cada classe deve ter 1 unica responsabilidade, para que assim cada classe possa manter a sua **coesão**.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto01.Entities; //importando..
using System.IO; //manipulação de arquivos..

namespace Projeto01.Controles
{
    public class ProdutoControle
    {
        //método para exportar os dados do produto
        //para um arquivo do tipo .txt
        public void ExportarParaTxt(Produto p)
        {
            //classe para gravação de arquivos..
            StreamWriter sw = new StreamWriter("c:\\temp\\produtos.txt");

            //escrever os dados do produto..
            sw.WriteLine("Id do Produto...: " + p.IdProduto);
            sw.WriteLine("Nome.....: " + p.Nome);
        }
    }
}
```



```

sw.WriteLine("Quantidade.....: " + p.Quantidade);
sw.WriteLine("Preço.....: " + p.Preco);
sw.WriteLine("...");

//fechando o arquivo..
sw.Close();
}

//método para exportar os dados do produto
//para um arquivo do tipo .csv
public void ExportarParaCsv(Produto p)
{
    //classe para gravação de arquivos..
    StreamWriter sw = new StreamWriter("c:\\temp\\produtos.csv");

    sw.WriteLine("{0};{1};{2};{3}",
        p.IdProduto, p.Nome, p.Preco, p.Quantidade);

    //fechando o arquivo..
    sw.Close();
}
}
}

```

## Executando:

Voltando ao método Main()

**ProdutoControle pc = new ProdutoControle();**  
 [Classe - Tipo]      [Objeto]    [Espaço de Memória]

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto01.Entities; //importando o namespace..
using Projeto01.Controles; //importando o namespace..

namespace Projeto01
{
    class Program
    {
        //método que executa o projeto Console..
        static void Main(string[] args)
        {
            //imprimindo mensagem..
            Console.WriteLine("Aula 01 - C# WebDeveloper");

            //declarando um objeto para a classe Produto..
            Produto p = new Produto();

            Console.WriteLine("\n - CONTROLE DE PRODUTOS - \n");
        }
    }
}

```

```
Console.Write("Id do Produto.....: ");
p.IdProduto = int.Parse(Console.ReadLine());

Console.Write("Nome do Produto...: ");
p.Nome = Console.ReadLine();

Console.Write("Preço.....: ");
p.Preco = double.Parse(Console.ReadLine());

Console.Write("Quantidade.....: ");
p.Quantidade = int.Parse(Console.ReadLine());

//imprimindo..
Console.WriteLine("\n - DADOS DO PRODUTO - \n");
Console.WriteLine("Id do Produto.: " + p.IdProduto);
Console.WriteLine("Nome.....: " + p.Nome);
Console.WriteLine("Preço.....: " + p.Preco);
Console.WriteLine("Quantidade.....: " + p.Quantidade);

Console.Write("\nInforme 1(TXT) ou 2(CSV)...: ");
int opcao = int.Parse(Console.ReadLine());

//criando um objeto para a classe de controle de produto..
ProdutoControle pc = new ProdutoControle();

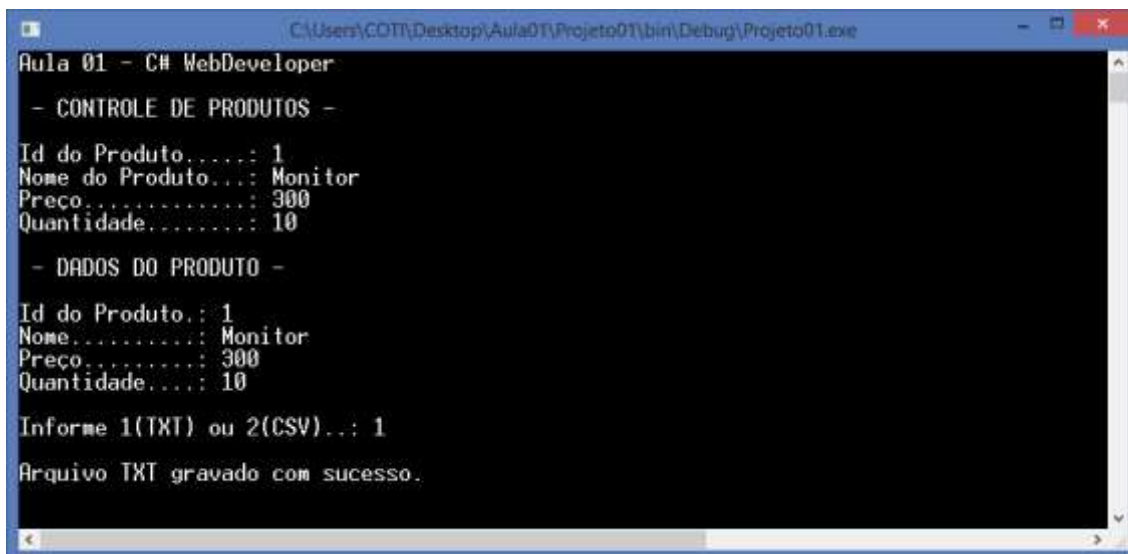
//testando..
switch(opcao)
{
    case 1:
        pc.ExportarParaTxt(p);
        Console.WriteLine("\nArquivo TXT gravado com sucesso.");
        break;

    case 2:
        pc.ExportarParaCsv(p);
        Console.WriteLine("\nArquivo CSV gravado com sucesso.");
        break;

    default:
        Console.WriteLine("\nOpção inválida.");
        break;
}

//pausar o prompt de comando..
Console.ReadKey();
}
}
}
```

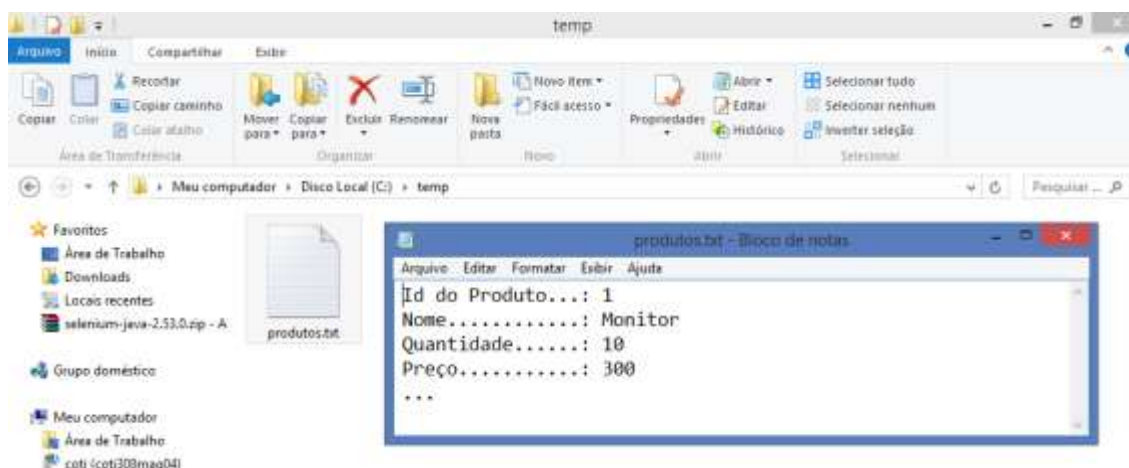
## Executando:



```

C:\Users\COTI\Desktop\Aula01\Projeto01\bin\Debug\Projeto01.exe
Aula 01 - C# WebDeveloper
- CONTROLE DE PRODUTOS -
Id do Produto.....: 1
Nome do Produto...: Monitor
Preço.....: 300
Quantidade.....: 10
- DADOS DO PRODUTO -
Id do Produto.: 1
Nome.....: Monitor
Preço.....: 300
Quantidade....: 10
Informe 1(TXT) ou 2(CSV)...: 1
Arquivo TXT gravado com sucesso.
  
```

## Arquivo gerado:



Alterando a implementação do StreamWriter para que o arquivo não seja sobrescrito mas sim incluído conteúdo ao final do arquivo:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto01.Entities; //importando..
using System.IO; //manipulação de arquivos..

namespace Projeto01.Controles
{
  
```

```
public class ProdutoControle
{
    //método para exportar os dados do produto
    //para um arquivo do tipo .txt
    public void ExportarParaTxt(Produto p)
    {
        //classe para gravação de arquivos..
        StreamWriter sw = new StreamWriter("c:\\temp\\produtos.txt", true);

        //escrever os dados do produto..
        sw.WriteLine("Id do Produto...: " + p.IdProduto);
        sw.WriteLine("Nome.....: " + p.Nome);
        sw.WriteLine("Quantidade.....: " + p.Quantidade);
        sw.WriteLine("Preço.....: " + p.Preco);
        sw.WriteLine("...");

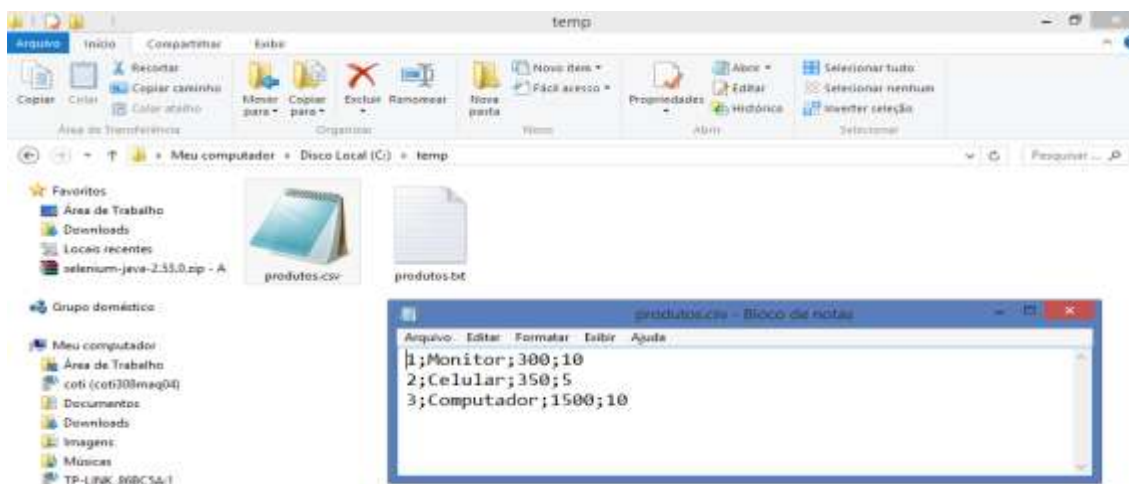
        //fechando o arquivo..
        sw.Close();
    }

    //método para exportar os dados do produto
    //para um arquivo do tipo .csv
    public void ExportarParaCsv(Produto p)
    {
        //classe para gravação de arquivos..
        StreamWriter sw = new StreamWriter("c:\\temp\\produtos.csv", true);

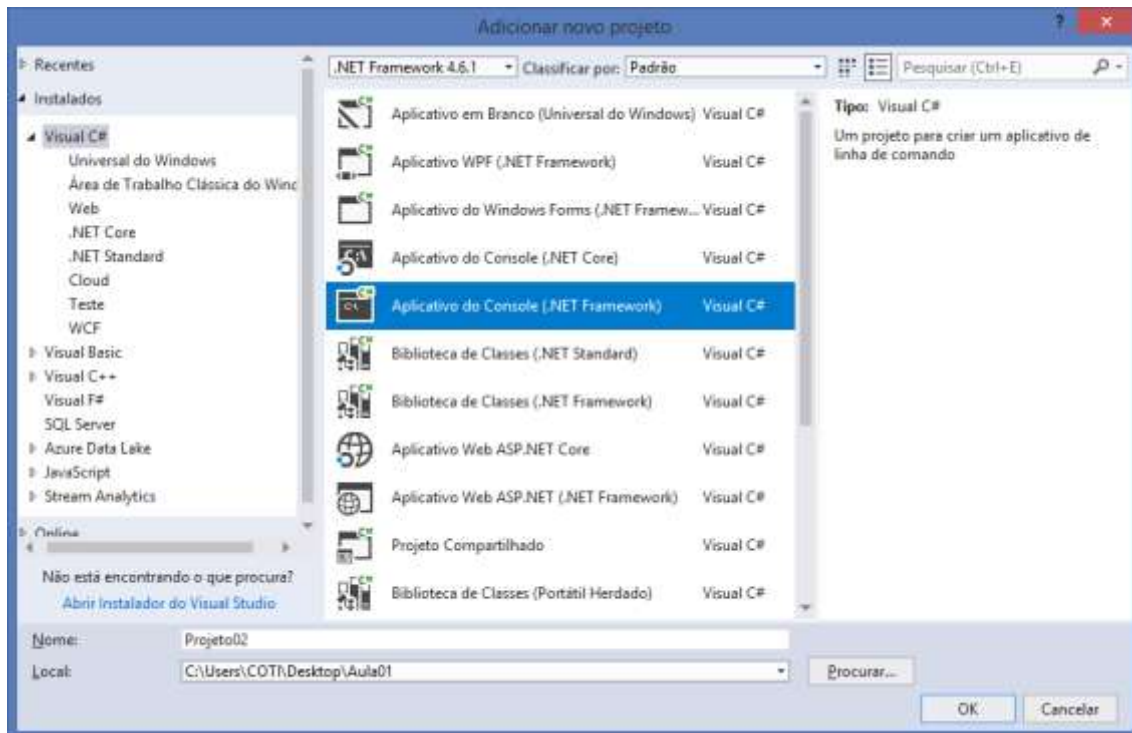
        sw.WriteLine("{0};{1};{2};{3}",
            p.IdProduto, p.Nome, p.Preco, p.Quantidade);

        //fechando o arquivo..
        sw.Close();
    }
}
```

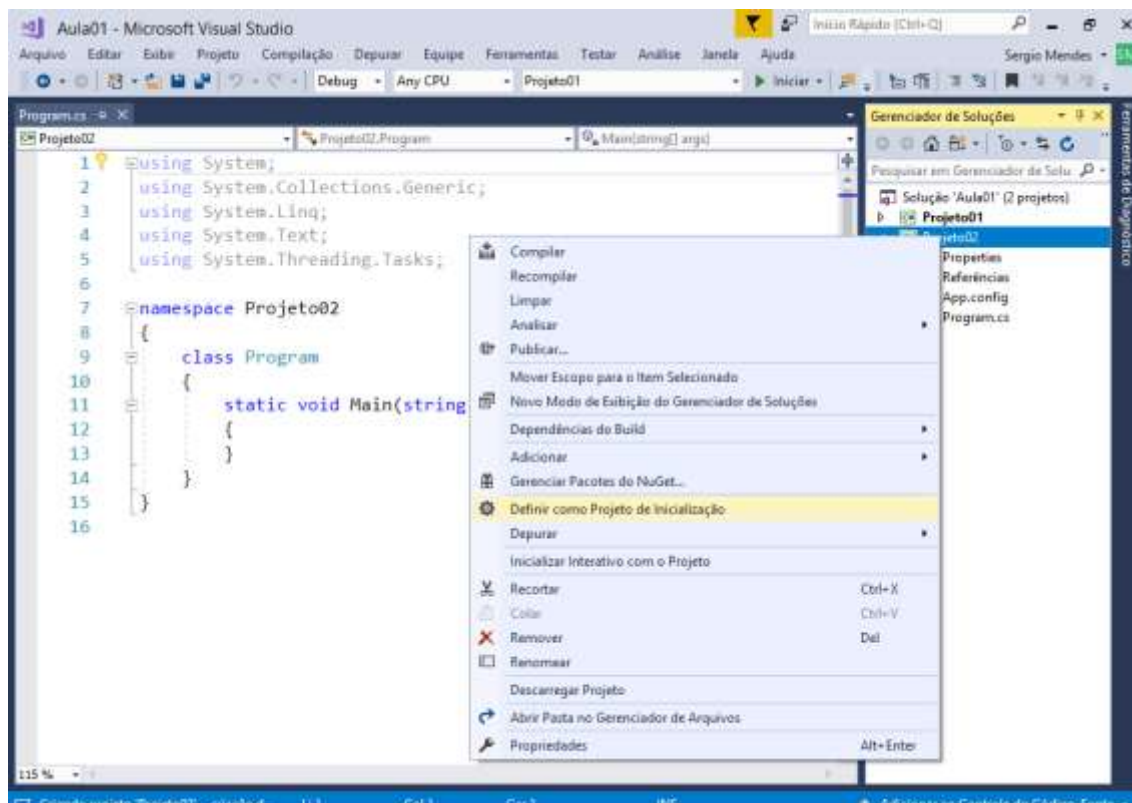
## Executando:



## Novo projeto:



## Definindo o projeto principal da solution: **Definir como projeto de inicialização**



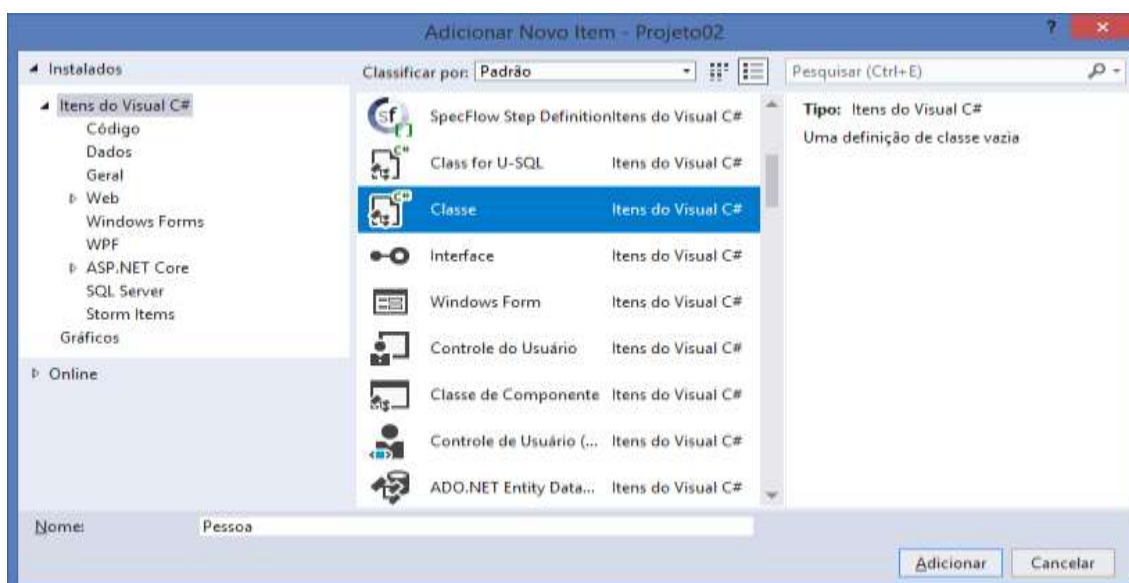
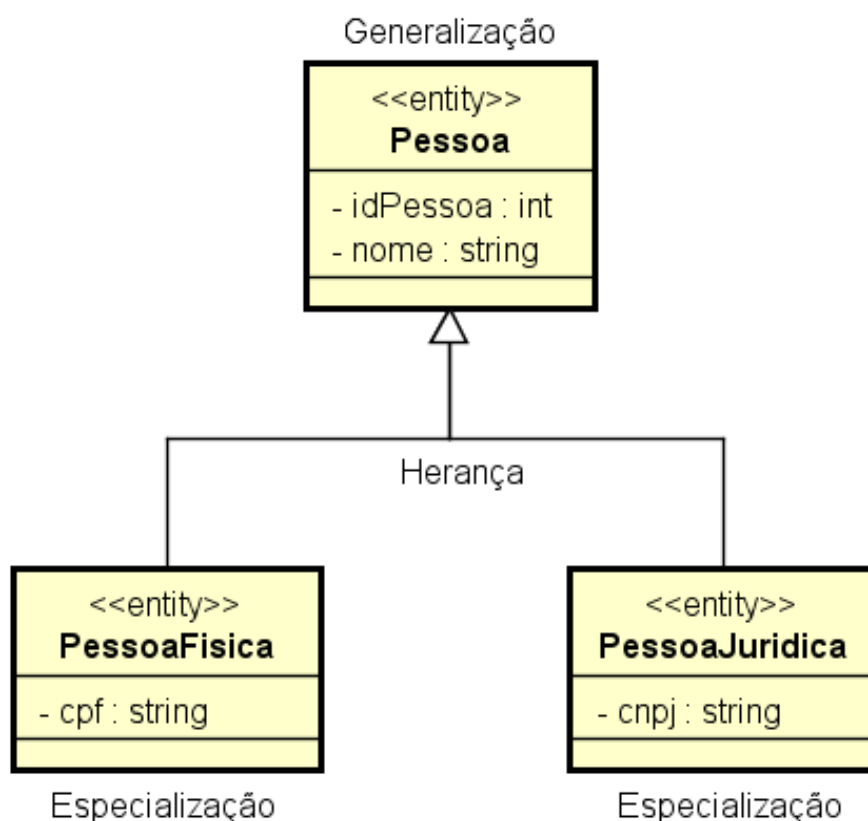


## Relacionamentos entre classes

Em Orientação a Objetos, podemos relacionar classes, basicamente, de 2 maneiras: **Herança (SER)** ou **Associação (TER)**

### Herança (SER)

Tipo de relacionamento entre classes que define uma relação de hierarquia, ou seja, superclasse e subclasses, também é chamada de **generalização / especialização**



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto02.Entities
{
    public class Pessoa
    {
        #region Atributos

        private int idPessoa;
        private string nome;

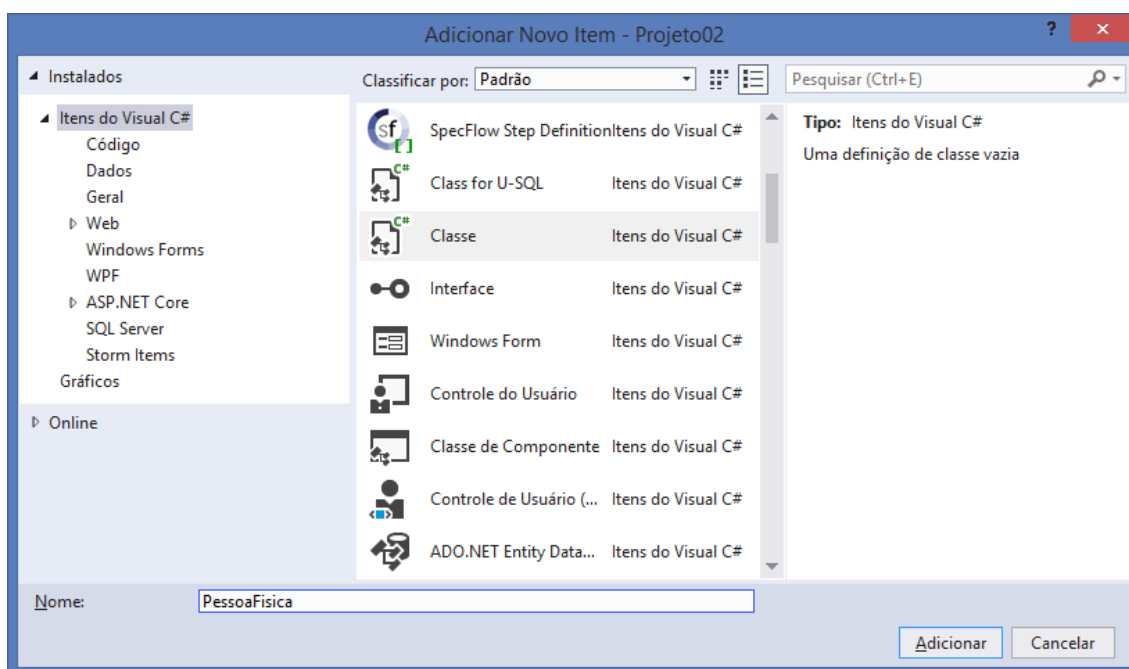
        #endregion

        #region Métodos de Encapsulamento

        public int IdPessoa
        {
            set { idPessoa = value; }
            get { return idPessoa; }
        }

        public string Nome
        {
            set { nome = value; }
            get { return nome; }
        }

        #endregion
    }
}
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto02.Entities
{
    //PessoaFisica É-UMA Pessoa (Herança)
    public class PessoaFisica : Pessoa
    {
        #region Atributos

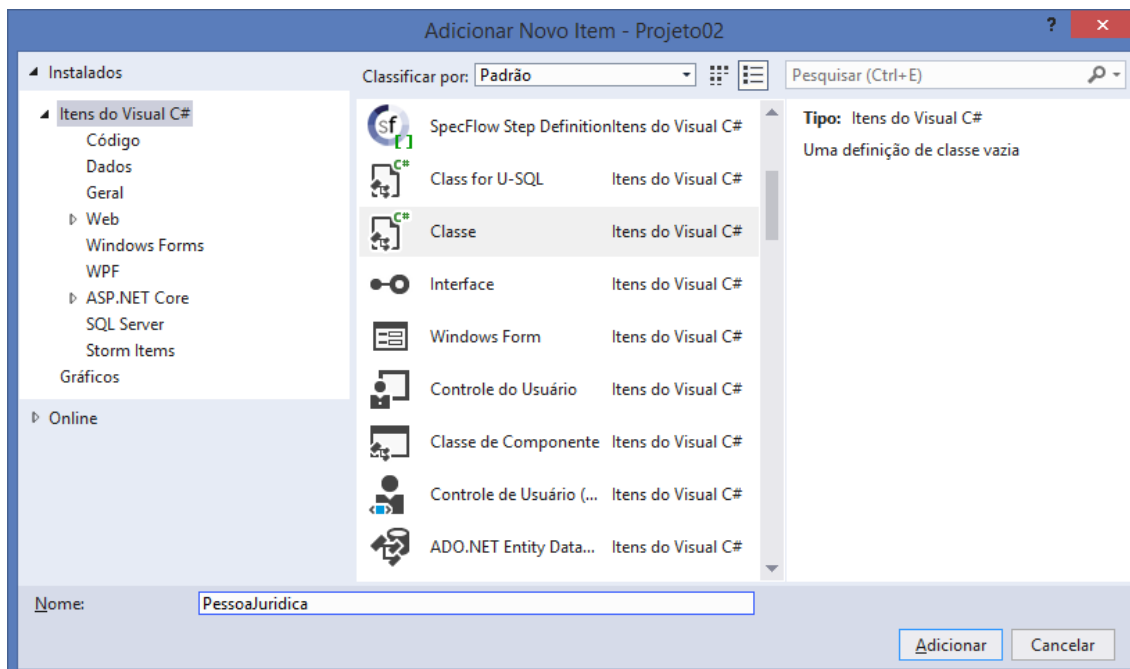
        private string cpf;

        #endregion

        #region Métodos de Encapsulamento

        public string Cpf
        {
            set { cpf = value; }
            get { return cpf; }
        }

        #endregion
    }
}
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto02.Entities
{
```



# C#.NET WebDeveloper

## Terça-feira, 09 de Janeiro de 2018

Orientação a Objetos, Classes, atributos e métodos.  
Encapsulamento, herança, Construtores e Sobrecarga de Métodos.

Aula  
**01**

```
//PessoaJuridica É-UMA Pessoa (Herança)
public class PessoaJuridica : Pessoa
{
    #region Atributos

    private string cnpj;

    #endregion

    #region Metodos de Encapsulamento

    public string Cnpj
    {
        set { cnpj = value; }
        get { return cnpj; }
    }

    #endregion
}
}
```