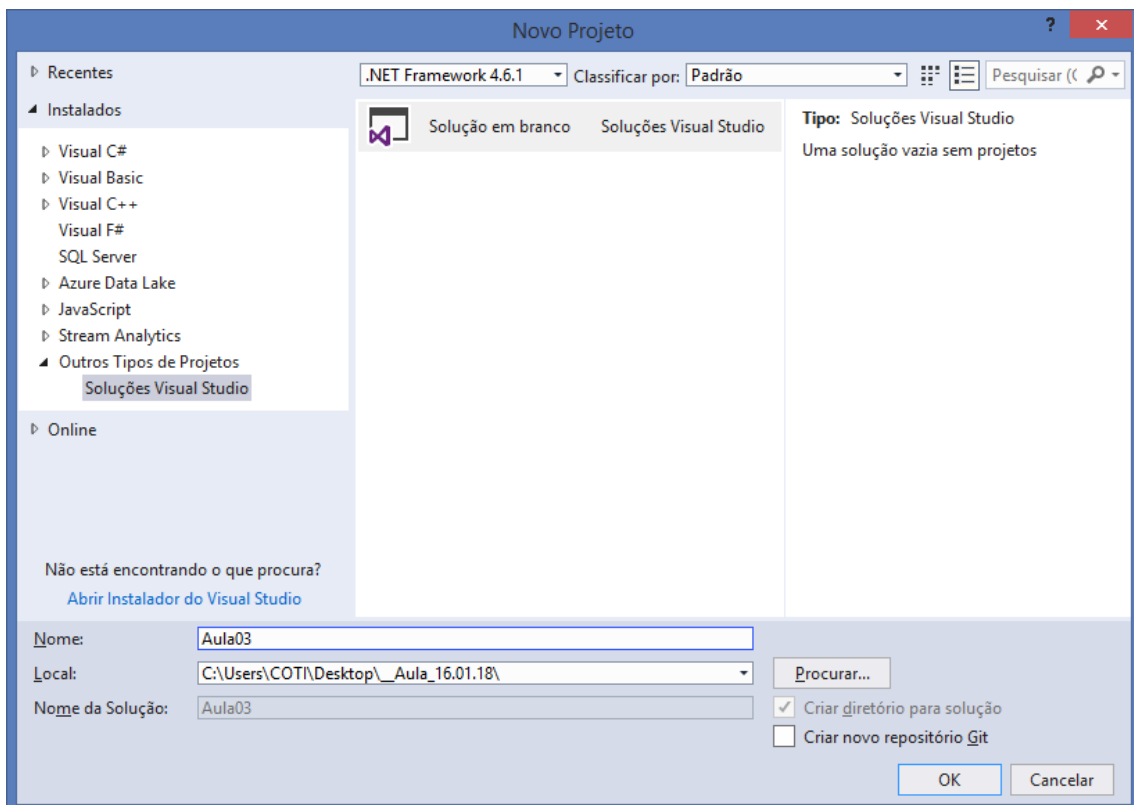
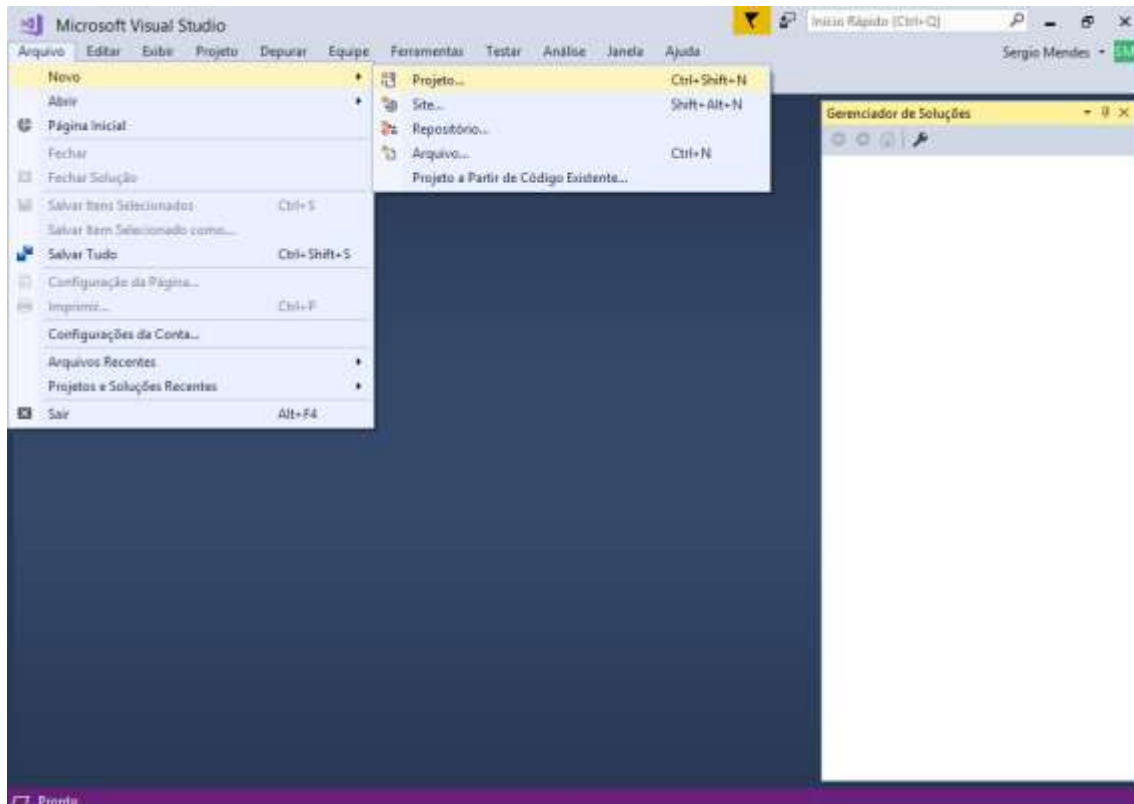
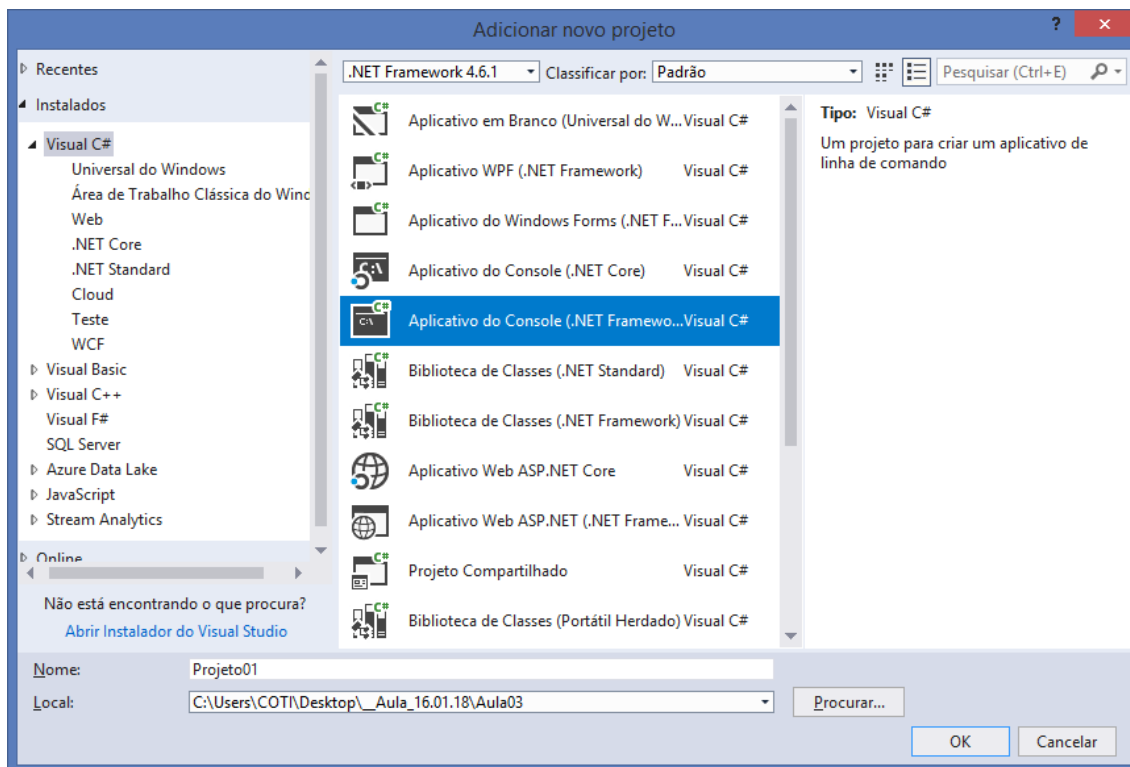


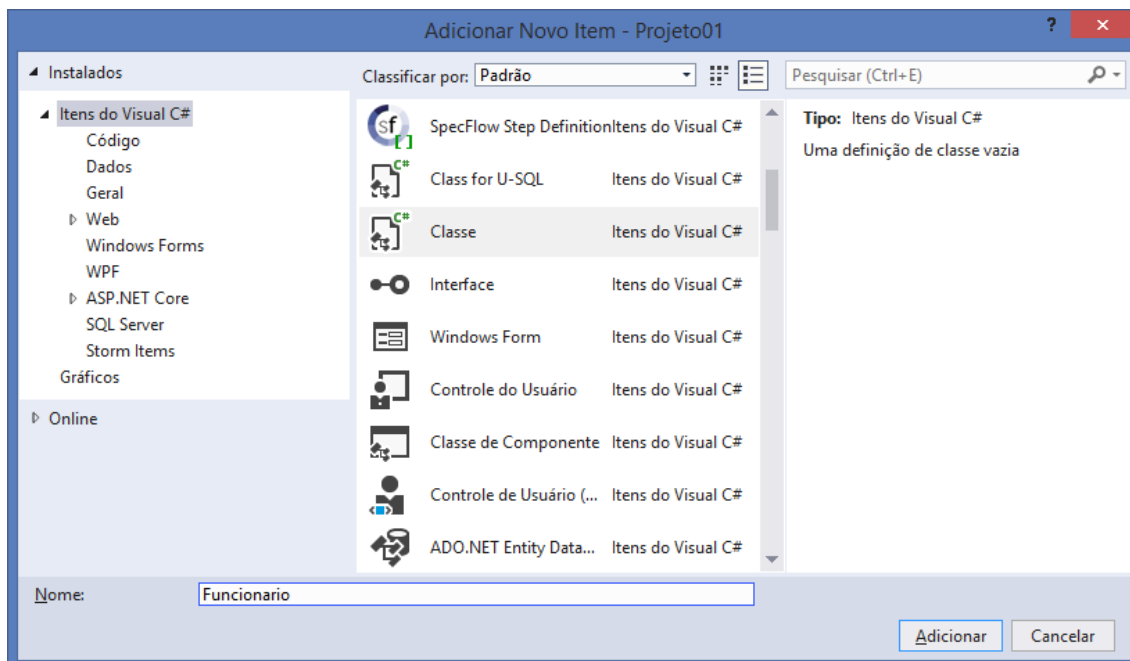
Criando uma nova solution em branco:



Criando um projeto **Console Application**



Classe de entidade: **Funcionario**



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

```
namespace Projeto01.Entidades
{
    public class Funcionario
    {
        //encapsulamento implicito..
        //[prop] + 2x[tab]
        public int IdFuncionario { get; set; }
        public string Nome { get; set; }
        public decimal Salario { get; set; }
        public DateTime DataAdmissao { get; set; }

        //construtor default [ctor] + 2x[tab]
        public Funcionario()
        {
            //vazio..
        }

        //sobrecarga de construtor (overloading)
        public Funcionario(int idFuncionario, string nome,
                           decimal salario, DateTime dataAdmissao)
        {
            IdFuncionario = idFuncionario;
            Nome = nome;
            Salario = salario;
            DataAdmissao = dataAdmissao;
        }

        //sobrescrita de método (override)
        public override string ToString()
        {
            return string.Format("{0}, {1}, {2}, {3}",
                                  IdFuncionario, Nome, Salario, DataAdmissao);
        }
    }
}
```

Utilizando Arrays (**Vetores**)

Desenvolvendo arranjos de objetos

Funcionario[] vetor = new Funcionario[4];

[Tipo - Vetor de Funcionario]

[Inicializando o vetor [tamanho]]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto01.Entidades; //importando..
```

```
namespace Projeto01
{
    class Program
    {
```

```
static void Main(string[] args)
{
    //declarando um array de funcionarios..
    Funcionario[] vetor = new Funcionario[4];

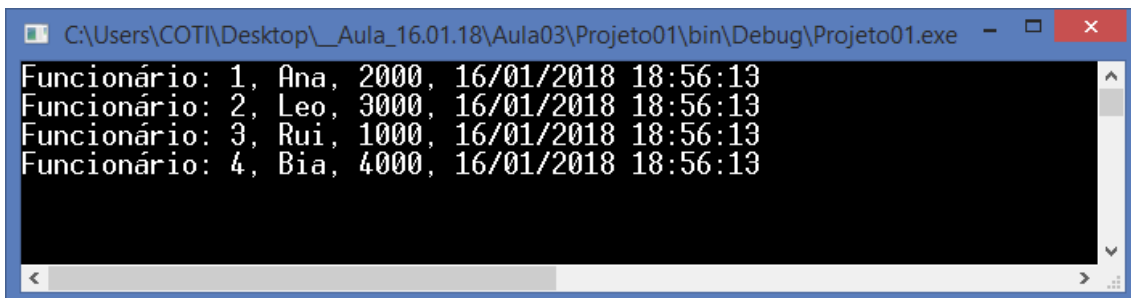
    //criando objetos da classe Funcionario..
    Funcionario f1 = new Funcionario(1, "Ana", 2000, DateTime.Now);
    Funcionario f2 = new Funcionario(2, "Leo", 3000, DateTime.Now);
    Funcionario f3 = new Funcionario(3, "Rui", 1000, DateTime.Now);
    Funcionario f4 = new Funcionario(4, "Bia", 4000, DateTime.Now);

    //adicionando os funcionarios no vetor..
    vetor[0] = f1;
    vetor[1] = f2;
    vetor[2] = f3;
    vetor[3] = f4;

    //percorrer o vetor..
    for (int i = 0; i < vetor.Length; i++)
    {
        //receber o funcionario contido na posição 'i' do vetor
        Funcionario f = vetor[i];
        //imprimindo..
        Console.WriteLine("Funcionário: " + f.ToString());
    }

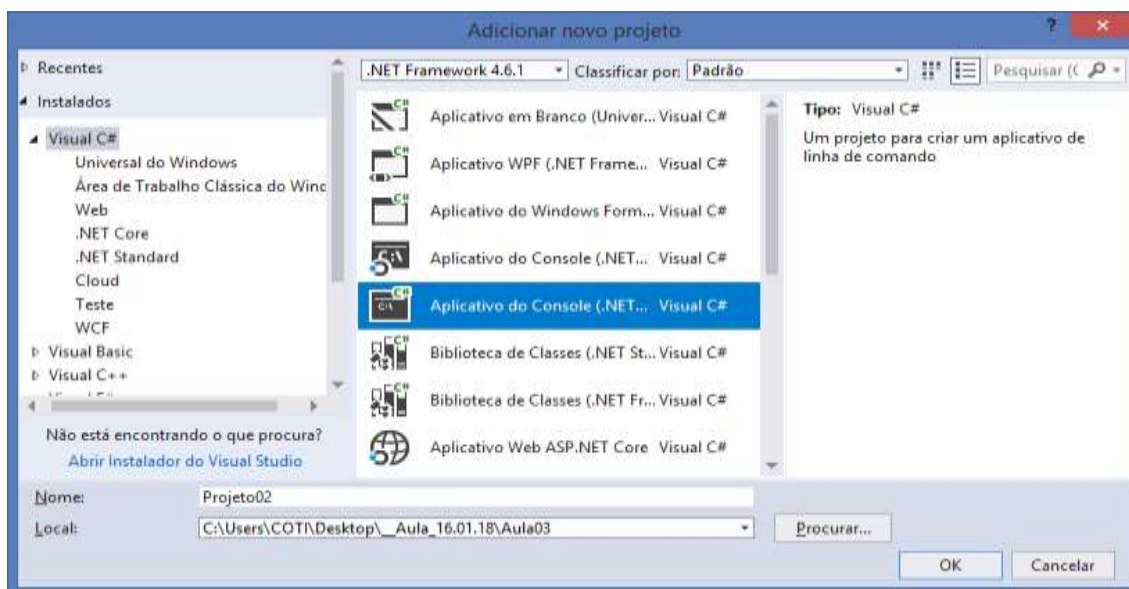
    Console.ReadKey(); //pausar a execução do prompt..
}
}
```

Saída do programa:



Listas (**System.Collections.Generic.List**)

Recurso para manipulação de arrays de objetos que permite a execução de métodos de busca, varredura, ordenação, agrupamento, etc..



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

namespace Projeto02.Entidades
{
    public class Pessoa
    {
        //propriedades..
        public int IdPessoa { get; set; }
        public string Nome { get; set; }

        //construtor default..
        public Pessoa()
        {

        }

        //sobrecarga de construtores (overloading)
        public Pessoa(int idPessoa, string nome)
        {
            IdPessoa = idPessoa;
            Nome = nome;
        }

        //sobrescrita do método ToString()..
        public override string ToString()
        {
            return $"Id Pessoa: {IdPessoa}, Nome: {Nome}";
        }
    }
}
```

List<Pessoa> lista = new List<Pessoa>();

[Lista do tipo 'Pessoa'] [Nome] [Espaço de memória]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto02.Entidades; //importando..

namespace Projeto02
{
    class Program
    {
        static void Main(string[] args)
        {
            //declarando objetos da classe Pessoa..
            Pessoa p1 = new Pessoa(1, "João Pedro");
            Pessoa p2 = new Pessoa(2, "Ana Paula");
            Pessoa p3 = new Pessoa(3, "José da Silva");
            Pessoa p4 = new Pessoa(4, "Ana Maria");

            //declarando uma lista de pessoa..
            List<Pessoa> lista = new List<Pessoa>();

            //adicionando pessoas na lista..
            lista.Add(p1);
            lista.Add(p2);
            lista.Add(p3);
            lista.Add(p4);

            Console.ReadKey();
        }
    }
}
```

foreach

Instrução simples para varredura de coleções de objetos.

Exemplo:

foreach(Pessoa p in lista)

[Laço] [Tipo] [Objeto] [Collection]

*** Para cada elemento contido na lista,
armazene um objeto Pessoa 'p'*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto02.Entidades; //importando..

namespace Projeto02
{

```

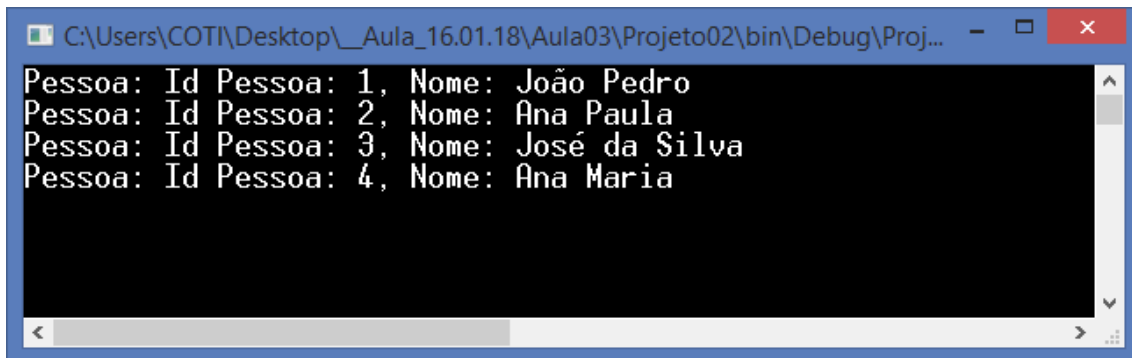
```
class Program
{
    static void Main(string[] args)
    {
        //declarando objetos da classe Pessoa..
        Pessoa p1 = new Pessoa(1, "João Pedro");
        Pessoa p2 = new Pessoa(2, "Ana Paula");
        Pessoa p3 = new Pessoa(3, "José da Silva");
        Pessoa p4 = new Pessoa(4, "Ana Maria");

        //declarando uma lista de pessoa..
        List<Pessoa> lista = new List<Pessoa>();

        //adicionando pessoas na lista..
        lista.Add(p1);
        lista.Add(p2);
        lista.Add(p3);
        lista.Add(p4);

        foreach(Pessoa p in lista)
        {
            Console.WriteLine("Pessoa: " + p.ToString());
        }

        Console.ReadKey();
    }
}
```



LAMBDA

O LAMBDA é um recurso da linguagem C# utilizado para filtrar, ordenar, agrupar, classificar etc... collections e outros tipos de dados. O uso do LAMBDA vem de tornando cada vez mais recorrente no .NET ao ponto de que muitas sintaxes do C# vem sendo simplifiocadas e substituidas pelo uso do LAMBDA.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto02.Entidades; //importando..
```

```
namespace Projeto02
{
    class Program
    {
        static void Main(string[] args)
        {
            //declarando objetos da classe Pessoa..
            Pessoa p1 = new Pessoa(1, "João Pedro");
            Pessoa p2 = new Pessoa(2, "Ana Paula");
            Pessoa p3 = new Pessoa(3, "José da Silva");
            Pessoa p4 = new Pessoa(4, "Ana Maria");

            //declarando uma lista de pessoa..
            List<Pessoa> lista = new List<Pessoa>();

            //adicionando pessoas na lista..
            lista.Add(p1);
            lista.Add(p2);
            lista.Add(p3);
            lista.Add(p4);

            Console.WriteLine("\n - LISTAGEM DE PESSOAS - \n");

            foreach(Pessoa p in lista)
            {
                Console.WriteLine("Pessoa: " + p.ToString());
            }

            Console.WriteLine("\n - FILTRAR PESSOAS PELO NOME: - \n");
            Console.Write("Informe o nome: ");
            string nome = Console.ReadLine();

            //Filtrando o conteudo da lista utilizando LAMBDA..
            List<Pessoa> resultado = lista
                .Where(p => p.Nome.StartsWith(nome))
                .OrderBy(p => p.Nome)
                .ToList();

            //varrer e imprimir os resultados..
            foreach(Pessoa p in resultado)
            {
                Console.WriteLine("Pessoa: " + p.ToString());
            }

            Console.ReadKey();
        }
    }
}
```


Executando:

```

C:\Users\COTI\Desktop\_Aula_16.01.18\Aula03\Projeto02\bin\Debug\Projeto02.exe

- LISTAGEM DE PESSOAS -

Pessoa: Id Pessoa: 1, Nome: João Pedro
Pessoa: Id Pessoa: 2, Nome: Ana Paula
Pessoa: Id Pessoa: 3, Nome: José da Silva
Pessoa: Id Pessoa: 4, Nome: Ana Maria

- FILTRAR PESSOAS PELO NOME: -

Informe o nome: Ana
Pessoa: Id Pessoa: 4, Nome: Ana Maria
Pessoa: Id Pessoa: 2, Nome: Ana Paula
  
```

Novo projeto:

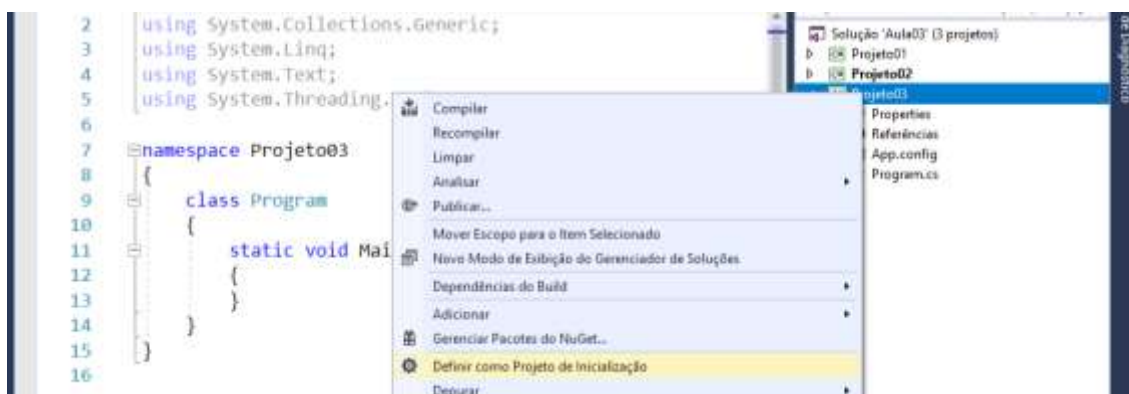
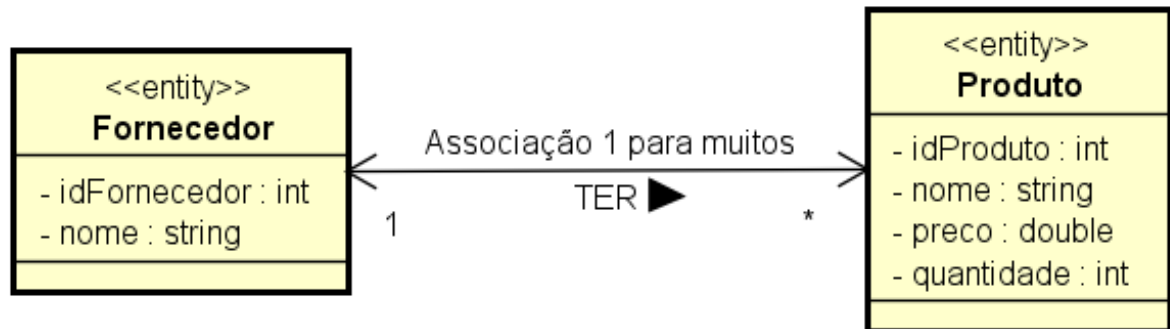


Diagrama de Classes

Relacionamento de Associação 1 para muitos



```

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

namespace Projeto03.Entidades
{
    public class Fornecedor
    {
        public int IdFornecedor { get; set; }
        public string Nome { get; set; }

        public Fornecedor()
        {
        }

        public Fornecedor(int idFornecedor, string nome)
        {
            IdFornecedor = idFornecedor;
            Nome = nome;
        }

        public override string ToString()
        {
            return $"Id Fornecedor: {IdFornecedor}, Nome: {Nome}";
        }
    }
}

```

```

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

namespace Projeto03.Entidades
{
    public class Produto
    {
    }
}

```

```

public int IdProduto { get; set; }
public string Nome { get; set; }
public decimal Preco { get; set; }
public int Quantidade { get; set; }

public Produto()
{
}

public Produto(int idProduto, string nome, decimal preco, int quantidade)
{
    IdProduto = idProduto;
    Nome = nome;
    Preco = preco;
    Quantidade = quantidade;
}

public override string ToString()
{
    return $"Id do Produto: {IdProduto}, Nome: {Nome},
           Preço: {Preco}, Quantidade: {Quantidade}";
}
}

```

Relacionando as entidades:

- Produto TEM 1 Fornecedor

```

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

namespace Projeto03.Entidades
{
    public class Produto
    {
        public int IdProduto { get; set; }
        public string Nome { get; set; }
        public decimal Preco { get; set; }
        public int Quantidade { get; set; }

        //Associação (TER-1)
        public Fornecedor Fornecedor { get; set; }

        public Produto()
        {
        }
    }
}

```

```
public Produto(int idProduto, string nome, decimal preco, int quantidade)
{
    IdProduto = idProduto;
    Nome = nome;
    Preço = preco;
    Quantidade = quantidade;
}

public override string ToString()
{
    return $"Id do Produto: {IdProduto}, Nome: {Nome},
           Preço: {Preço}, Quantidade: {Quantidade}";
}
}

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

namespace Projeto03.Entidades
{
    public class Fornecedor
    {
        public int IdFornecedor { get; set; }
        public string Nome { get; set; }

        //Relacionamento TER-Muitos
        public List<Produto> Produtos { get; set; }

        public Fornecedor()
        {
        }

        public Fornecedor(int idFornecedor, string nome)
        {
            IdFornecedor = idFornecedor;
            Nome = nome;
        }

        public override string ToString()
        {
            return $"Id Fornecedor: {IdFornecedor}, Nome: {Nome}";
        }
    }
}
```

Testando:

Instanciando um objeto da classe Fornecedor:

Fornecedor f = new Fornecedor();
[Classe - Tipo] [Objeto] [Construtor]

Precisamos também inicializar a lista de produtos contida em Fornecedor:

f.Produtos = new List<Produto>();
[Propriedade Lista] [Inicializando (Espaço de Memória)]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto03.Entidades;

namespace Projeto03
{
    class Program
    {
        static void Main(string[] args)
        {
            Fornecedor f = new Fornecedor(1, "Loja de Informática");
            //instanciando..
            f.Produtos = new List<Produto>(); //instanciando..

            Produto p1 = new Produto(1, "Notebook", 2000, 10);
            Produto p2 = new Produto(2, "Mouse", 30, 20);
            Produto p3 = new Produto(3, "Monitor", 400, 5);

            //adicionando os produtos na lista..
            f.Produtos.Add(p1);
            f.Produtos.Add(p2);
            f.Produtos.Add(p3);

            //imprimindo..
            Console.WriteLine("Fornecedor: " + f.ToString());
            foreach(Produto p in f.Produtos)
            {
                Console.WriteLine("\tProduto: " + p.ToString());
            }

            Console.ReadKey();
        }
    }
}
```

Utilizando funções de totalizador com LAMBDA:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto03.Entidades;

namespace Projeto03
{
    class Program
    {
        static void Main(string[] args)
        {
            Fornecedor f = new Fornecedor(1, "Loja de Informática");
            //instanciando..
            f.Produtos = new List<Produto>(); //instanciando..

            Produto p1 = new Produto(1, "Notebook", 2000, 10);
            Produto p2 = new Produto(2, "Mouse", 30, 20);
            Produto p3 = new Produto(3, "Monitor", 400, 5);

            //adicionando os produtos na lista..
            f.Produtos.Add(p1);
            f.Produtos.Add(p2);
            f.Produtos.Add(p3);

            //imprimindo..
            Console.WriteLine("Fornecedor: " + f.ToString());
            foreach(Produto p in f.Produtos)
            {
                Console.WriteLine("\tProduto: " + p.ToString());
            }

            //quantidade total de produtos..
            int somatorioQuantidade = f.Produtos.Sum(p => p.Quantidade);
            decimal mediaPreco = f.Produtos.Average(p => p.Preco);
            decimal maiorPreco = f.Produtos.Max(p => p.Preco);
            decimal menorPreco = f.Produtos.Min(p => p.Preco);
            int qtdProdutos = f.Produtos.Count(p => p.Preco <= 1000);

            Console.WriteLine("Quantidade Total.....: "
                               + somatorioQuantidade);
            Console.WriteLine("Media de Preços.....: " + mediaPreco);
            Console.WriteLine("Maior Preço.....: " + maiorPreco);
            Console.WriteLine("Menor Preço.....: " + menorPreco);
            Console.WriteLine("Produtos com Preço até 1mil.: " + qtdProdutos);

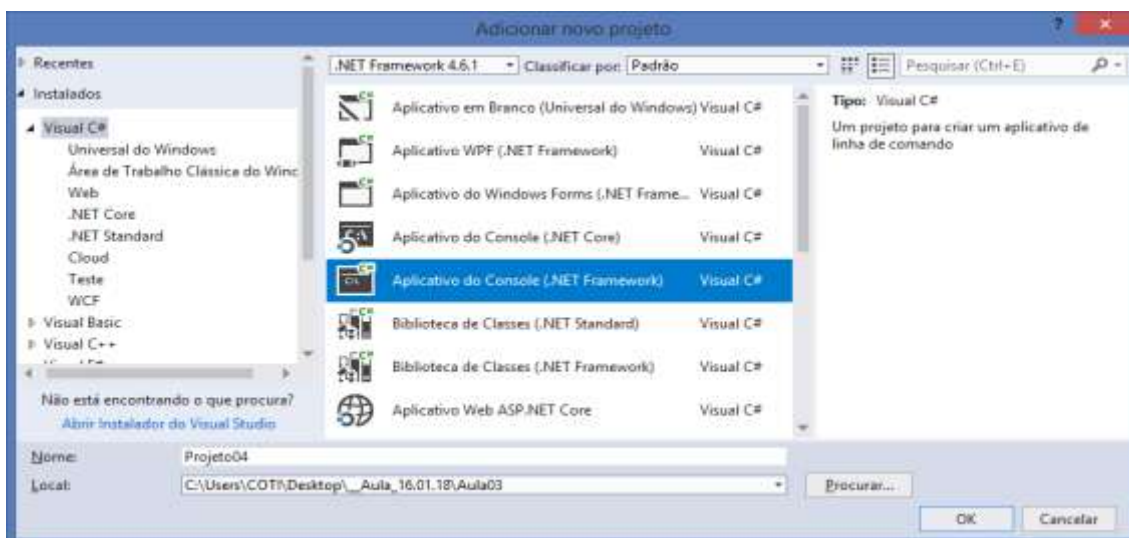
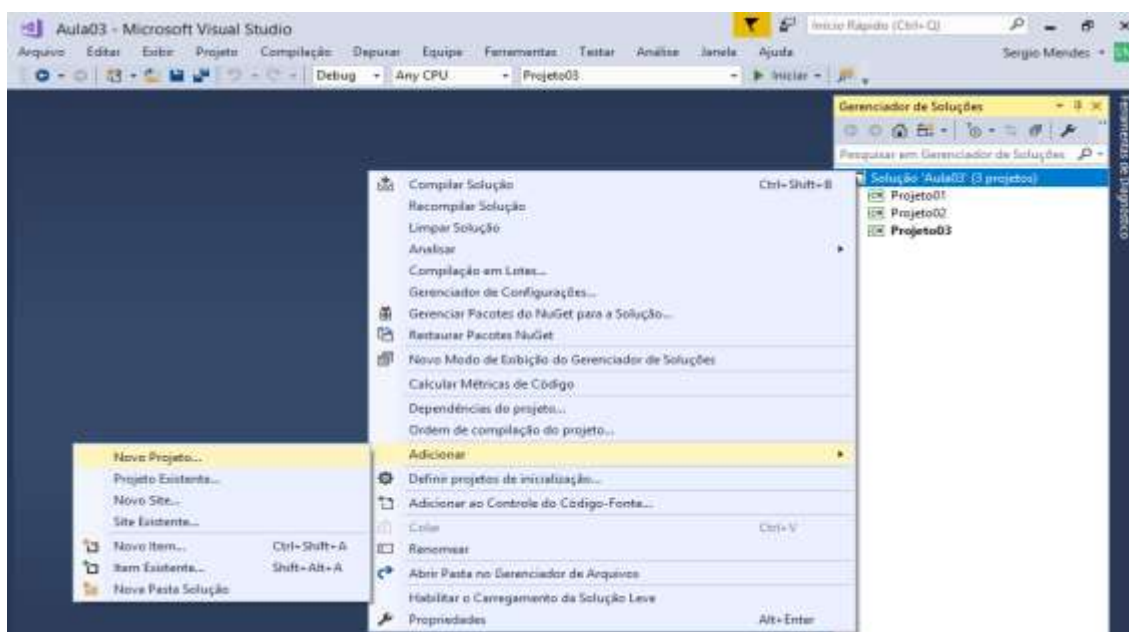
            Console.ReadKey();
        }
    }
}
```

Saida do programa:

```

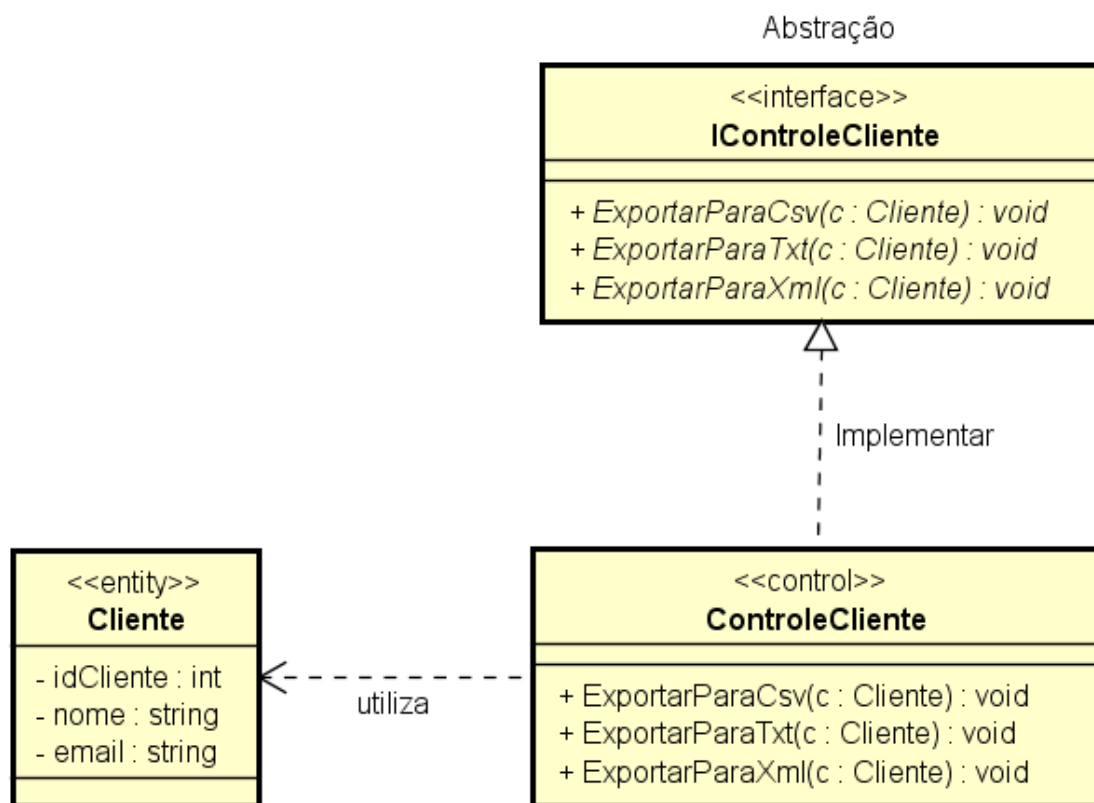
C:\Users\COTI\Desktop\_Aula_16.01.18\Aula03\Projeto03\bin\Debug\Projeto03.exe
Fornecedor: Id Fornecedor: 1, Nome: Loja de Informática
          Produto: Id do Produto: 1, Nome: Notebook, Preço: 2000, Quantidade: 10
          Produto: Id do Produto: 2, Nome: Mouse, Preço: 30, Quantidade: 20
          Produto: Id do Produto: 3, Nome: Monitor, Preço: 400, Quantidade: 5
Quantidade Total.....: 35
Media de Preços.....: 810
Maior Preço.....: 2000
Menor Preço.....: 30
Produtos com Preço até 1mil.: 2
  
```

Novo projeto:

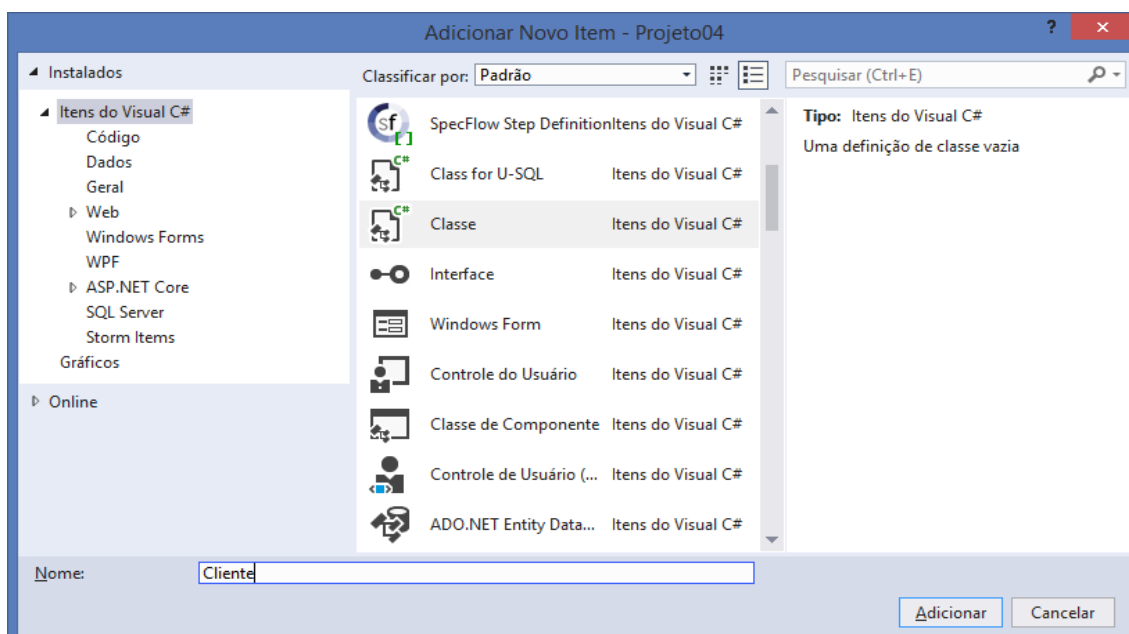


Interfaces

São uma ferramenta de programação Orientada a Objetos utilizada para declarar métodos abstratos, ou seja, métodos que deverão ser implementados por classes que herdarem a interface



Criando a entidade Cliente:




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

namespace Projeto04.Entidades
{
    public class Cliente
    {
        //propriedades..
        public int IdCliente { get; set; }
        public string Nome { get; set; }
        public string Email { get; set; }

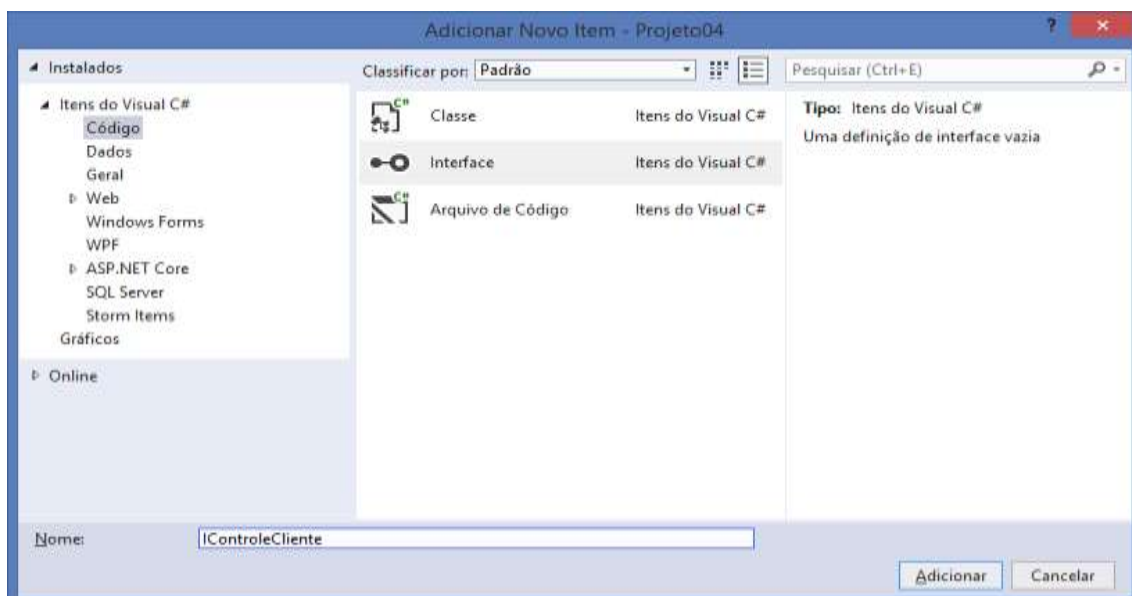
        //construtor padrão..
        public Cliente()
        {
            //ctor + 2x[tab]
        }

        //sobrecarga de construtores (overloading)
        public Cliente(int idCliente, string nome, string email)
        {
            IdCliente = idCliente;
            Nome = nome;
            Email = email;
        }

        public override string ToString()
        {
            return $"{IdCliente}, {Nome}, {Email}";
        }
    }
}
```

Criando uma interface:

**** Regra (Nomes de interfaces começando com a letra I)**



Regras sobre interfaces:

1. Interfaces não podem ter atributos
2. Interfaces não podem ter construtores
3. Interfaces não podem ter métodos com corpo
4. Interfaces somente podem ter métodos abstratos
5. Todos os métodos de uma interface já são implicitamente publicos, não podemos sequer declarar sua visibilidade

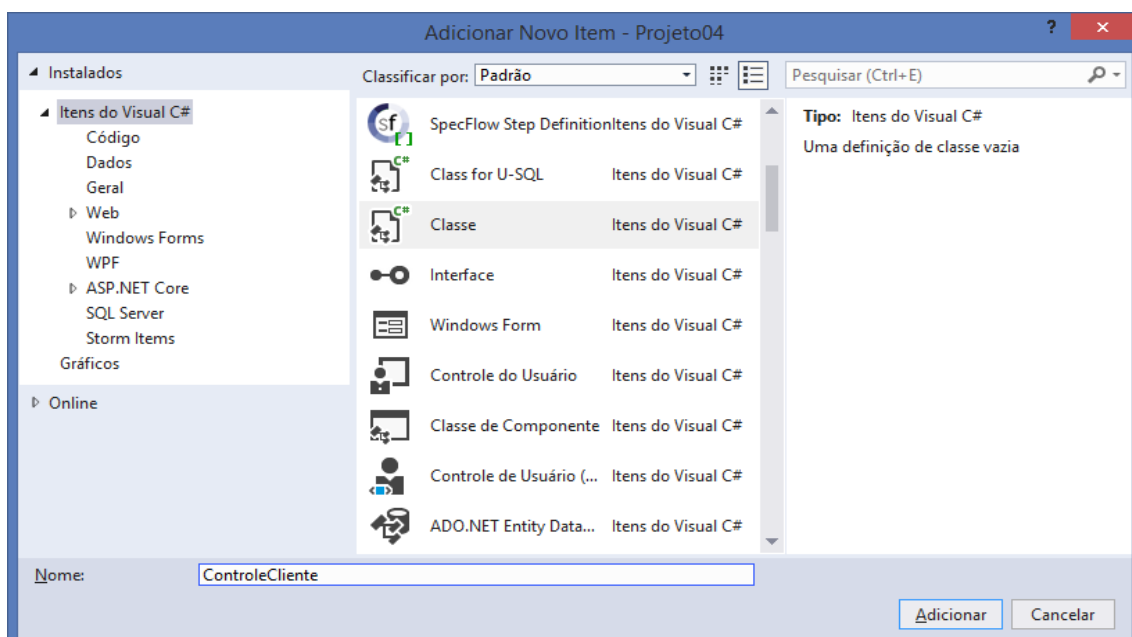
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto04.Entidades;

namespace Projeto04.Contratos
{
    public interface IControleCliente
    {
        //método abstrato..
        void ExportarParaTxt(Cliente c);

        //método abstrato..
        void ExportarParaCsv(Cliente c);

        //método abstrato..
        void ExportarParaXml(Cliente c);
    }
}
```

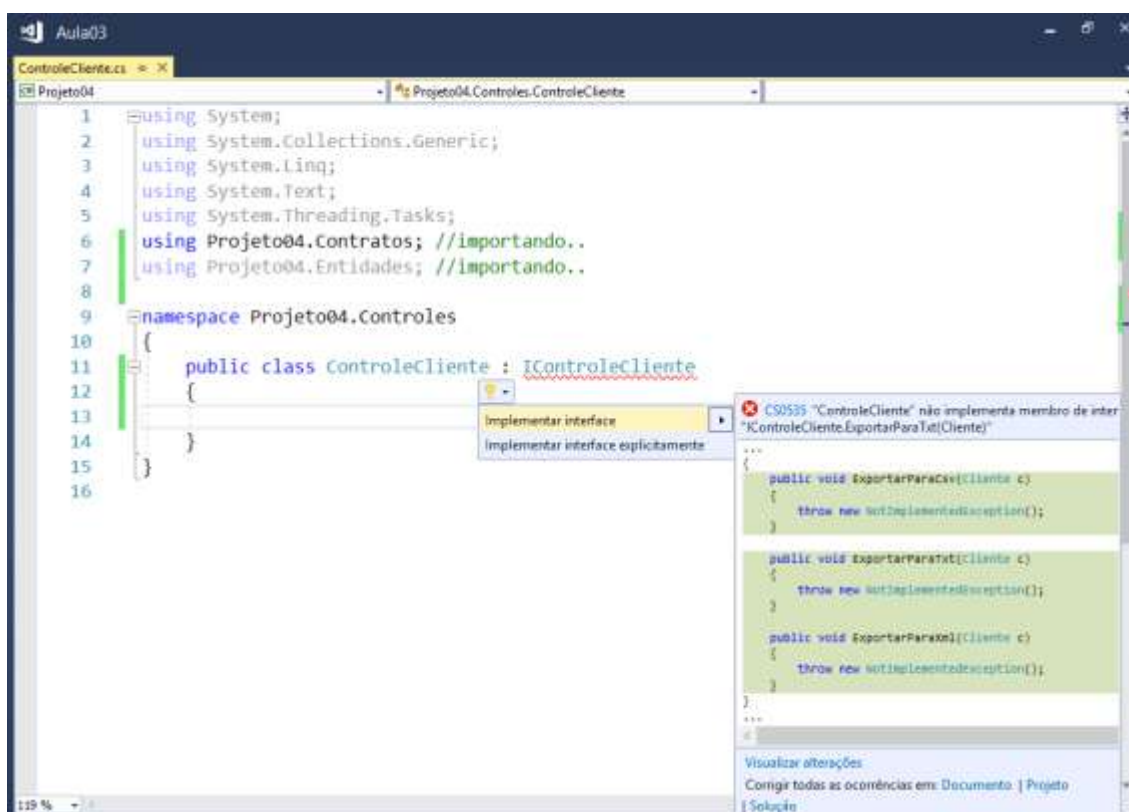
Criando uma classe para implementar a interface:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto04.Contratos; //importando..
using Projeto04.Entidades; //importando..

namespace Projeto04.Controles
{
    public class ControleCliente : IControleCliente
    {
    }
}
```

Implementando a interface:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto04.Contratos; //importando..
using Projeto04.Entidades; //importando..

namespace Projeto04.Controles
{
    public class ControleCliente : IControleCliente
    {

```

```
public void ExportarParaCsv(Cliente c)
{
    throw new NotImplementedException();
}

public void ExportarParaTxt(Cliente c)
{
    throw new NotImplementedException();
}

public void ExportarParaXml(Cliente c)
{
    throw new NotImplementedException();
}
}
}
```

Implementando os métodos da interface:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto04.Contratos; //importando..
using Projeto04.Entidades; //importando..
using System.IO;
using System.Xml;

namespace Projeto04.Controles
{
    public class ControleCliente : IControleCliente
    {
        public void ExportarParaCsv(Cliente c)
        {
            StreamWriter sw = new StreamWriter("c:\\temp\\clientes.csv", true);
            sw.WriteLine($"{c.IdCliente};{c.Nome};{c.Email}");
            sw.Close();
        }

        public void ExportarParaTxt(Cliente c)
        {
            StreamWriter sw = new StreamWriter("c:\\temp\\clientes.txt", true);
            sw.WriteLine("Id do Cliente....: " + c.IdCliente);
            sw.WriteLine("Nome.....: " + c.Nome);
            sw.WriteLine("Email.....: " + c.Email);
            sw.WriteLine("...");
            sw.Close();
        }
    }
}
```

```
public void ExportarParaXml(Cliente c)
{
    XmlWriter xml = XmlWriter.Create("c:\\temp\\clientes.xml");
    xml.WriteStartDocument(); //<?xml version='1.0' encoding='UTF-8'?>

    xml.WriteStartElement("cliente"); //<cliente>

        xml.WriteElementString("idcliente", c.IdCliente.ToString());
        //<idcliente>1</idcliente>
        xml.WriteElementString("nome", c.Nome);
        //<nome>Ana</nome>
        xml.WriteElementString("email", c.Email);
        //<email>ana@gmail.com</email>

    xml.WriteEndElement(); //</cliente>
    xml.Close();
}
}
```

Executando:

```
using Projeto04.Controles;
using Projeto04.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto04
{
    class Program
    {
        static void Main(string[] args)
        {
            Cliente c = new Cliente(1, "Sergio Mendes", "sergio.coti@gmail.com");

            try //tentativa
            {
```

```

ControleCliente cc = new ControleCliente();

cc.ExportarParaTxt(c);
cc.ExportarParaCsv(c);
cc.ExportarParaXml(c);

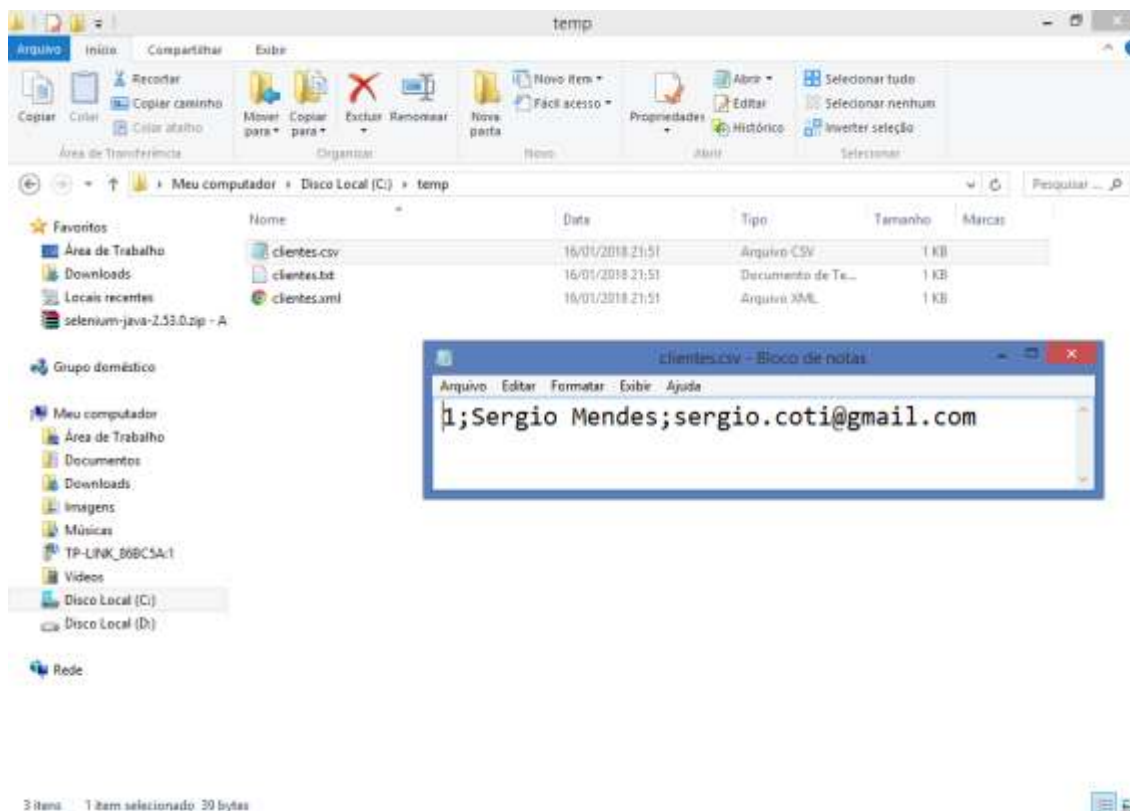
Console.WriteLine("\nArquivos gerados com sucesso.");
}
catch(Exception e) //captura da exceção..
{
    //imprimir mensagem de erro..
    Console.WriteLine("Erro: " + e.Message);
}

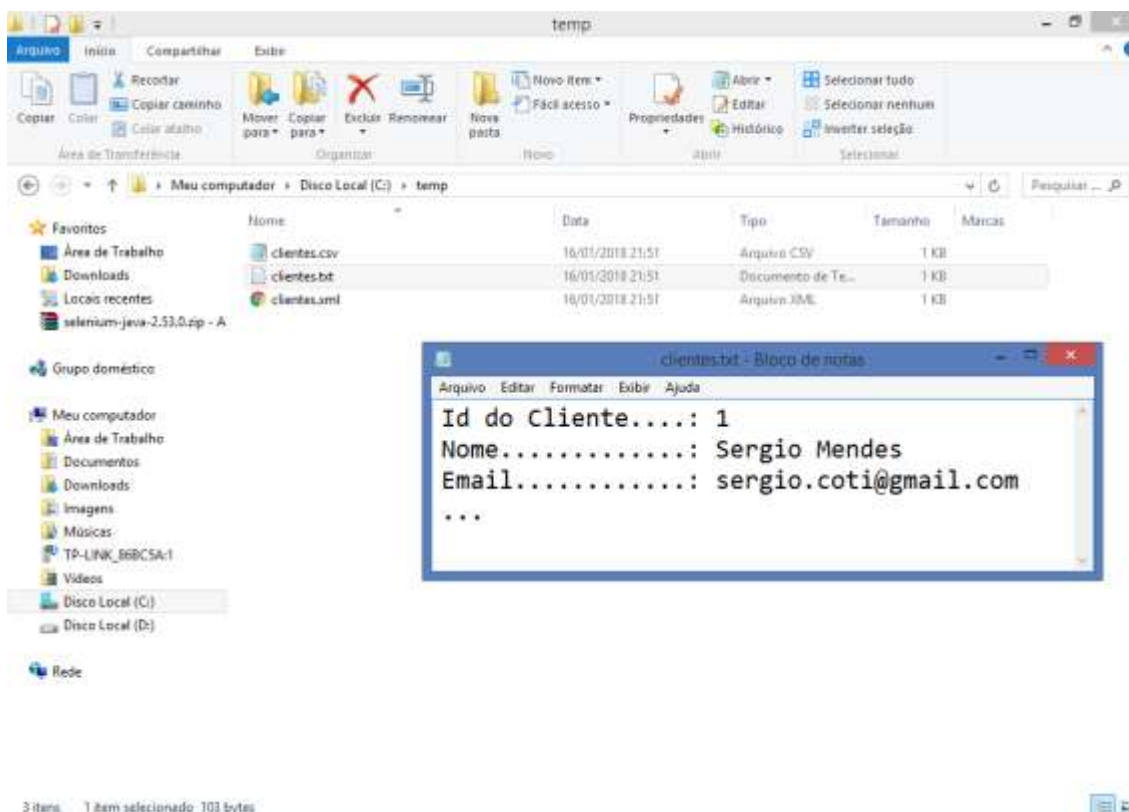
Console.ReadKey();
}
}
}

```

Saida do programa:

Arquivos gerados...





temp

Arquivo Início Compartilhar Exibir

Copiar Colar Recortar Copiar caminho Copiar atalho Área de Transferência Organizar Novo Novo pasta Propriedades Abrir Histórico Selecionar tudo Selecionar nenhum Inverter seleção Selecionar

Meu computador > Disco Local (C:) > temp

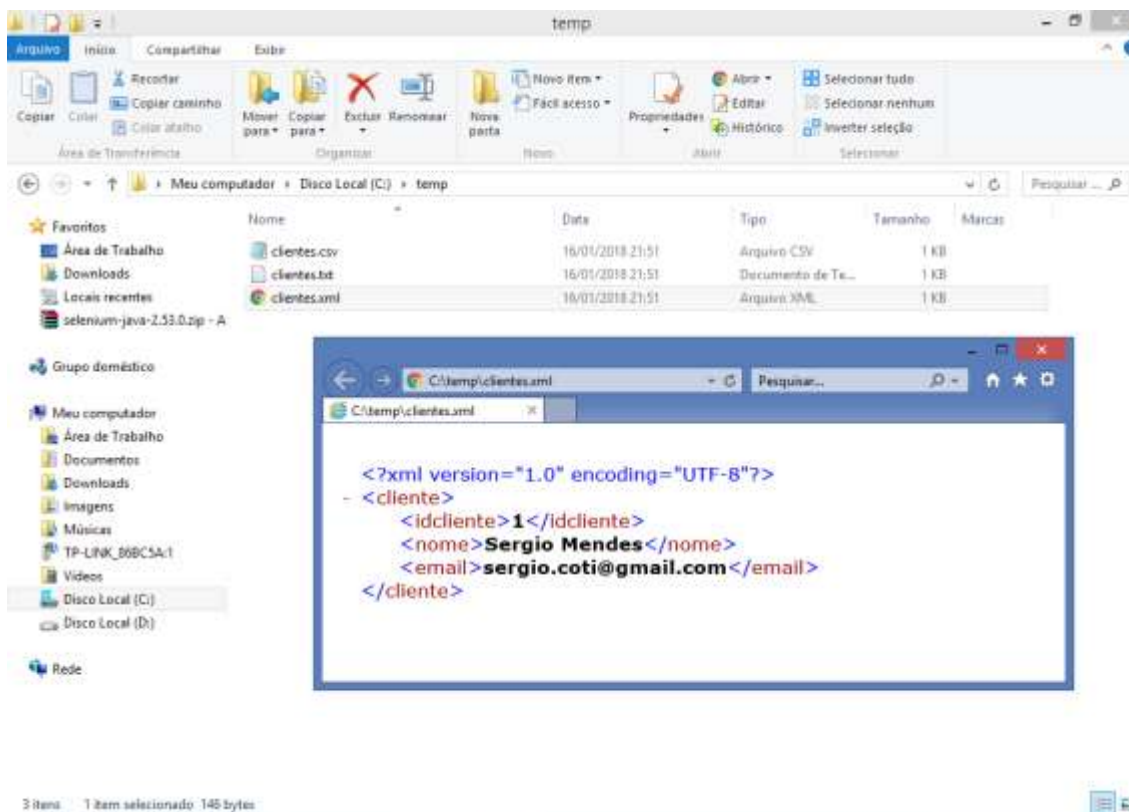
Nome	Data	Tipo	Tamanho	Marca
clientes.csv	16/01/2018 21:51	Arquivo CSV	1 KB	
clientes.txt	16/01/2018 21:51	Documento de Te...	1 KB	
clientes.xml	16/01/2018 21:51	Arquivo XML	1 KB	

clientes.txt - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

Id do Cliente....: 1
Nome.....: Sergio Mendes
Email.....: sergio.coti@gmail.com
...

3 itens 1 item selecionado 103 bytes



temp

Arquivo Início Compartilhar Exibir

Copiar Colar Recortar Copiar caminho Copiar atalho Área de Transferência Organizar Novo Novo pasta Propriedades Abrir Histórico Selecionar tudo Selecionar nenhum Inverter seleção Selecionar

Meu computador > Disco Local (C:) > temp

Nome	Data	Tipo	Tamanho	Marca
clientes.csv	16/01/2018 21:51	Arquivo CSV	1 KB	
clientes.txt	16/01/2018 21:51	Documento de Te...	1 KB	
clientes.xml	16/01/2018 21:51	Arquivo XML	1 KB	

C:\temp\clientes.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cliente>
  <idcliente>1</idcliente>
  <nome>Sergio Mendes</nome>
  <email>sergio.coti@gmail.com</email>
</cliente>
```

3 itens 1 item selecionado 146 bytes