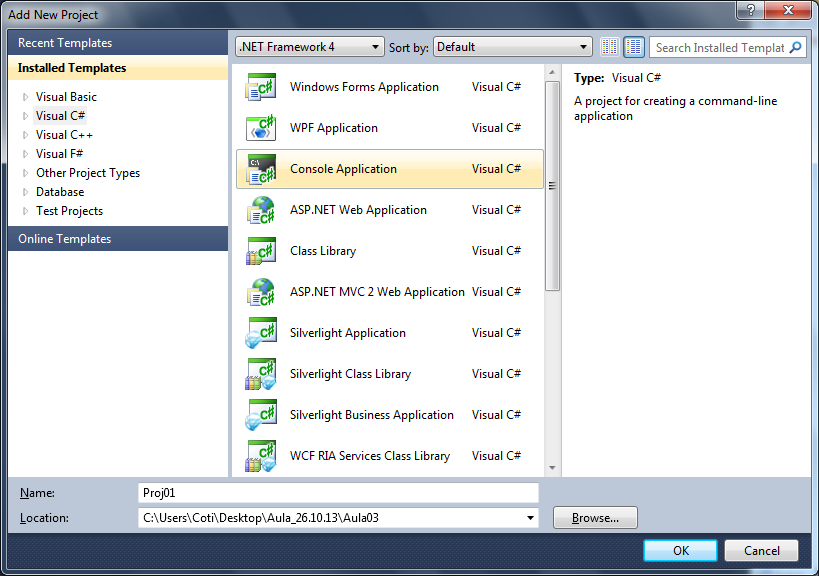


Criação do Projeto Console (DOS)



Relacionamentos entre Classes:

* Herança (**SER**)
* Associação (**TER**)
  + Multiplicidades:
    - 1 para 1
    - 1 para Muitos / Muitos para 1
    - Muitos para Muitos

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Proj01.Model

{

public class Funcionario

{

//Atributos

private int idFuncionario;

private string nome;

private double salario;

**private Departamento depto; //Associação (TER-1)**

//Construtores [ctor + 2xtab]

public Funcionario()

{

//Default

}

//Sobrecarga de Construtores (Overloading)

public Funcionario(int idFuncionario, string nome, double salario)

{

this.idFuncionario = idFuncionario;

this.nome = nome;

this.salario = salario;

}

//Propriedades

public int IdFuncionario

{

set { idFuncionario = value; }

get { return idFuncionario; }

}

public string Nome

{

set { nome = value; }

get { return nome; }

}

public double Salario

{

set { salario = value; }

get { return salario; }

}

public Departamento Depto

{

set { depto = value; }

get { return depto; }

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Proj01.Model

{

public class Departamento

{

//[prop] + 2x[tab]

public int IdDepartamento { get; set; }

public string Nome { get; set; }

**public List<Funcionario> Funcionarios { get; set; }**

//Associação (TER-Muitos)

//Construtores

public Departamento()

{

//Default

}

//Sobrecarga de Construtores

public Departamento(int IdDepartamento, string Nome)

{

this.IdDepartamento = IdDepartamento;

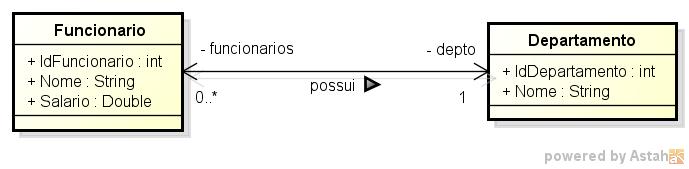
this.Nome = Nome;

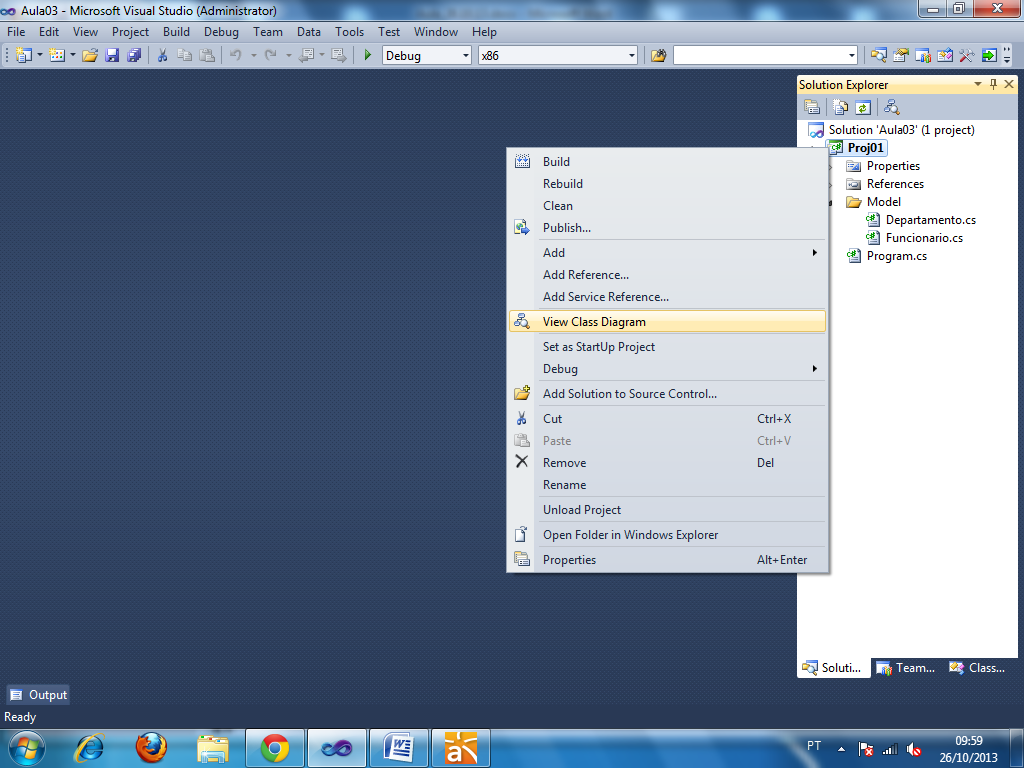
}

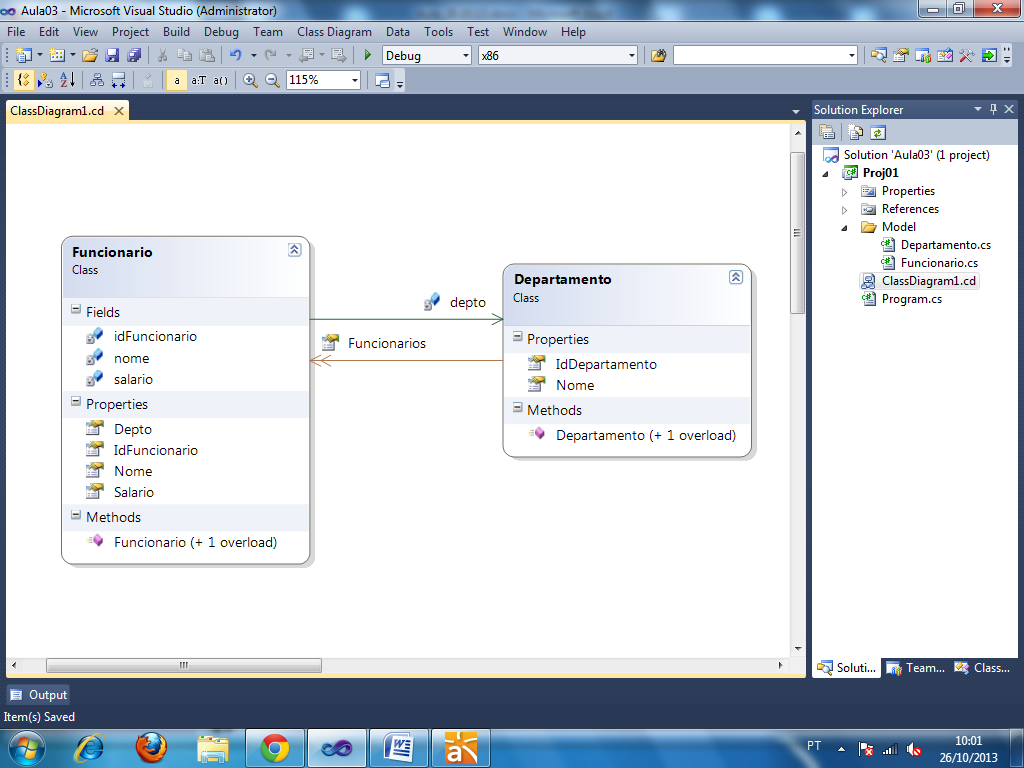
}

}

Diagrama de Classes







Manipulação de Listas de Objetos...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Proj01.Model; //importando o conteudo do namespace

namespace Proj01

{

class Program

{

static void Main(string[] args)

{

Departamento d1 = new Departamento(1, "Desenv de Sistemas");

Departamento d2 = new Departamento(2, "Engenharia");

Funcionario f1 = new Funcionario(1, "Leonardo", 2500.0);

Funcionario f2 = new Funcionario(2, "Bruna", 3000.0);

Funcionario f3 = new Funcionario(3, "Wendel", 4000.0);

Funcionario f4 = new Funcionario(4, "Guilherme", 3500.0);

Funcionario f5 = new Funcionario(5, "Monique", 4500.0);

//Definindo espaço de memória para a lista do Departamento 'd1'

d1.Funcionarios = new List<Funcionario>(); //lista vazia

d1.Funcionarios.Add(f1); //Adicionando elementos na lista

d1.Funcionarios.Add(f2);

d1.Funcionarios.Add(f3);

Console.WriteLine("Departamento: {0}, Nome: {1}",

d1.IdDepartamento, d1.Nome);

Console.WriteLine("Quantidade de Funcionários: " +

d1.Funcionarios.Count);

//Percorrer a lista (for tradicional)

for (int i = 0; i < d1.Funcionarios.Count; i++)

{

Console.WriteLine("Codigo..............: " +

d1.Funcionarios[i].IdFuncionario);

Console.WriteLine("Nome do Funcionario.: " +

d1.Funcionarios[i].Nome);

Console.WriteLine("Salario.............: " +

d1.Funcionarios[i].Salario);

Console.WriteLine("...");

}

//Inicializando a lista do objeto Departamento 'd2'

d2.Funcionarios = new List<Funcionario>(); //Lista vazia

d2.Funcionarios.Add(f4); //Adicionando funcionários dentro da lista

d2.Funcionarios.Add(f5);

Console.WriteLine("Departamento: {0}, Nome: {1}",

d2.IdDepartamento, d2.Nome);

Console.WriteLine("Quantidade de Funcionários: " +

d2.Funcionarios.Count);

//Percorrer a lista (foreach -> para cada...)

//foreach -> for que não utiliza índice (não varre

especificando a posição)

//Para cada elemento da lista, leia um Funcionario 'f'

foreach(Funcionario f in d2.Funcionarios)

{

Console.WriteLine("Codigo..............: " + f.IdFuncionario);

Console.WriteLine("Nome do Funcionario.: " + f.Nome);

Console.WriteLine("Salario.............: " + f.Salario);

Console.WriteLine("...");

}

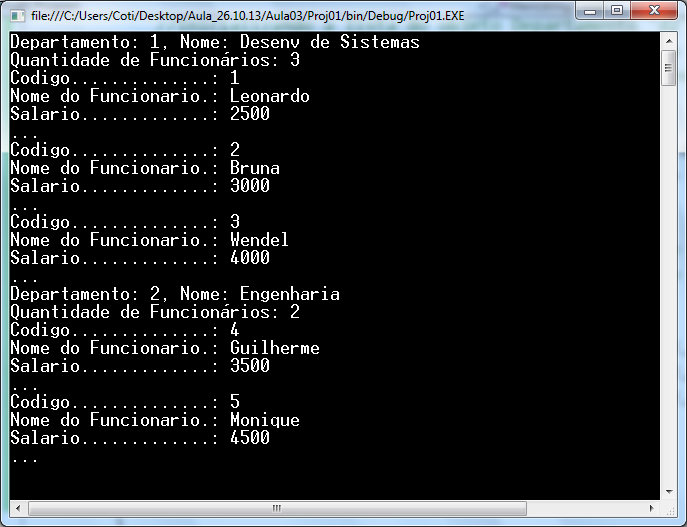
Console.ReadKey(); //Pausar

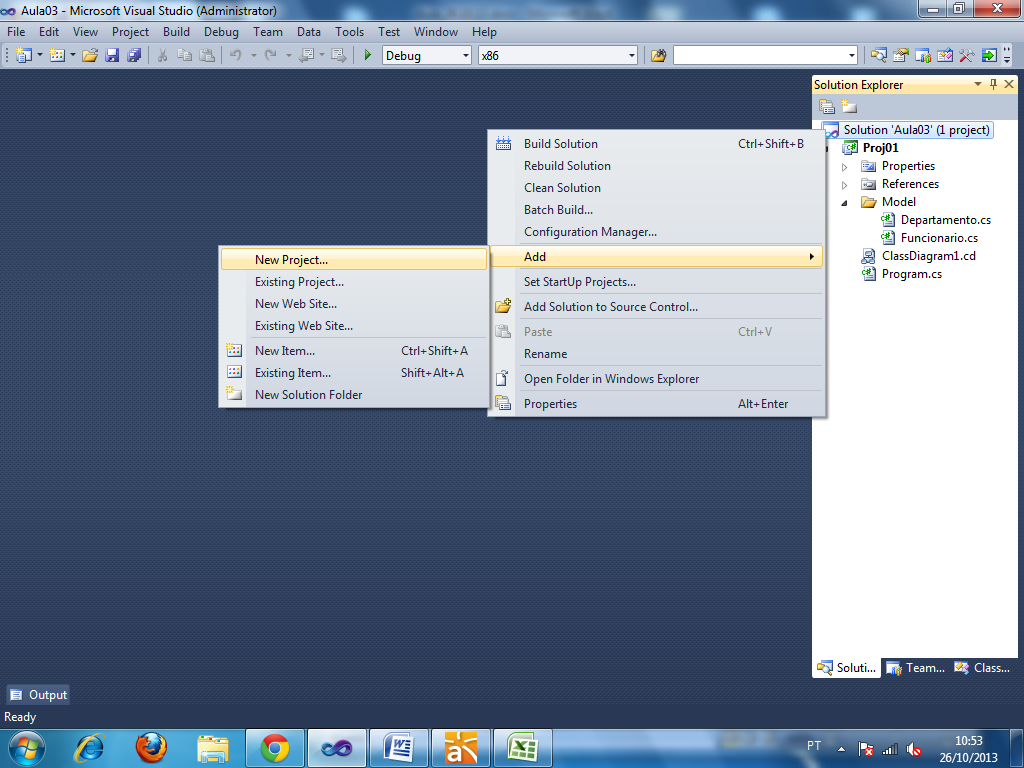
}

}

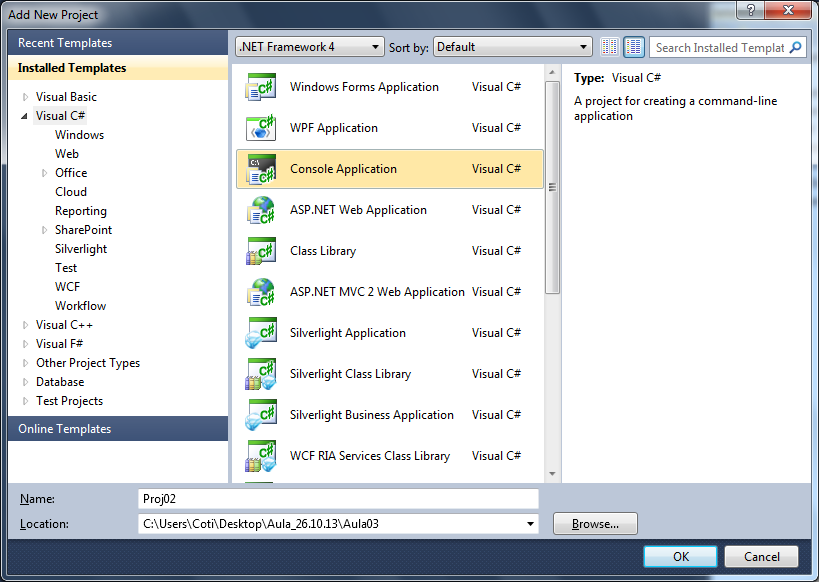
}

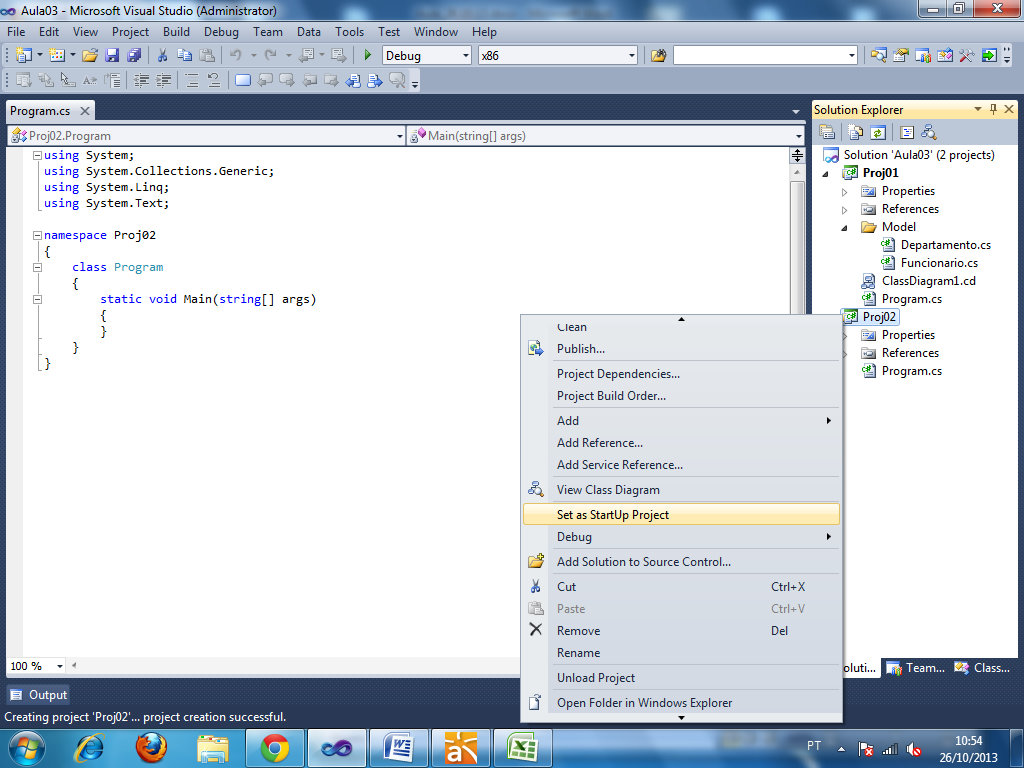
Executando [f5]...

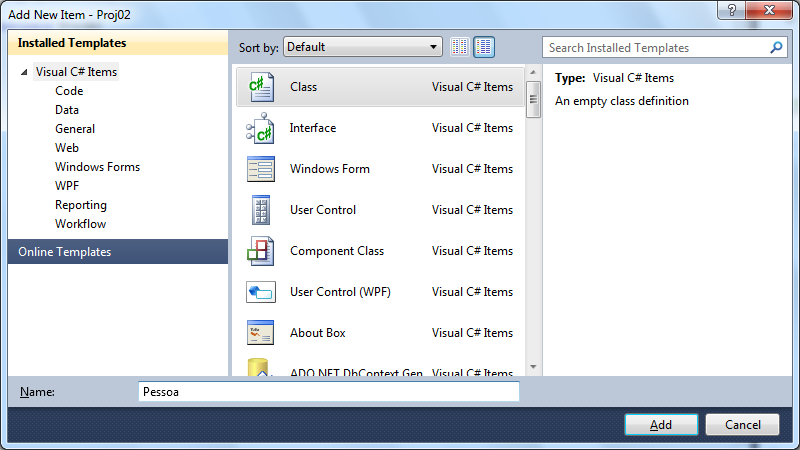




Novo Projeto...







using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Proj02.Model

{

public class Pessoa

{

public int IdPessoa { get; set; }

public string Nome { get; set; }

public char Sexo { get; set; }

public Pessoa() //[ctor] + 2x[tab]

{

//Construtor default

}

public Pessoa(int IdPessoa, string Nome, char Sexo)

{

this.IdPessoa = IdPessoa;

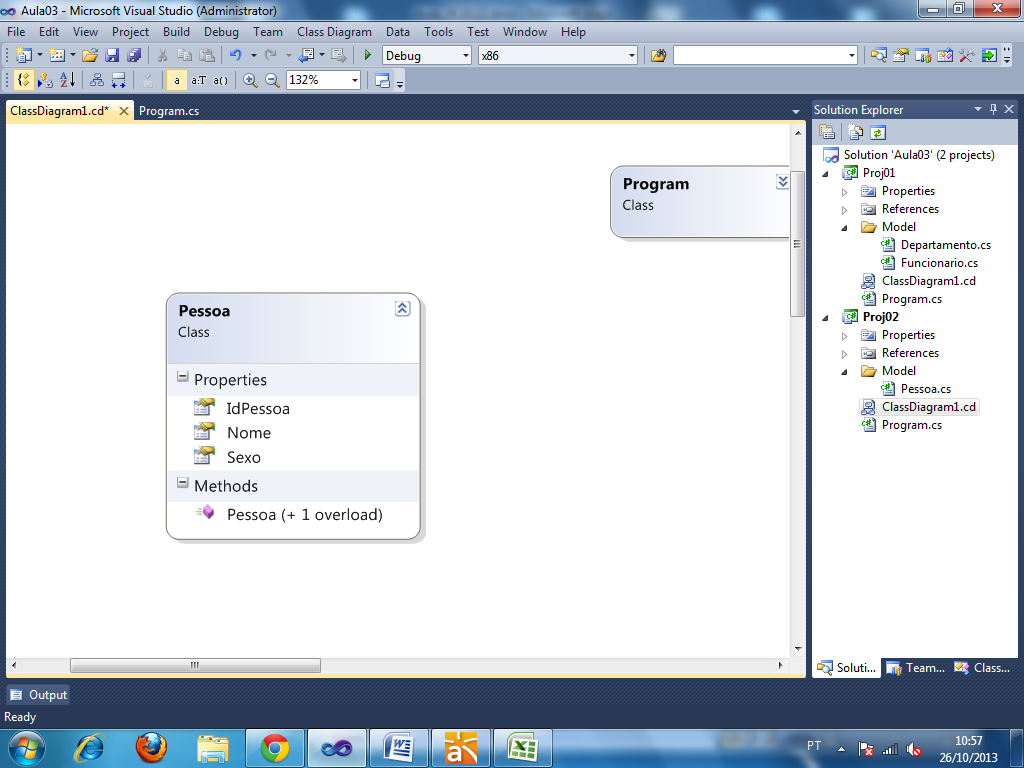
this.Nome = Nome;

this.Sexo = Sexo;

}

}

}



Coleções baseadas em Mapas

Um Mapa é utilizado para armazenamento de coleções de elementos identificados por uma Chave Única.

* **Por exemplo:**

CHAVE VALOR

p1 new Pessoa(1, “Pedro”, ‘M’)

p2 new Pessoa(2, “Joao”, ‘M’)

Classe Dictionary (Mapas)...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Proj02.Model;

namespace Proj02

{

class Program

{

static void Main(string[] args)

{

//Criando uma mapa

**Dictionary<string, Pessoa> mapa = new Dictionary<string, Pessoa>();**

**mapa.Add("p1", new Pessoa(1, "Pedro", 'M'));**

**mapa.Add("p2", new Pessoa(2, "Carla", 'F'));**

**mapa.Add("p3", new Pessoa(3, "João", 'M'));**

**// [chave] [conteudo -> valor]**

Console.WriteLine("Codigo....: " + **mapa["p1"].IdPessoa**);

Console.WriteLine("Nome......: " + **mapa["p1"].Nome**);

Console.WriteLine("Sexo......: " + **mapa["p1"].Sexo**);

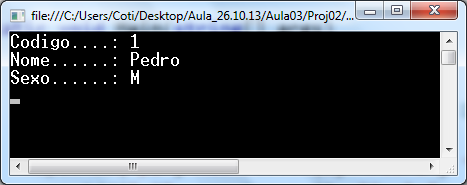
Console.ReadKey();

}

}

}

Executando...



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Proj02.Model;

namespace Proj02

{

class Program

{

static void Main(string[] args)

{

//Criando uma mapa

Dictionary<string, Pessoa> mapa = new Dictionary<string, Pessoa>();

mapa.Add("p1", new Pessoa(1, "Pedro", 'M'));

mapa.Add("p2", new Pessoa(2, "Carla", 'F'));

mapa.Add("p3", new Pessoa(3, "João", 'M'));

// [chave] [conteudo -> valor]

string chave = "p2";

**if (mapa.ContainsKey(chave))**

{

Console.WriteLine("Codigo....: " + **mapa[chave].IdPessoa)**;

Console.WriteLine("Nome......: " + **mapa[chave].Nome)**;

Console.WriteLine("Sexo......: " + **mapa[chave].Sexo)**;

}

else

{

Console.WriteLine("Chave não encontrada.");

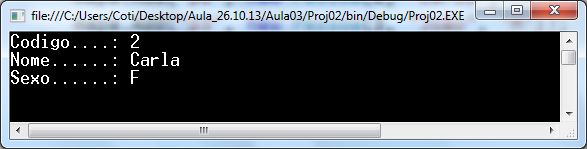
}

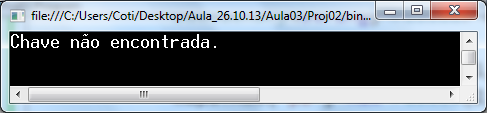
Console.ReadKey();

}

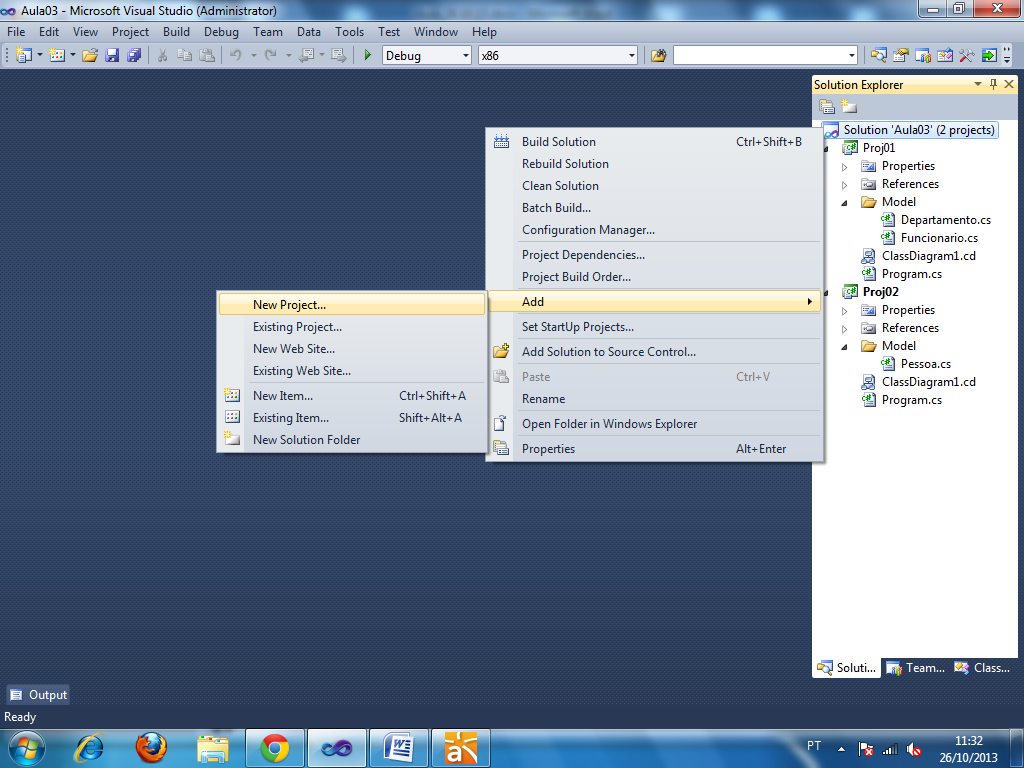
}

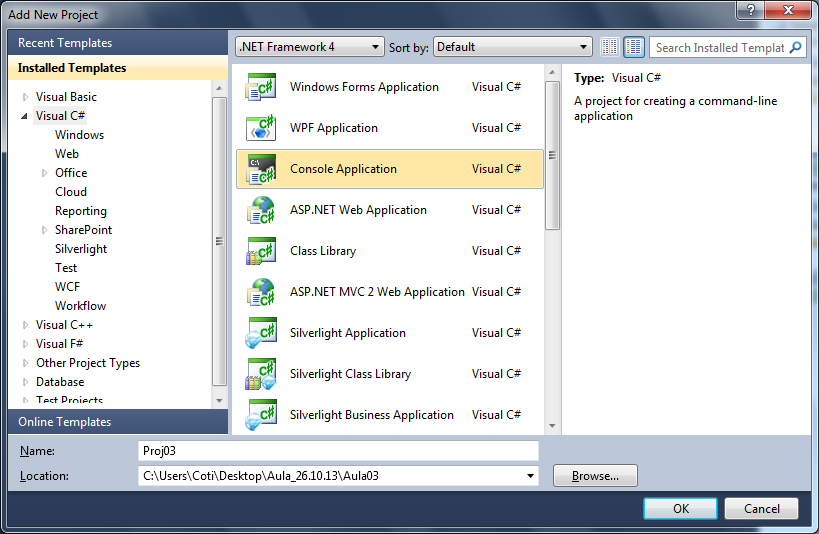
}





Novo Projeto...

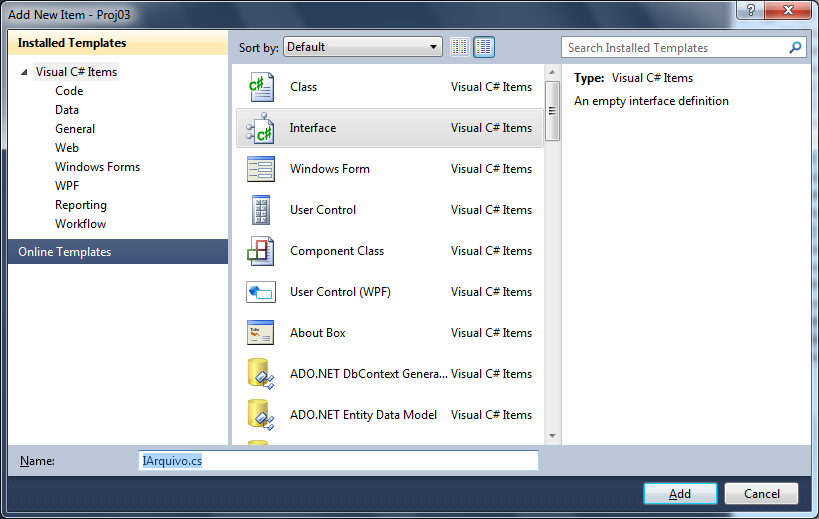




Interfaces

São utilizadas em programação Orientada a Objetos para definir padrões que serão seguidos por Classes.

Obs: Toda interface criada em C# terá, por boa prática, o seu nome começando com a letra **I**



Regras sobre interfaces...

* Interfaces não podem ter atributos
* Interfaces não podem ter construtores
* Métodos em interfaces já são implicitamente públicos, e não podemos sequer especificar sua visibilidade.
* Os Métodos de uma interface não podem ter corpo (conteúdo), apenas assinatura, ou seja, são apenas prototipados

Importante!

Quando uma classe herdar uma ou mais interfaces, a Classe será obrigada a implementar, ou seja, fornecer corpo para todos os métodos declarados ou prototipados na interface.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Proj03.Model

{

//Nível abstrato de programação (prototipação)

public interface IArquivo

{

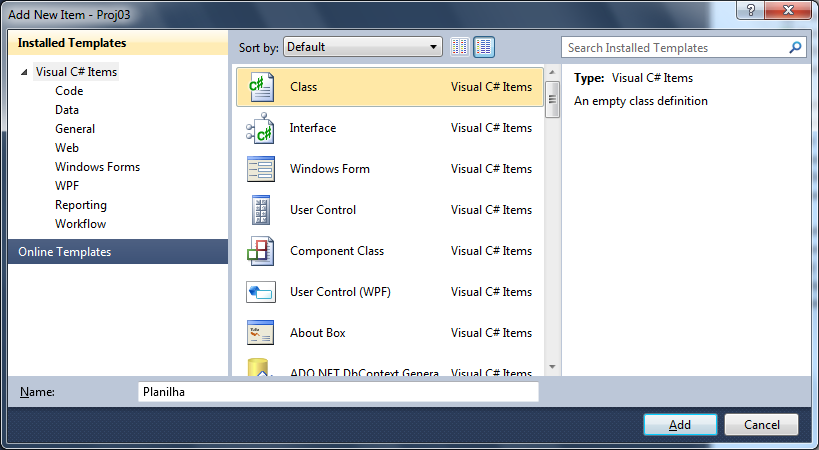
void AbrirArquivo();

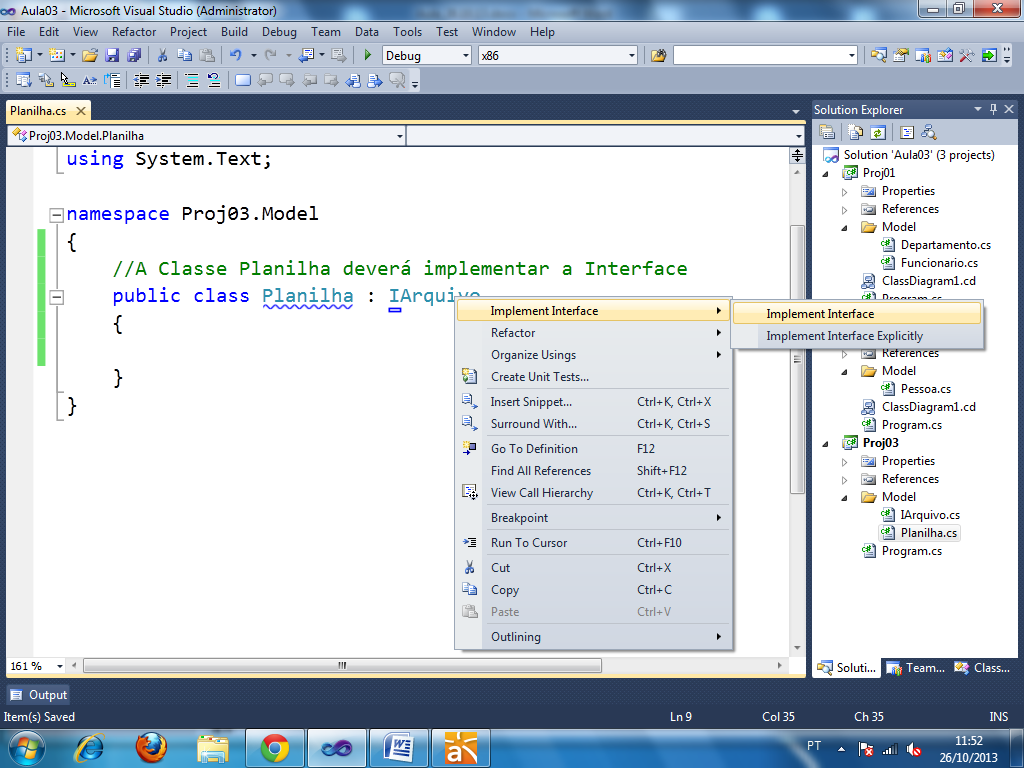
void GravarDados(string conteudo);

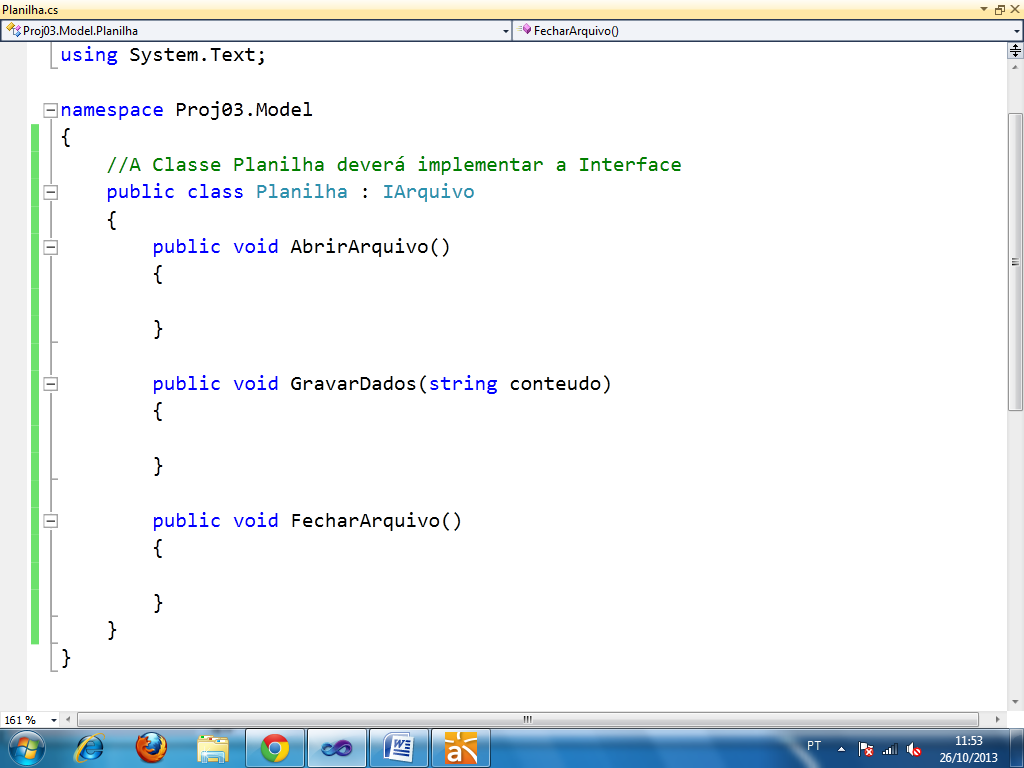
void FecharArquivo();

}

}







using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO; //Arquivos (Input e Output)

namespace Proj03.Model

{

//A Classe Planilha deverá implementar a Interface

public class Planilha : IArquivo

{

//Atributo

private StreamWriter sw;

public void AbrirArquivo()

{

//true -> indicar que o conteudo do arquivo será sempre adicionado

sw = new StreamWriter("d:\\planilha.csv", true);

}

public void GravarDados(string conteudo)

{

sw.WriteLine(conteudo); //Escrever dentro do arquivo

}

public void FecharArquivo()

{

sw.Close(); //Fechar o Arquivo

}

}

}

Demais regras sobre interfaces...

* Uma Classe pode herdar muitas interfaces...

**interface IA**

{

void MethodA();

}

**interface IB**

{

void MethodB();

}

**class C : IA, IB**

{

public void MethodA()

{

}

public void MethodB()

{

}

}

* Uma interface pode herdar outras interfaces...

interface IA

{

void MethodA();

}

interface IB

{

void MethodB();

}

**interface IC : IA, IB**

{

void MethodC();

}

**class D : IC**

{

public void MethodC()

{

}

public void MethodA()

{

}

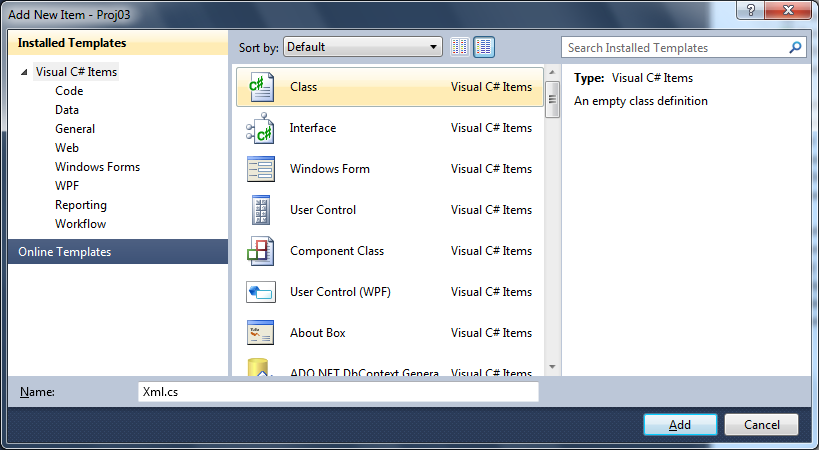
public void MethodB()

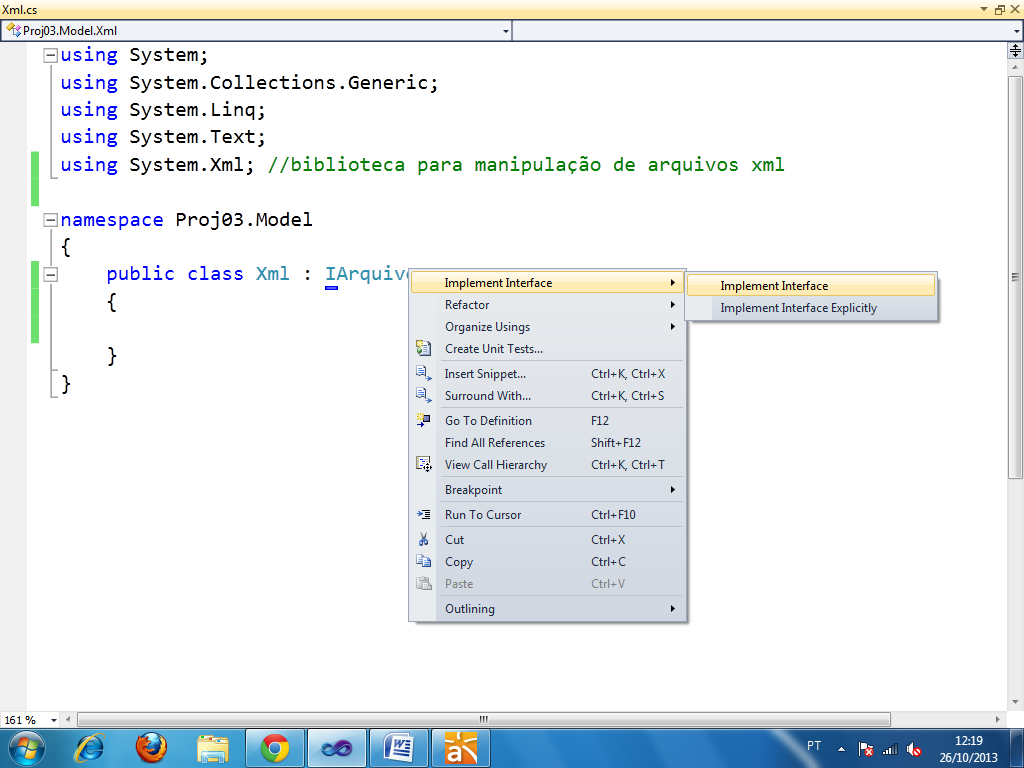
{

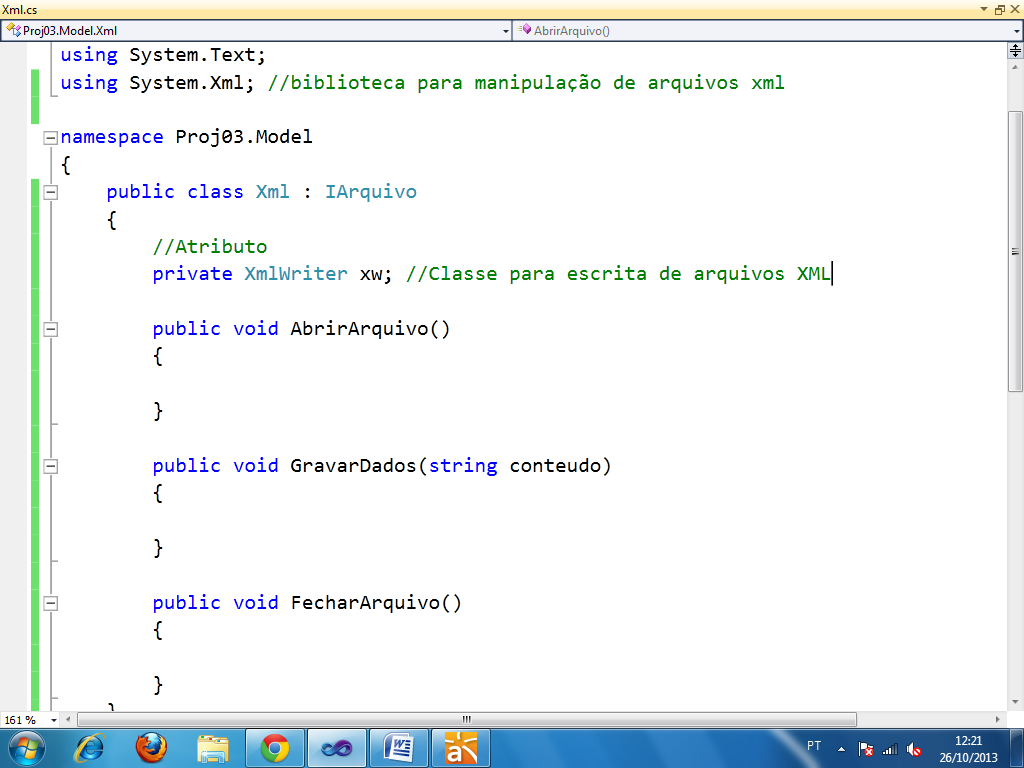
}

}

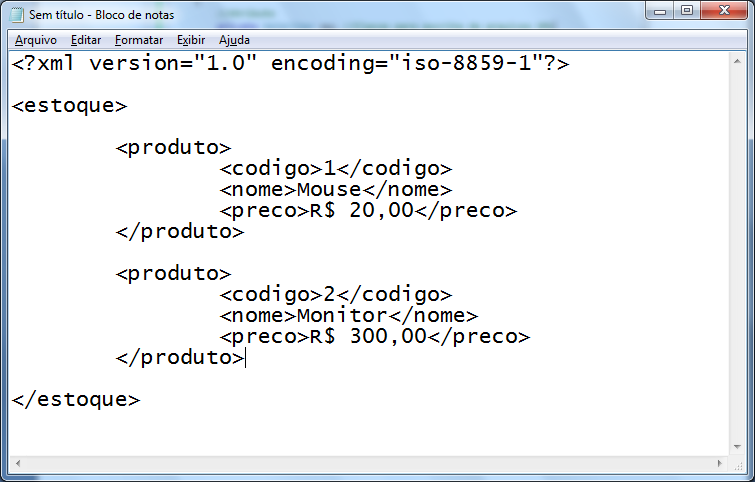
Gravação de arquivo XML

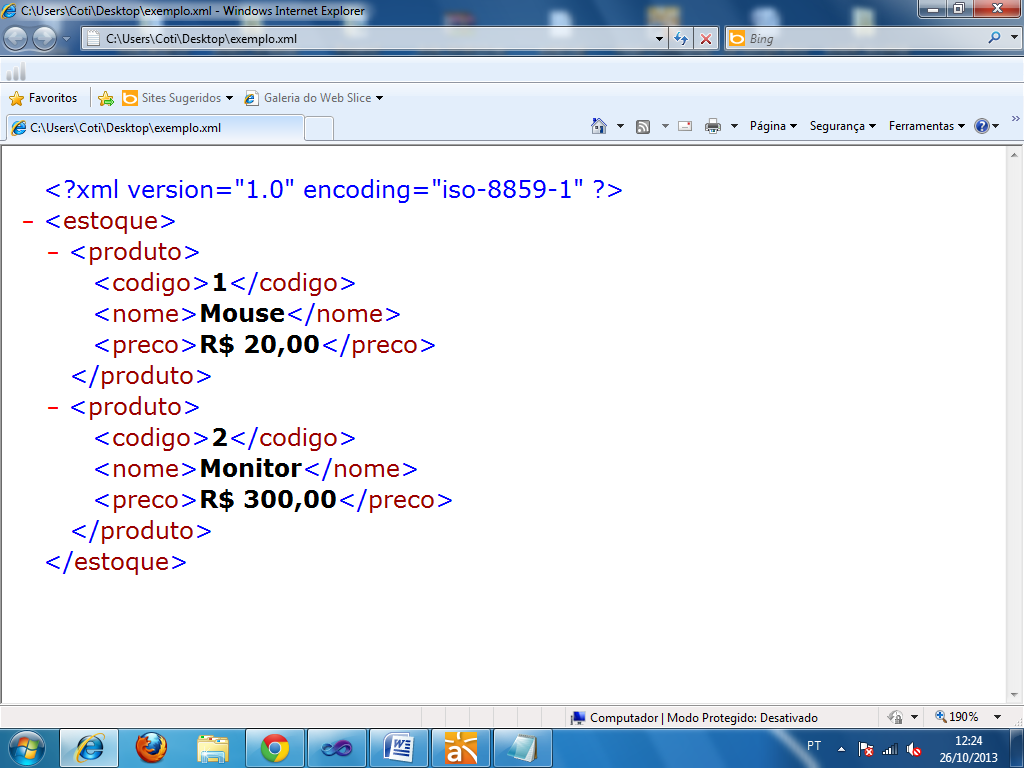






Exemplo de arquivo XML





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Xml; //biblioteca para manipulação de arquivos xml

namespace Proj03.Model

{

public class Xml : IArquivo

{

private XmlWriter xw; //Classe para escrita de arquivos XML

public void AbrirArquivo()

{

//Criar um arquivo de extensão XML

xw = XmlWriter.Create("d:\\dados.xml");

xw.WriteStartDocument(); //cria o cabeçalho XML

}

public void GravarDados(string conteudo)

{

xw.WriteStartElement("conteudo"); //criar uma tag <conteudo>

xw.WriteElementString("texto", conteudo); //<texto>..</texto>

xw.WriteEndElement(); //fechar a tag </conteudo>

}

public void FecharArquivo()

{

xw.WriteEndDocument(); //finalizar o XML

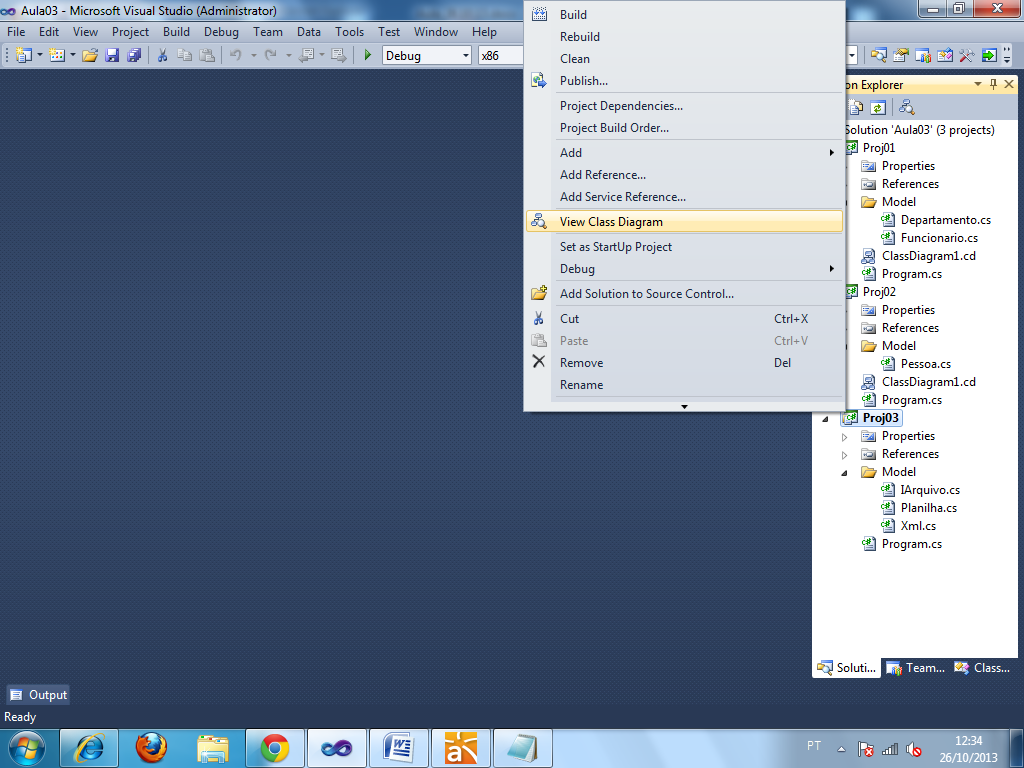
xw.Close(); //fechando o arquivo

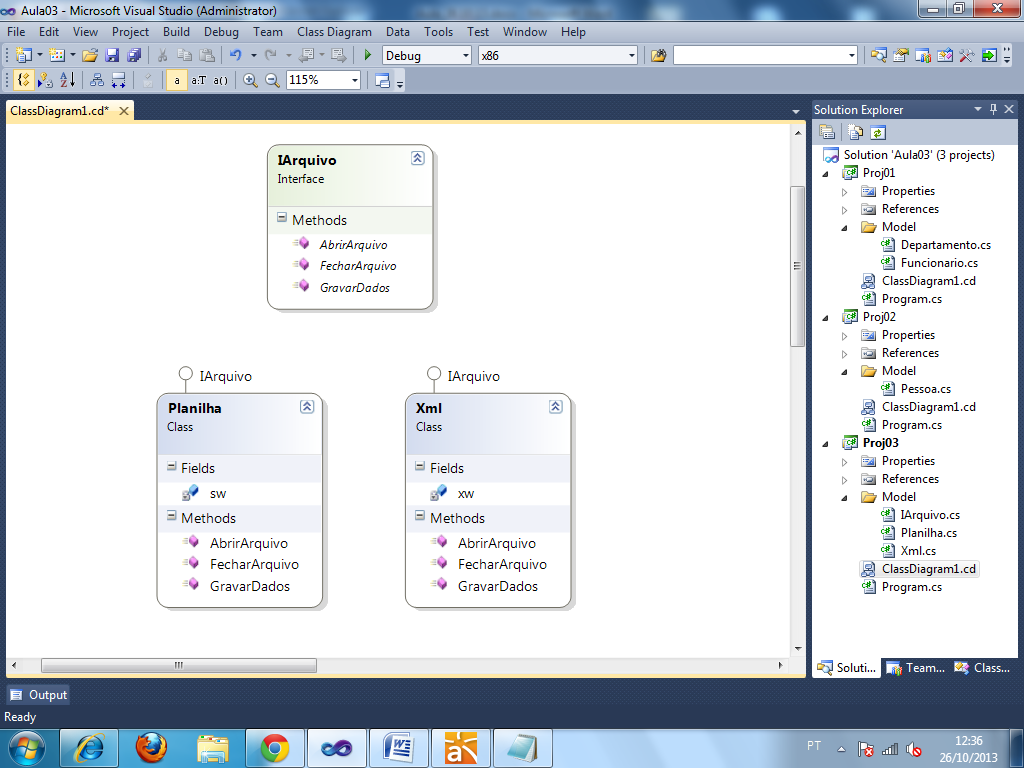
}

}

}

Diagrama de Classes...





Polimorfismo

Capacidade de um programa orientado a objetos mudar o seu comportamento em tempo de execução. Mais precisamente: alterar o seu comportamento dependendo da instância de um objeto...

\*\* Não podemos instanciar uma interface pois esta não possui um construtor:

IArquivo a = new IArquivo();

Porém, podemos instanciar interfaces utilizando os construtores das Classes que implementam a interface.

IArquivo a = **new Planilha();**

IArquivo a = **new Xml();**

[Interface] [Objeto] [Instância 🡪 Polimorfismo]

Podemos afirmar que o objeto criado a partir da interface terá o seu comportamento alterado dependendo da instância que lhe foi passada na inicialização.

Exemplo...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Proj03.Model;

namespace Proj03

{

class Program

{

static void Main(string[] args)

{

IArquivo a = null; //sem espaço de memória

try

{

Console.Write("Digite [1]Excel ou [2]XML..: ");

int opcao = Convert.ToInt32(Console.ReadLine());

Console.Write("Digite seu Nome............: ");

string conteudo = Console.ReadLine();

switch(opcao)

{

case 1:

a = new Planilha(); //Polimorfismo

break;

case 2:

a = new Xml(); //Polimorfismo

break;

}

a.AbrirArquivo();

a.GravarDados(conteudo);

a.FecharArquivo();

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

}

catch(Exception e)

{

Console.WriteLine("Erro: " + e.Message);

}

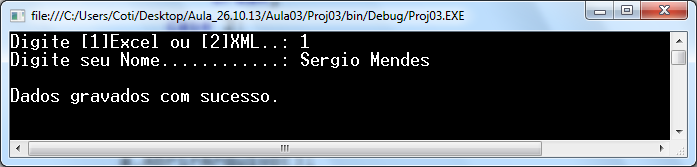
Console.ReadKey(); //Pausar

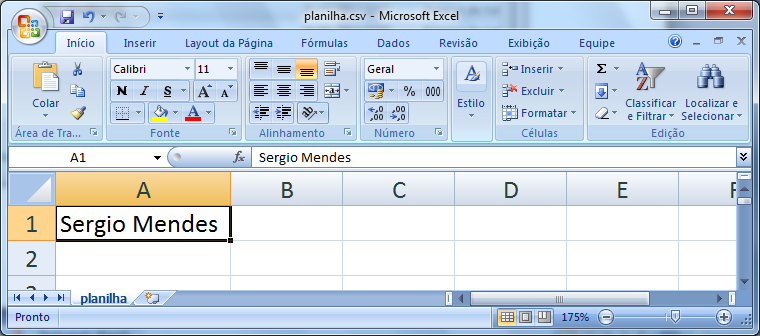
}

}

}

Executando...





Executando...

