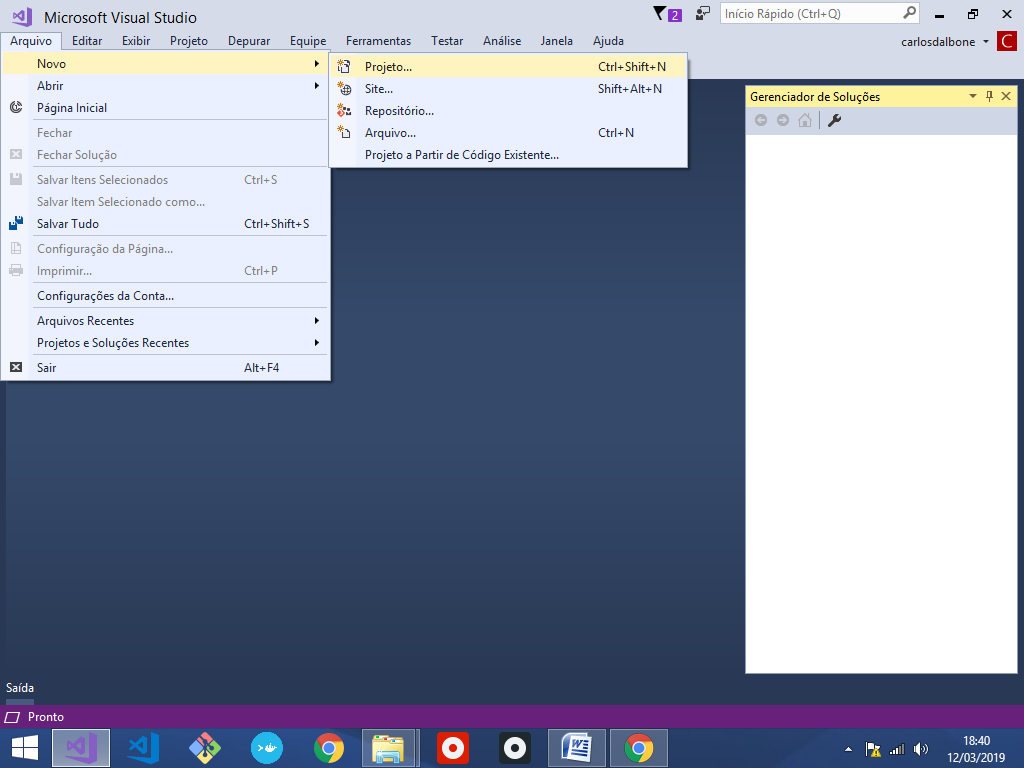
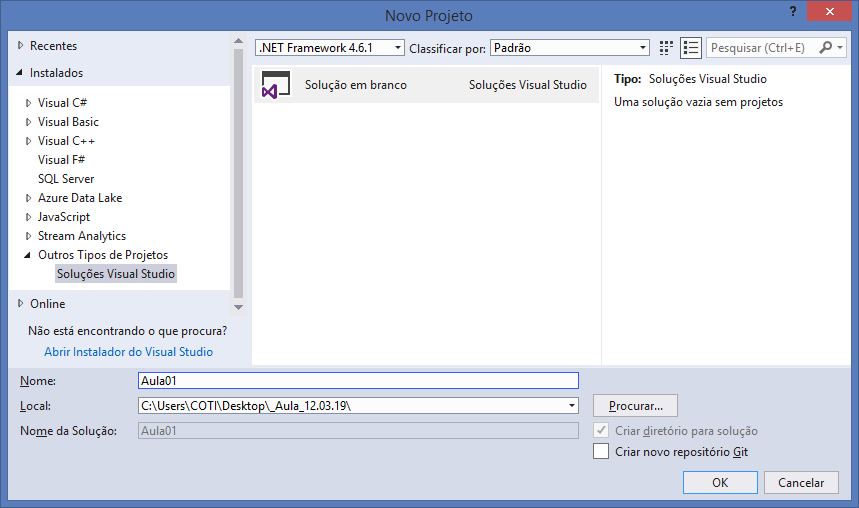
Visual Studio 2017 Community

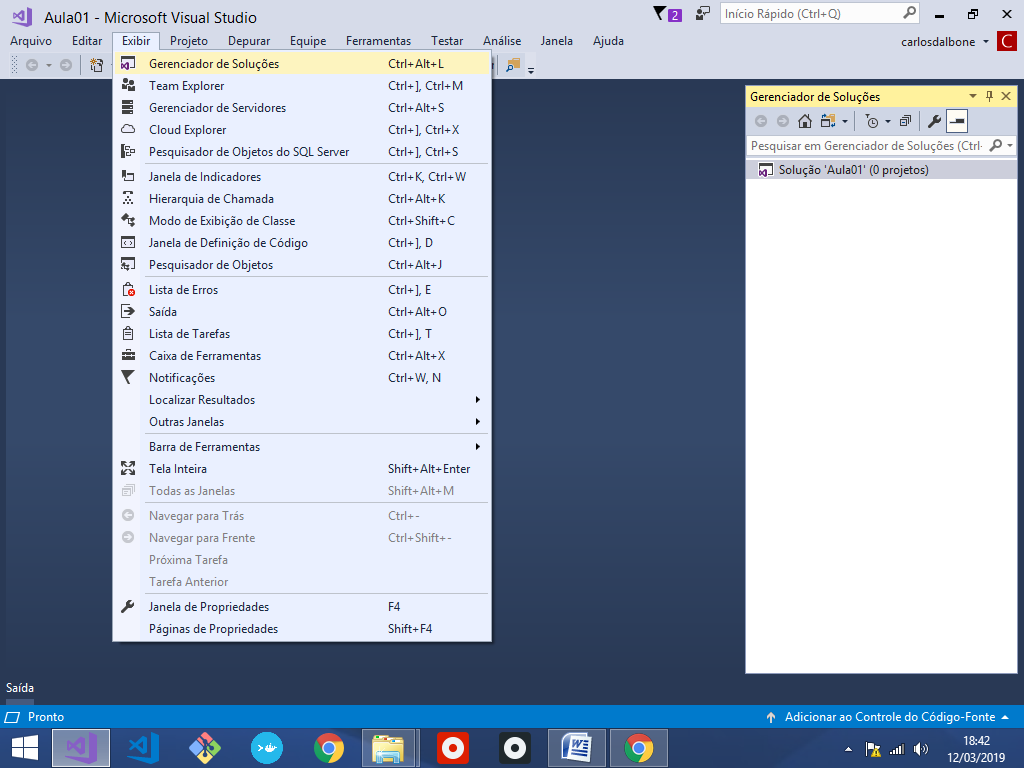
IDE: Ambiente integrado de desenvolvimento

Download: https://visualstudio.microsoft.com/pt-br/downloads/

**Passo 1) Criando uma solution (pasta de trabalho em branco)**

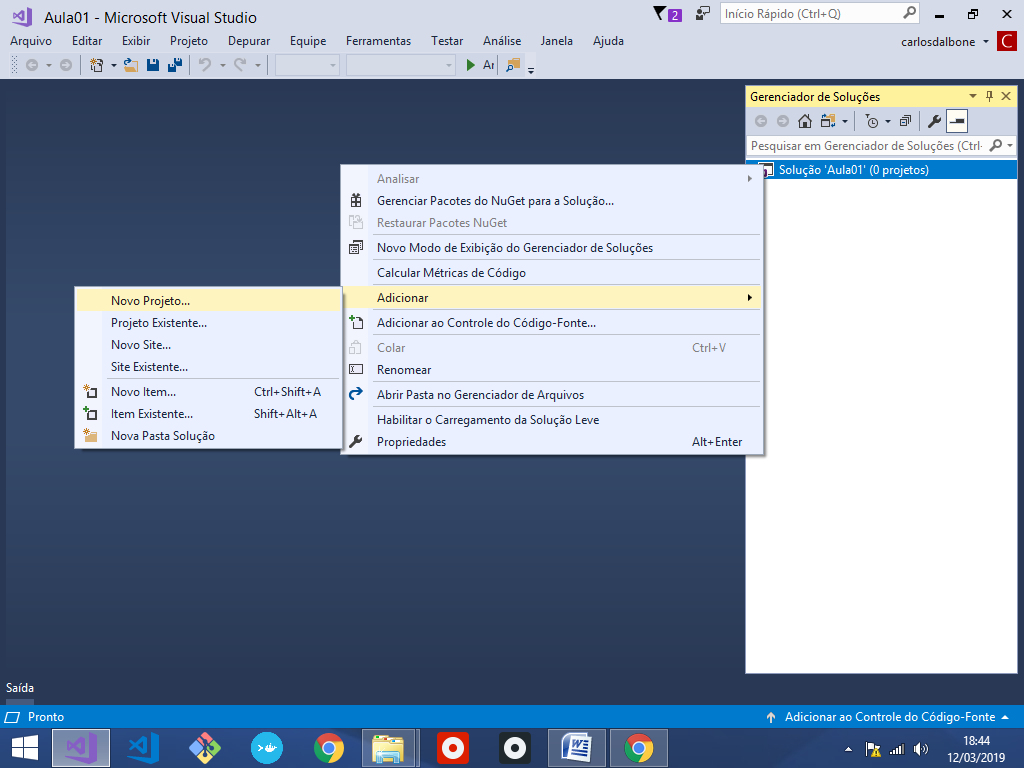


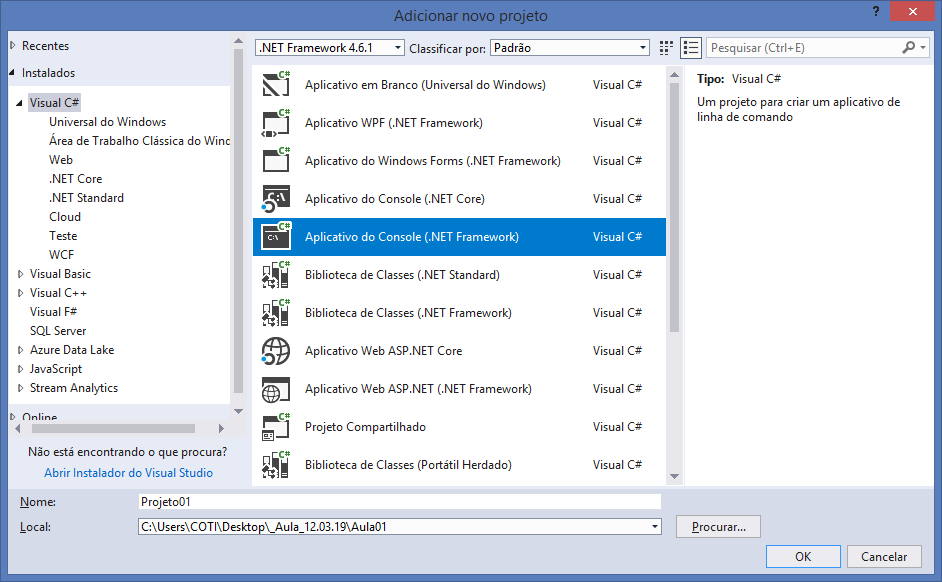




Console Application

Projetos criados para execução em prompt de comando (DOS)





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto01

{

class Program

{

//função que executa o projeto

static void Main(string[] args)

{

//impressão no prompt de comando

Console.WriteLine("Seja bem vindo ao .NET");

//pausar a execução do prompt até que

//qualquer tecla seja pressionada

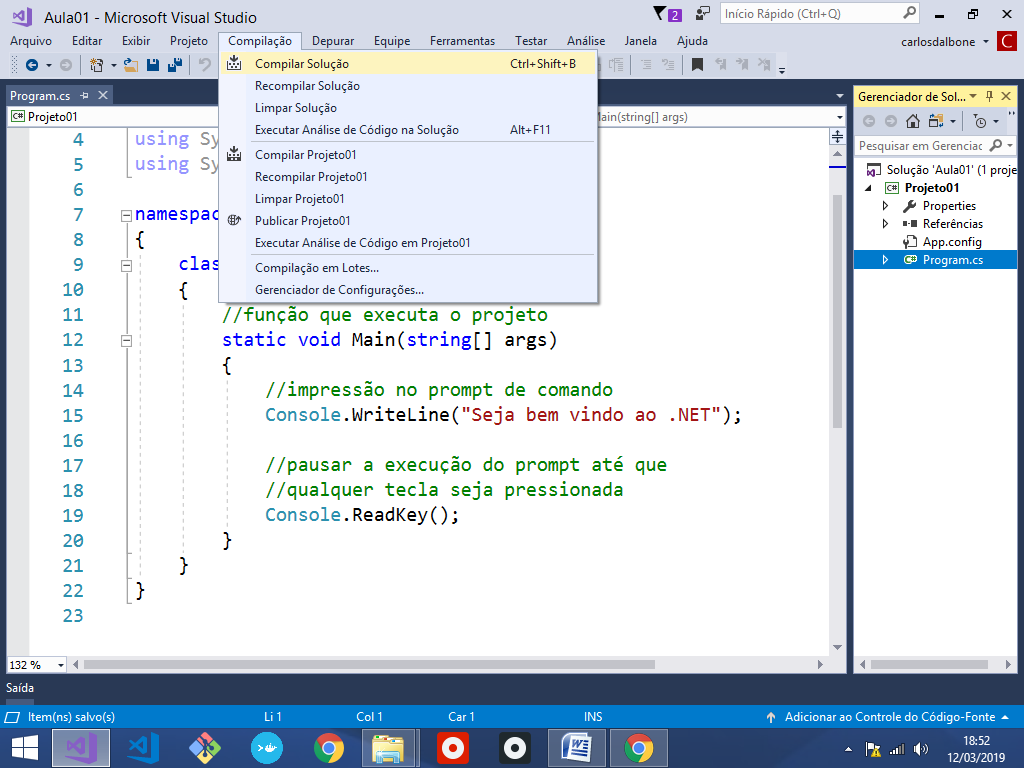
Console.ReadKey();

}

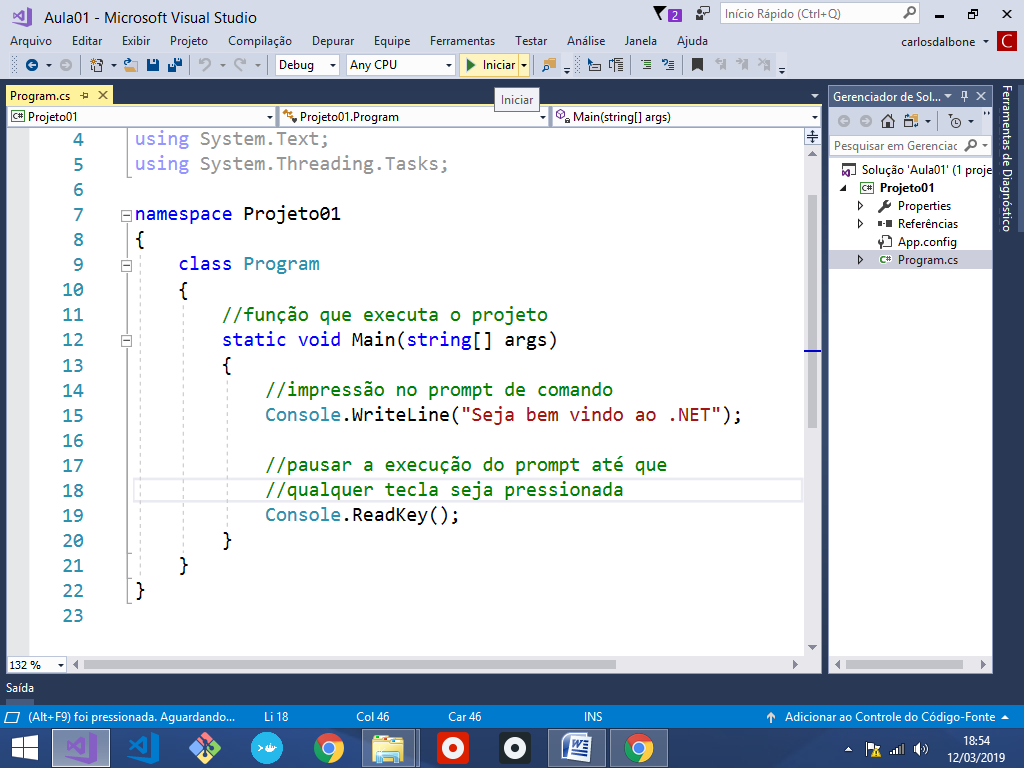
}

}

**Compilando o código-fonte:**



**Executando:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto01

{

class Program

{

//função que executa o projeto

static void Main(string[] args)

{

//impressão no prompt de comando

Console.WriteLine("Seja bem vindo ao .NET");

//cw -> 2x[tab]

Console.Write("Informe seu nome: ");

string nome = Console.ReadLine();

//imprimindo

Console.WriteLine("\nOlá: " + nome);

//pausar a execução do prompt até que

//qualquer tecla seja pressionada

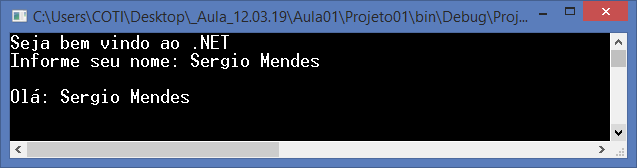
Console.ReadKey();

}

}

}

**Executando:**



Exercício:

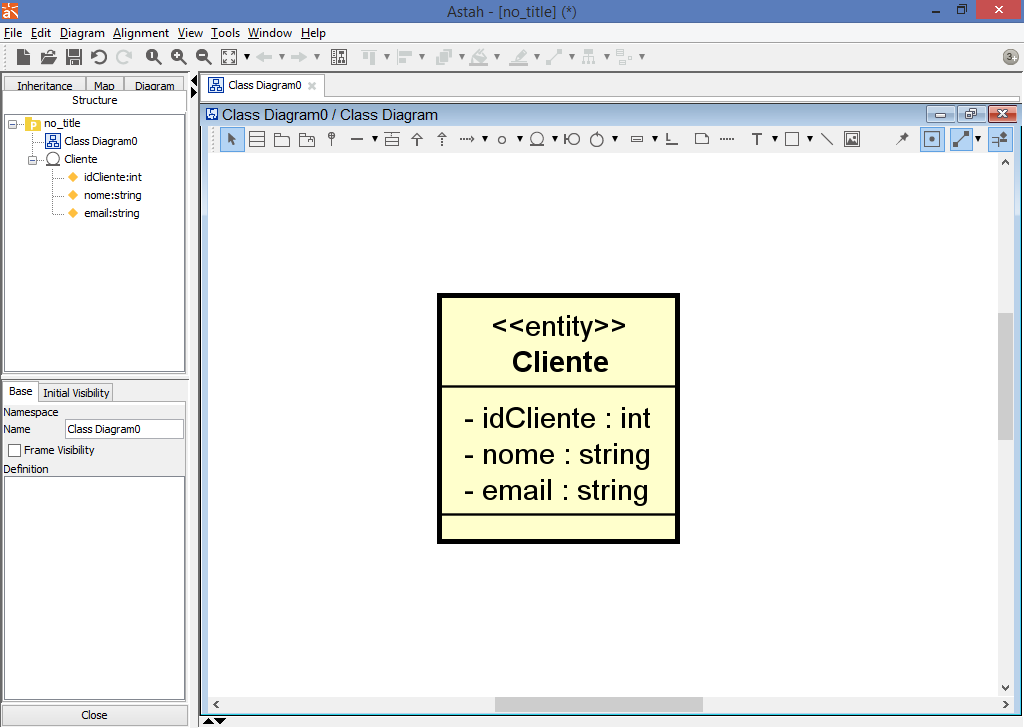
Criar um programa em C# que leia os dados de um cliente composto de Id, Nome e Email e em seguida grave os dados deste cliente em um arquivo de extensão CSV.

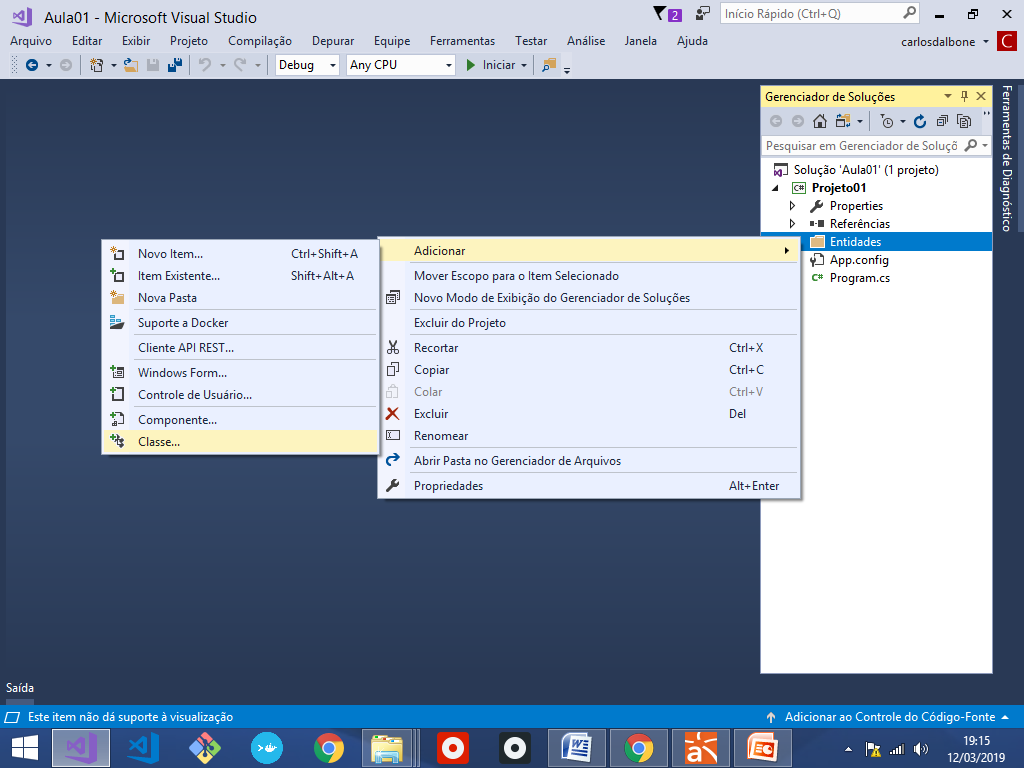
**Primeiro Passo:**

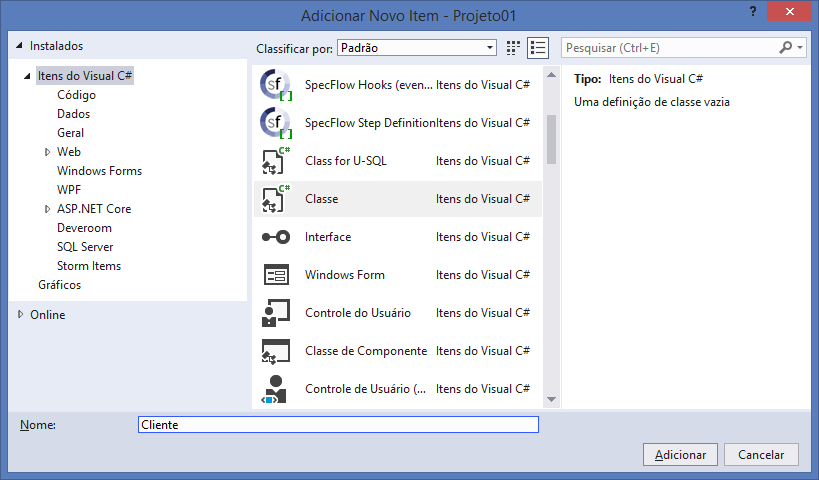
Criando uma classe para representar o "Cliente"

Classes

* As *classes* são os tipos do C# mais fundamentais. Uma classe é uma estrutura de dados que combina ações (métodos e outros membros da função) e estado (campos) em uma única unidade.
* Uma classe fornece uma definição para *instâncias* da classe criadas dinamicamente, também conhecidas como *objetos*.
* As classes dão suporte à *herança* e *polimorfismo*, mecanismos nos quais *classes derivadas* podem estender e especializar *classes base*.







using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//namespace -> define o local da classe

//dentro do projeto (caminho virtual)

namespace Projeto01.Entidades

{

//definição da classe

//public -> acesso total

public class Cliente

{

}

}

**Declarando os atributos:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//namespace -> define o local da classe

//dentro do projeto (caminho virtual)

namespace Projeto01.Entidades

{

//definição da classe

//public -> acesso total

public class Cliente

{

//atributos (dados)

public int idCliente;

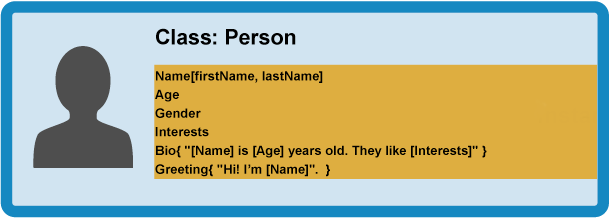
public string nome;

public string email;

}

}

Objeto



Consiste de uma variável criada a partir do espaço de memória de uma classe. Também é chamado de instancia da classe, por exemplo:

**Cliente cliente = new Cliente();**

[Classe] [Objeto] [Criando instância - **Construtor**]

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entidades; //importando

namespace Projeto01

{

class Program

{

//função que executa o projeto

static void Main(string[] args)

{

//impressão no prompt de comando

Console.WriteLine("\n - CADASTRO DE CLIENTE - \n");

Cliente cliente = new Cliente();

Console.Write("Informe o Id do Cliente......: ");

cliente.idCliente = int.Parse(Console.ReadLine());

Console.Write("Informe o Nome do Cliente....: ");

cliente.nome = Console.ReadLine();

Console.Write("Informe o Email do Cliente...: ");

cliente.email = Console.ReadLine();

Console.WriteLine("\nDados do Cliente:");

Console.WriteLine("\tId do Cliente.: " + cliente.idCliente);

Console.WriteLine("\tNome..........: " + cliente.nome);

Console.WriteLine("\tEmail.........: " + cliente.email);

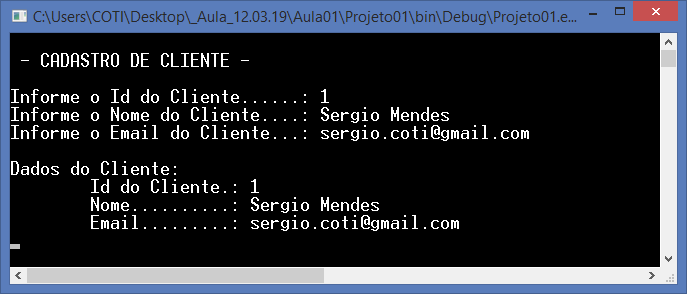
Console.ReadKey();

}

}

}

Executando:



**public**

* Define acesso total para uma classe, atributo ou método.

**internal**

* Permite acesso somente dentro do mesmo namespace ou arquivo.

**protected**

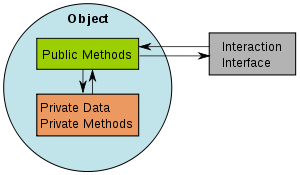
* Permite (para atributos ou métodos) acesso somente por meio de herança.

**private**

* Permite (para atributos ou métodos) acesso somente dentro da própria classe onde o elemento foi declarado.

Encapsulamento

* Ao invés de declararmos os atributos como públicos, iremos mantê-los com visibilidade "private" e criar métodos que permitam acessar os atributos.
* Um exemplo de encapsulamento ocorre quando uma classe declara seus atributos como privados e cria métodos públicos que permitem acessar indiretamente os atributos.
* Estes métodos são   
  chamados de **set** e **get**
* **set** (entrada de dados)
* **get** (saida de dados)



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//namespace -> define o local da classe

//dentro do projeto (caminho virtual)

namespace Projeto01.Entidades

{

//definição da classe

//public -> acesso total

public class Cliente

{

//atributos (dados)

private int idCliente;

private string nome;

private string email;

//métodos set/get

public int IdCliente //propriedade (método set/get)

{

set { idCliente = value; } //entrada

get { return idCliente; } //saída

}

public string Nome

{

set { nome = value; }

get { return nome; }

}

public string Email

{

set { email = value; }

get { return email; }

}

}

}

**Executando:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entidades; //importando

namespace Projeto01

{

class Program

{

//função que executa o projeto

static void Main(string[] args)

{

//impressão no prompt de comando

Console.WriteLine("\n - CADASTRO DE CLIENTE - \n");

Cliente cliente = new Cliente();

Console.Write("Informe o Id do Cliente......: ");

cliente.IdCliente = int.Parse(Console.ReadLine());

Console.Write("Informe o Nome do Cliente....: ");

cliente.Nome = Console.ReadLine();

Console.Write("Informe o Email do Cliente...: ");

cliente.Email = Console.ReadLine();

//imprimindo..

Console.WriteLine("\nDados do Cliente:");

Console.WriteLine("\tId do Cliente.: " + cliente.IdCliente);

Console.WriteLine("\tNome..........: " + cliente.Nome);

Console.WriteLine("\tEmail.........: " + cliente.Email);

//pausar a execução do prompt até que

//qualquer tecla seja pressionada

Console.ReadKey();

}

}

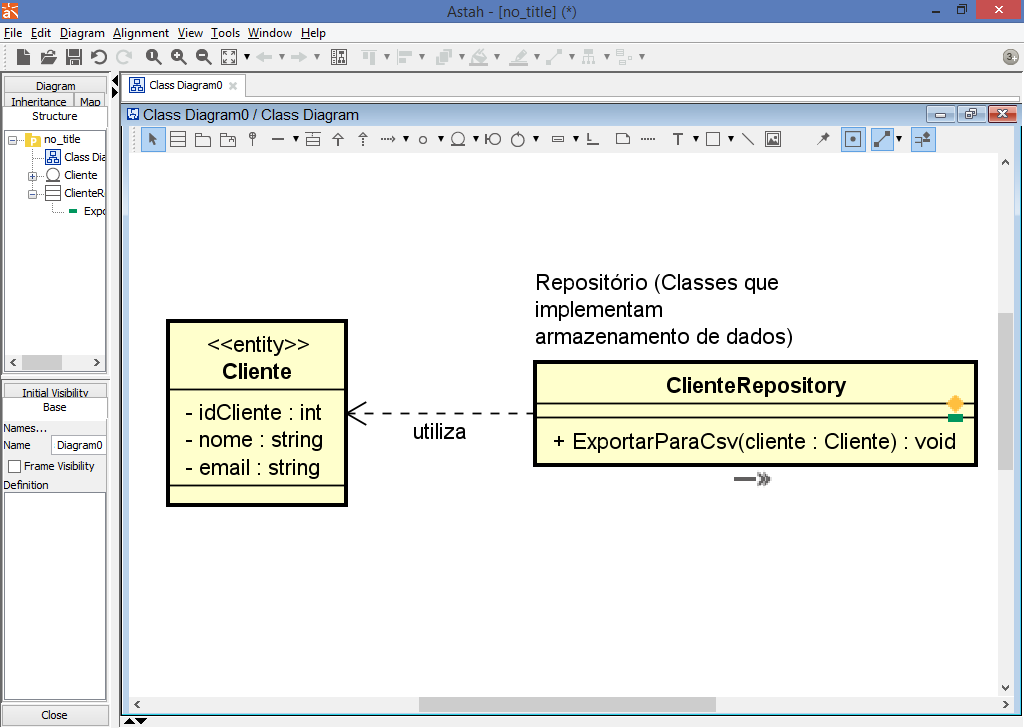
}

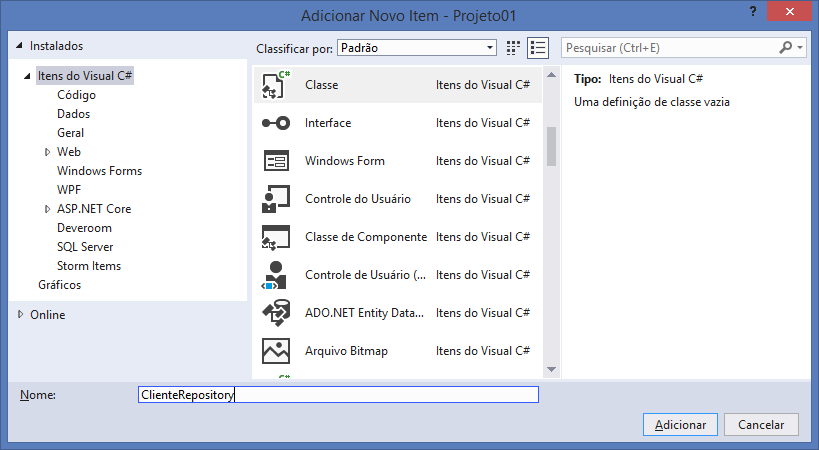
**Saída do programa:**



Criando uma classe para exportar os dados

de cliente para um arquivo CSV:





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entidades;

using System.IO;

namespace Projeto01.Repositorios

{

/// <summary>

/// Classe para armazenamento de dados de Clientes

/// </summary>

public class ClienteRepository

{

/// <summary>

/// Método para exportar os dados de Cliente para arquivo CSV

/// </summary>

/// <param name="cliente">Objeto da classe Cliente</param>

public void ExportarParaCsv(Cliente cliente)

{

using (StreamWriter writer = new StreamWriter

("c:\\temp\\clientes.csv"))

{

writer.WriteLine("{0};{1};{2}",

cliente.IdCliente, cliente.Nome, cliente.Email);

}

}

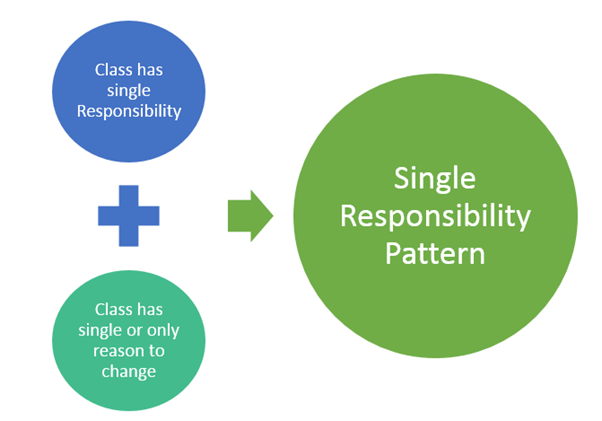
}

}

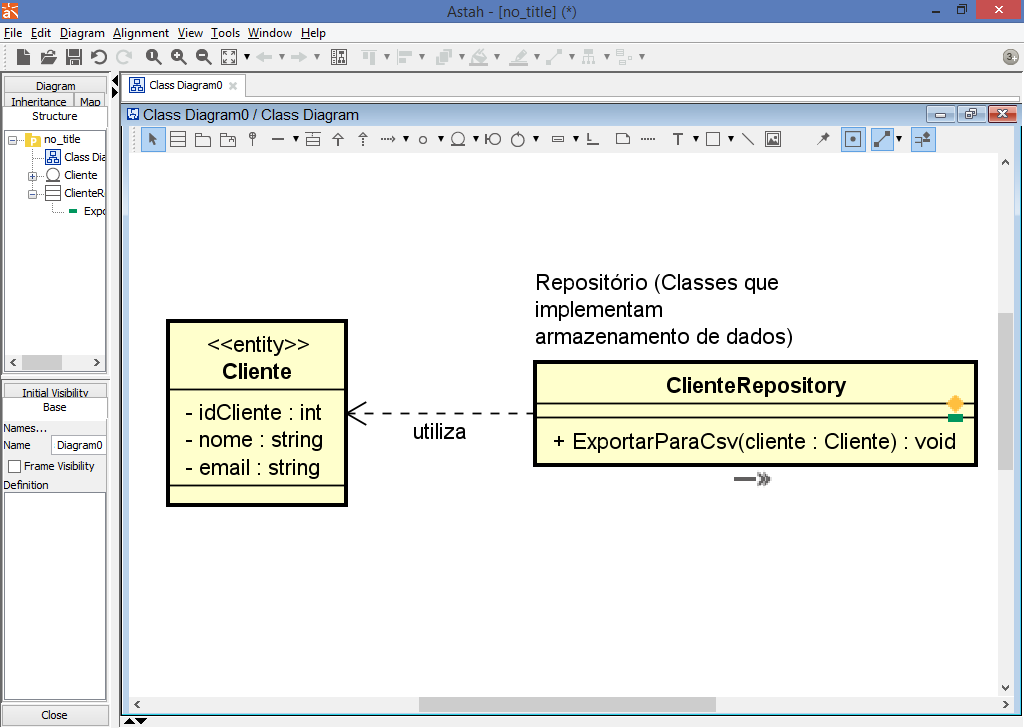
SRP

Princípio de Responsabilidade Única

|  |  |  |
| --- | --- | --- |
| **SRP** | The Single Responsibility Principle Principio da Responsabilidade Única | Uma classe deve ter um, e somente um, motivo para mudar. |







Erros em Tempo de Execução

Chamados de "Exceções"

Para que possamos tratar e capturar estes tipos de exceções

podemos utilizar um bloco denominado **try / catch**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entidades; //importando

using Projeto01.Repositorios; //importando

namespace Projeto01

{

class Program

{

//função que executa o projeto

static void Main(string[] args)

{

//impressão no prompt de comando

Console.WriteLine("\n - CADASTRO DE CLIENTE - \n");

Cliente cliente = new Cliente();

Console.Write("Informe o Id do Cliente......: ");

cliente.IdCliente = int.Parse(Console.ReadLine());

Console.Write("Informe o Nome do Cliente....: ");

cliente.Nome = Console.ReadLine();

Console.Write("Informe o Email do Cliente...: ");

cliente.Email = Console.ReadLine();

//imprimindo..

Console.WriteLine("\nDados do Cliente:");

Console.WriteLine("\tId do Cliente.: " + cliente.IdCliente);

Console.WriteLine("\tNome..........: " + cliente.Nome);

Console.WriteLine("\tEmail.........: " + cliente.Email);

//instanciando a classe de repositorio

ClienteRepository clienteRepository = new ClienteRepository();

**try //tentativa**

**{**

**clienteRepository.ExportarParaCsv(cliente);**

**Console.WriteLine("\nDados gravados em CSV com sucesso!");**

**}**

**catch(Exception e) //captura da exceção**

**{**

**Console.WriteLine("Ocorreu um erro: " + e.Message);**

**}**

//pausar a execução do prompt até que

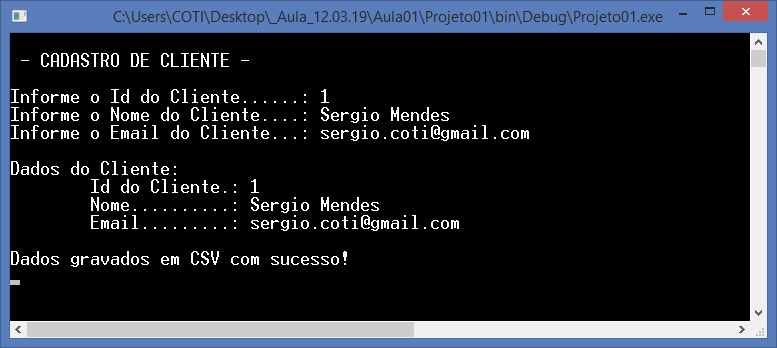
//qualquer tecla seja pressionada

Console.ReadKey();

}

}

}



**Incluindo uma flag para que o StreamWriter não**

**sobrescreva o conteúdo do arquivo.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entidades;

using System.IO;

namespace Projeto01.Repositorios

{

/// <summary>

/// Classe para armazenamento de dados de Clientes

/// </summary>

public class ClienteRepository

{

/// <summary>

/// Método para exportar os dados de Cliente para arquivo CSV

/// </summary>

/// <param name="cliente">Objeto da classe Cliente</param>

public void ExportarParaCsv(Cliente cliente)

{

using (StreamWriter writer

= new StreamWriter("c:\\temp\\clientes.csv", **true**))

{

writer.WriteLine("{0};{1};{2}",

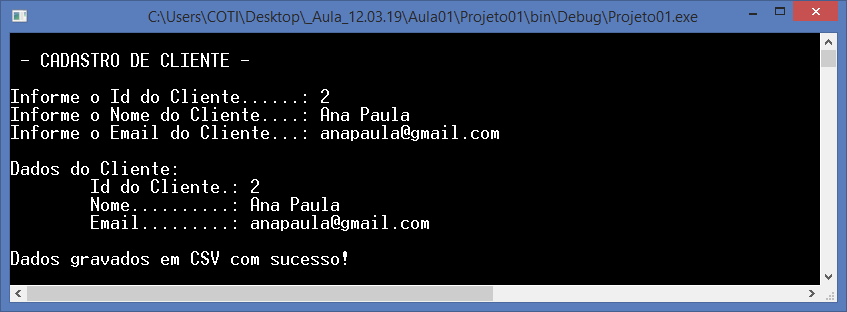
cliente.IdCliente, cliente.Nome, cliente.Email);

}

}

}

}



**Recursividade:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entidades; //importando

using Projeto01.Repositorios; //importando

namespace Projeto01

{

class Program

{

//função que executa o projeto

static void Main(string[] args)

{

//impressão no prompt de comando

Console.WriteLine("\n - CADASTRO DE CLIENTE - \n");

Cliente cliente = new Cliente();

Console.Write("Informe o Id do Cliente......: ");

cliente.IdCliente = int.Parse(Console.ReadLine());

Console.Write("Informe o Nome do Cliente....: ");

cliente.Nome = Console.ReadLine();

Console.Write("Informe o Email do Cliente...: ");

cliente.Email = Console.ReadLine();

//imprimindo..

Console.WriteLine("\nDados do Cliente:");

Console.WriteLine("\tId do Cliente.: " + cliente.IdCliente);

Console.WriteLine("\tNome..........: " + cliente.Nome);

Console.WriteLine("\tEmail.........: " + cliente.Email);

//instanciando a classe de repositorio

ClienteRepository clienteRepository = new ClienteRepository();

try //tentativa

{

clienteRepository.ExportarParaCsv(cliente);

Console.WriteLine("\nDados gravados em CSV com sucesso!");

}

catch(Exception e) //captura da exceção

{

Console.WriteLine("Ocorreu um erro: " + e.Message);

}

Console.Write("\nDeseja continuar? (S)im ou (N)ão: ");

string opcao = Console.ReadLine();

if (opcao.Equals("S", StringComparison.OrdinalIgnoreCase))

{

Console.Clear();

//recurvidade..

Main(args);

}

else

{

Console.WriteLine("Bye!");

}

//pausar a execução do prompt até que

//qualquer tecla seja pressionada

//Console.ReadKey();

}

}

}

