

Linguagem de Programação 3

Aula 1

Introdução ao .NET Framework e ao Visual Studio

`l.bertholdo@ifsp.edu.br`

Conteúdo

- O que é .NET Framework?
 - Histórico
 - Estrutura
 - Common Language Runtime – CLR
 - Assembly
 - Namespace
 - Class Library
- Visual Studio
 - Estrutura Básica de um Programa C#
 - Classes
 - Declaração de Variáveis e Constantes
 - Leitura e Apresentação de Dados
 - Comentários
 - Regras e Convenções de Nomenclatura
 - Exemplo de Programa em C#

O que é .NET Framework?

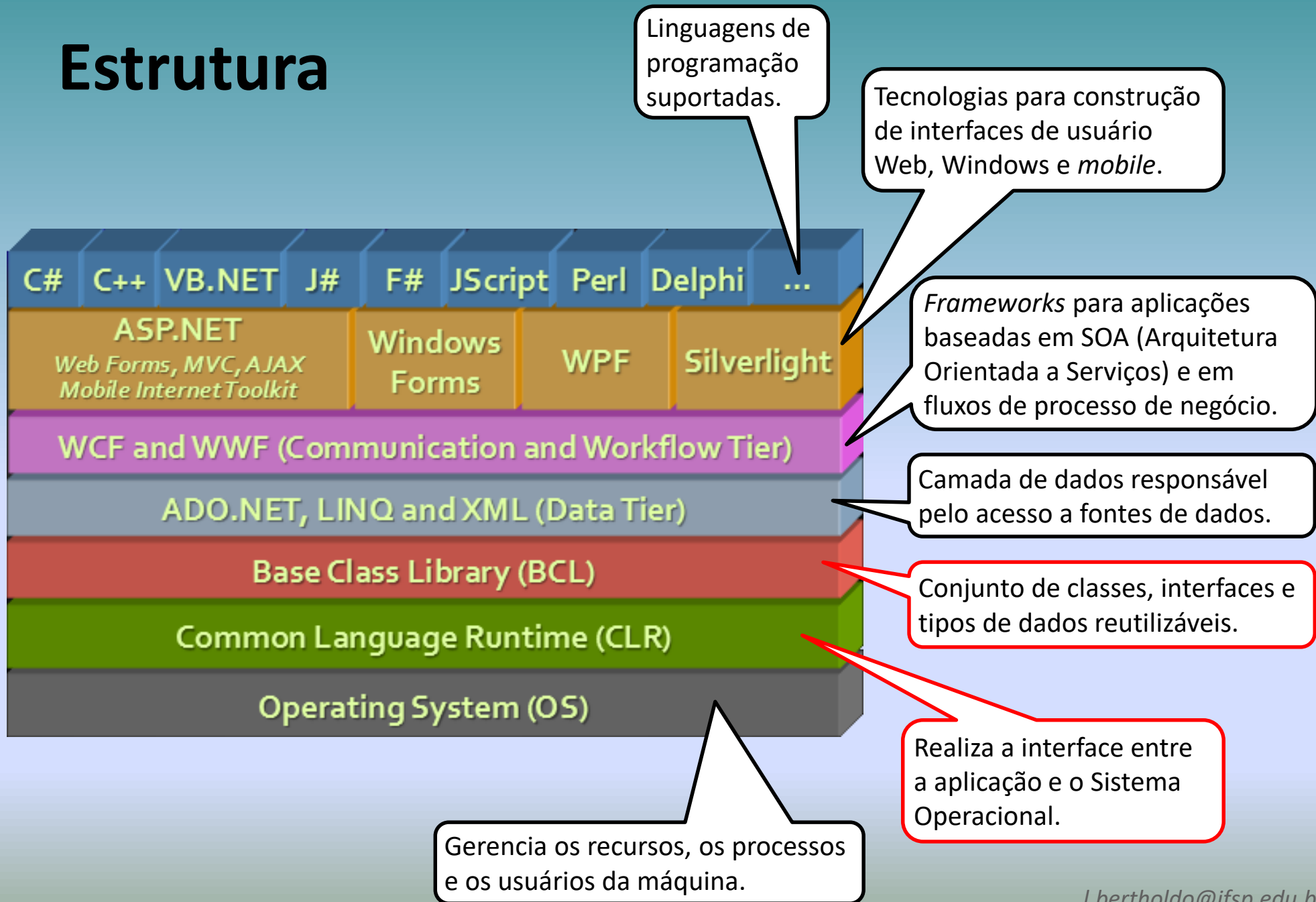
- Plataforma de software para desenvolvimento e execução de aplicações Windows, web e *mobile*.
- As aplicações geradas na plataforma .NET podem ser executadas em qualquer dispositivo que possua este *framework*.
- A plataforma .NET fornece recursos para executar várias linguagens de programação, entre elas: C#, C++ e Visual Basic.

Histórico

- O desenvolvimento da plataforma .NET foi iniciado no final da década de 1990, originalmente com o nome de *Next Generation Windows Services (NGWS)*.

Version number	CLR version	Release date	Development tool	Included in	
				Windows	Windows Server
1.0	1.0	2002-02-13	Visual Studio .NET ^[23]	XP SP1 ^[a]	N/A
1.1	1.1	2003-04-24	Visual Studio .NET 2003 ^[23]	XP SP2, SP3 ^[b]	2003
2.0	2.0	2005-11-07	Visual Studio 2005 ^[25]	N/A	2003, 2003 R2, ^[26] 2008 SP2, 2008 R2 SP1
3.0	2.0	2006-11-06	Expression Blend ^{[27][c]}	Vista	2008 SP2, 2008 R2 SP1
3.5	2.0	2007-11-19	Visual Studio 2008 ^[28]	7, 8, 8.1, 10 ^[d]	2008 R2 SP1
4.0	4	2010-04-12	Visual Studio 2010 ^[29]	N/A	N/A
4.5	4	2012-08-15	Visual Studio 2012 ^[30]	8	2012
4.5.1	4	2013-10-17	Visual Studio 2013 ^[31]	8.1	2012 R2
4.5.2	4	2014-05-05	N/A	N/A	N/A
4.6	4	2015-07-20	Visual Studio 2015 ^[32]	10 v1507	N/A
4.6.1	4	2015-11-30 ^[33]	Visual Studio 2015 Update 1	10 v1511	N/A
4.6.2	4	2016-08-02 ^[34]		10 v1607	2016
4.7	4	2017-04-05 ^[35]	Visual Studio 2017	10 v1703	N/A
4.7.1	4	2017-10-17 ^[36]	Visual Studio 2017	10 v1709	2016 v1709
4.7.2	4	2018-04-30 ^[37]	Visual Studio 2017	10 v1803	2019
4.8	4	Developing ^[38]	Visual Studio 2019 (Planning) ^[39]	10 v1903 (Planning)	N/A

Estrutura



Common Language Runtime – CLR

- Ambiente de tempo de execução (*runtime*) que faz a interface entre a aplicação e o sistema operacional.
- O CLR é análogo à máquina virtual da plataforma Java (*Java Runtime Environment – JRE*), sendo responsável por tarefas como: gerenciamento de memória, verificação de segurança do código, tratamento de erros, coleta de lixo.
- O código gerado pelo compilador e executado neste ambiente é chamado de **código gerenciado**.

Common Language Runtime – CLR

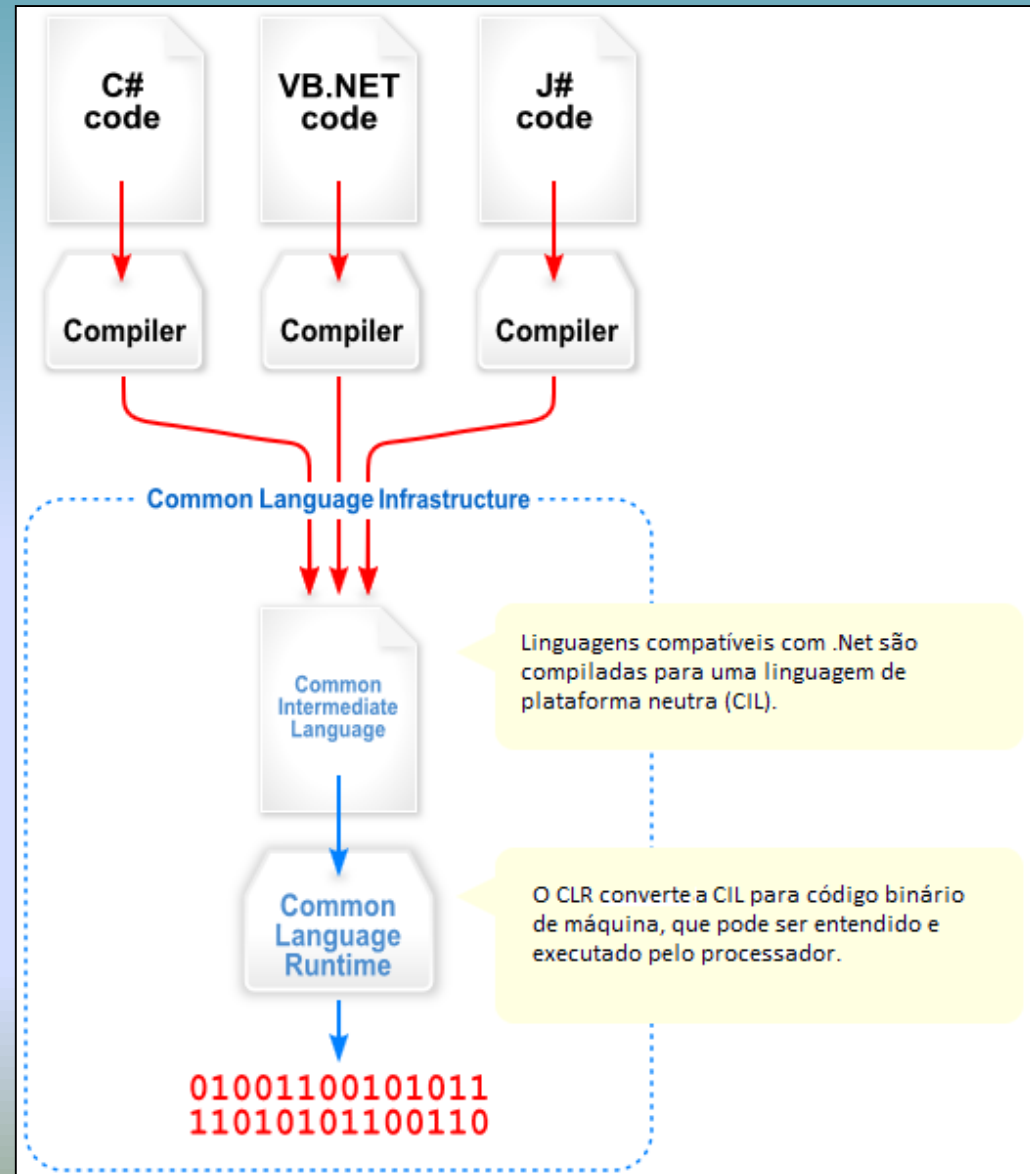
- Common Language Infrastructure – CLI
 - Especificação desenvolvida pela Microsoft que descreve o código executável e o ambiente de tempo de execução, os quais formam o núcleo da plataforma .NET.
 - A CLI permite que códigos compilados em diferentes linguagens de alto nível possam ser executados dentro de um mesmo ambiente de tempo de execução, no caso o CLR.

Common Language Runtime – CLR

- Common Intermediate Language – CIL
 - Linguagem de baixo nível definida pela especificação CLI. Equivale ao *bytecode* gerado na plataforma Java.
 - Trata-se de um código intermediário que contém metadados descritivos do código, tipos de dados declarados e métodos implementados em uma aplicação.
 - Somente linguagens de programação que tenham ambiente *runtime* compatível com a especificação CLI podem converter seu código para instruções CIL.
 - A CIL era originalmente conhecida como *Microsoft Intermediate Language (MSIL)* durante as versões beta das linguagens .NET.

Common Language Runtime – CLR

- CLI x CIL x CLR



Common Language Runtime – CLR

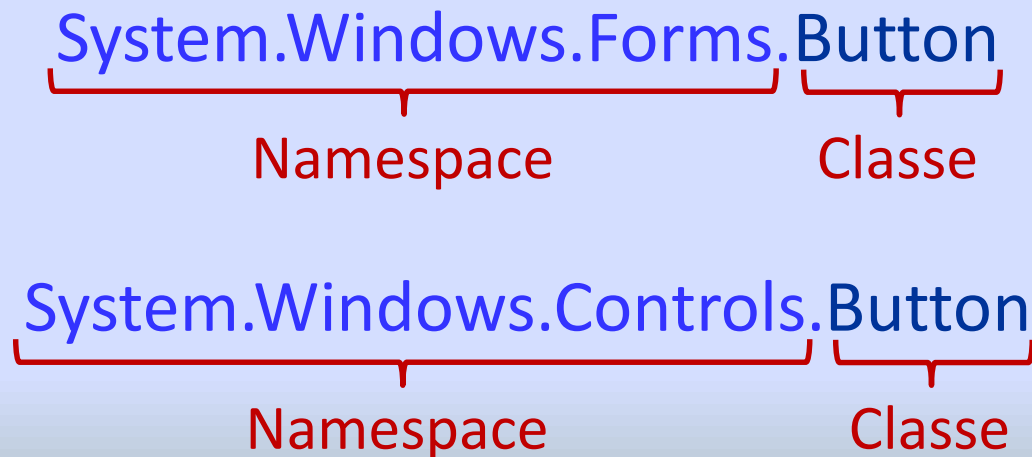
- Garbage Collection (Coleta de Lixo)
 - Mecanismo do CLR responsável pelo monitoramento do uso de memória quando se cria um objeto.
 - O Garbage Collection descarta de forma automática os objetos que não são mais usados por uma aplicação, liberando a memória utilizada.

Assembly

- Arquivo executável, DLL (*Dynamic-Link Library*) ou EXE, usado para “empacotar” aplicações no ambiente .NET.
- Um assembly contém instruções em código intermediário compilado (instruções CIL) que possuem todas as informações necessárias para a execução da aplicação.
- Entre as informações estão: metadados de tipos, arquivos de imagem e texto usados pela aplicação, descrição de como os elementos do assembly se relacionam entre si, dados sobre a versão do assembly, etc.

Namespace

- Recurso usado para organizar classes de forma hierárquica, evitando ambiguidades e simplificando o referenciamento ao utilizar uma biblioteca de classes.
- Por exemplo: duas classes com o mesmo nome não serão confundidas se estiverem em *namespaces* diferentes.



Class Library do .NET Framework

- Extensa biblioteca de classes, interfaces e tipos de dados reutilizáveis que pode ser utilizada por aplicações escritas em qualquer linguagem suportada pelo .NET Framework.
- Representa a base na qual os aplicativos, componentes e controles da plataforma .NET são criados.
- As classes da biblioteca .NET Framework são organizadas por meio dos *namespaces*.
- O principal *namespace* do .NET Framework é o **System**. Ele contém classes básicas que definem tipos de dados, eventos, manipuladores de eventos, interfaces, atributos e outros.

Class Library

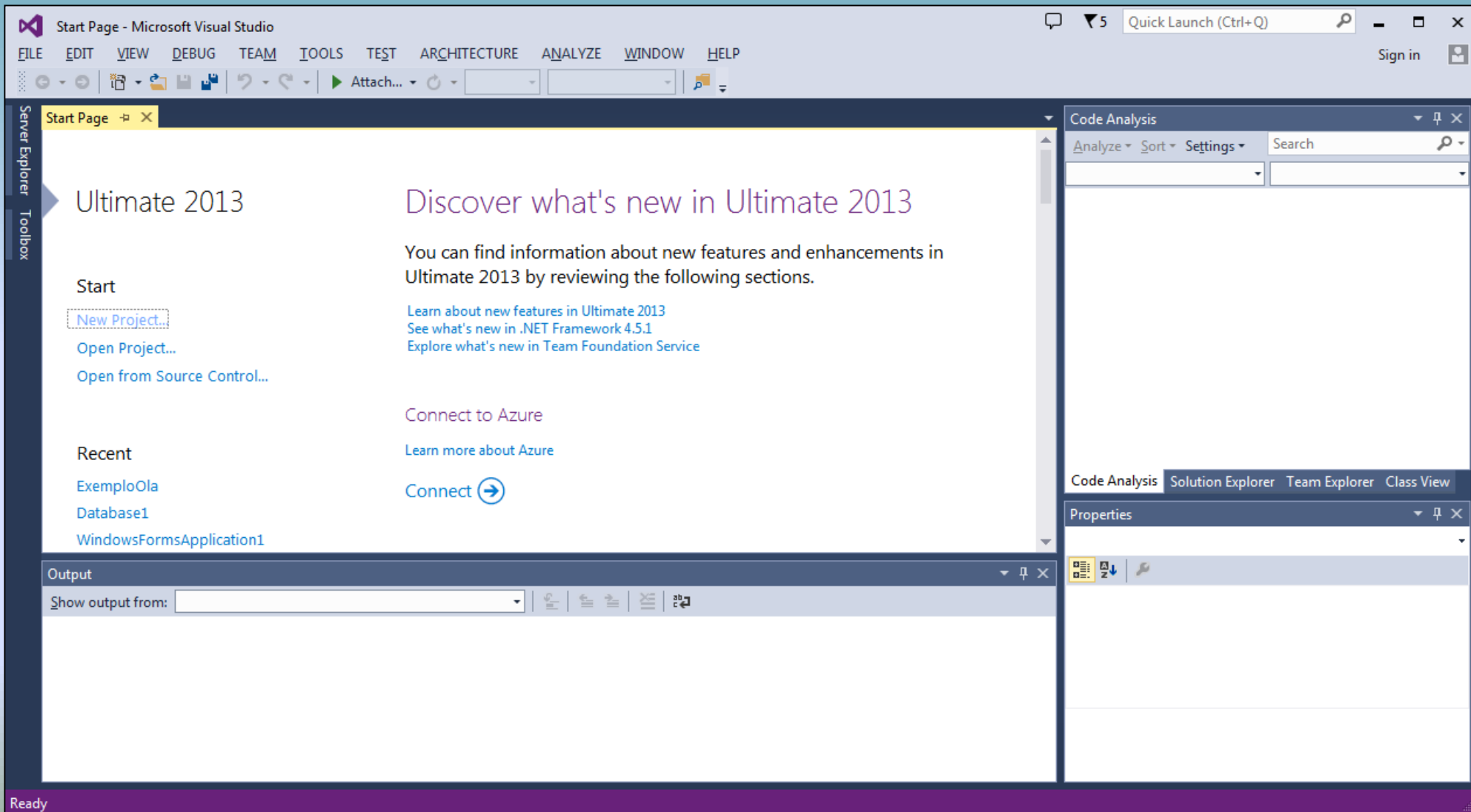
- Além da *class library* disponibilizada pelo .NET Framework, o desenvolvedor também pode criar suas próprias bibliotecas de classes.
- Projetos do tipo “*class library*” geram um arquivo DLL (*assembly*) quando são compilados.
- Estas DLLs podem ser instanciadas por várias aplicações, proporcionando reuso e padronização de código. Por exemplo: biblioteca para validação de CPF.

Visual Studio

- Microsoft Visual Studio é um ambiente integrado de desenvolvimento (*Integrated Development Environment – IDE*) que reúne ferramentas de apoio à construção de aplicações.
- A IDE permite construir projetos que combinam simultaneamente diversas linguagens de programação como C#, C++, Visual Basic e F#.

Visual Studio

- Tela Inicial



Visual Studio

- Soluções e Projetos
 - Para simplificar a gestão de arquivos e diretórios durante a fase de desenvolvimento das aplicações, o Visual Studio usa os conceitos de **solução** e **projeto**.
 - Um **projeto** é o conjunto de todos os itens envolvidos no desenvolvimento de uma aplicação, tais como códigos, arquivos de texto e imagem, ícones, conexões com bases de dados, etc. A extensão de um arquivo de projeto é **CSPROJ**.
 - Uma **solução** pode ser constituída por um único projeto ou por vários projetos relacionados. A extensão de um arquivo de solução é **SLN**.

Visual Studio

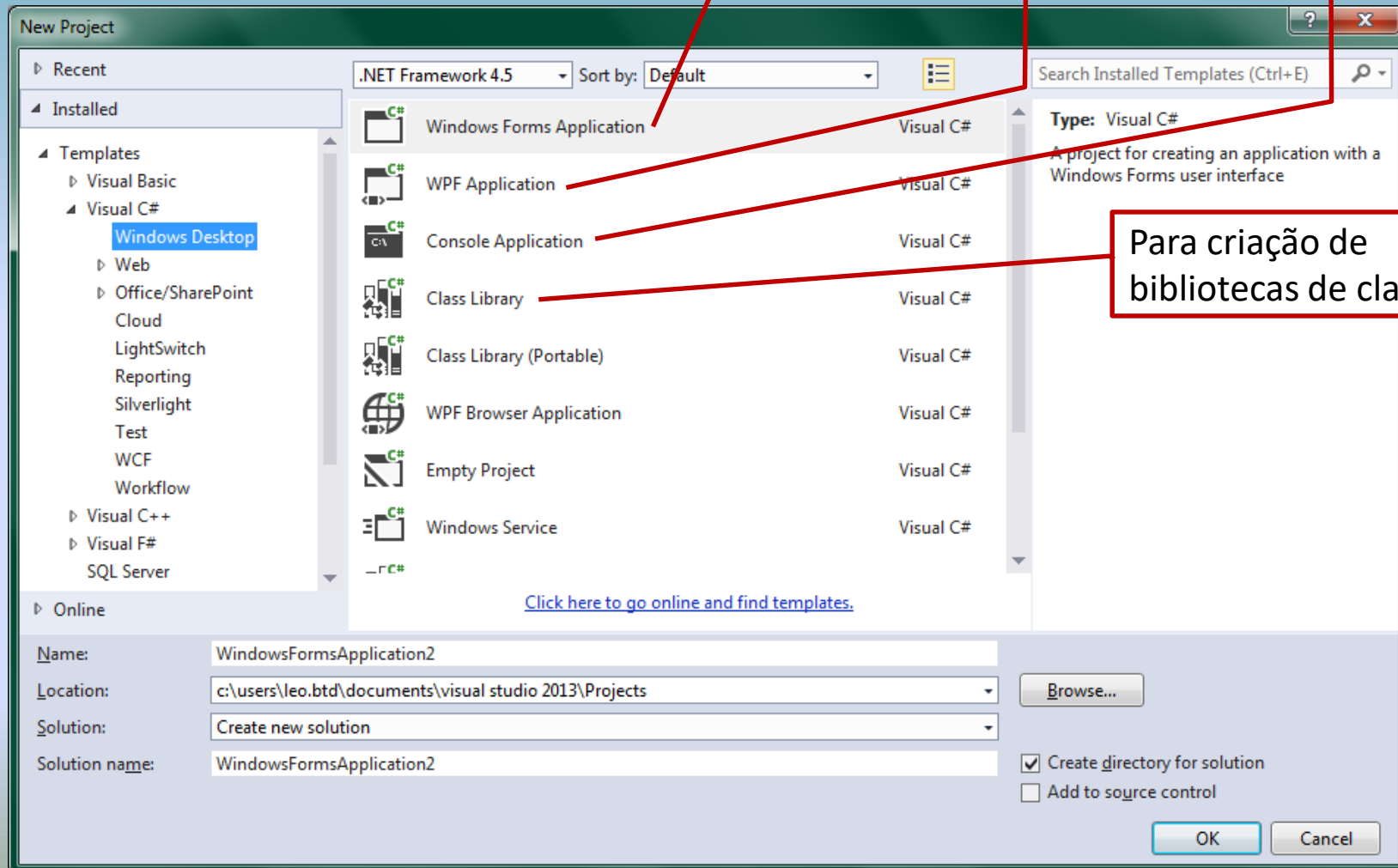
- Novo Projeto

Menu File >> New >> Project

Para aplicações desktop
(com aparência idêntica às
aplicações do Windows)

Para aplicações baseadas
em linhas de comando do
Windows (MS-DOS)

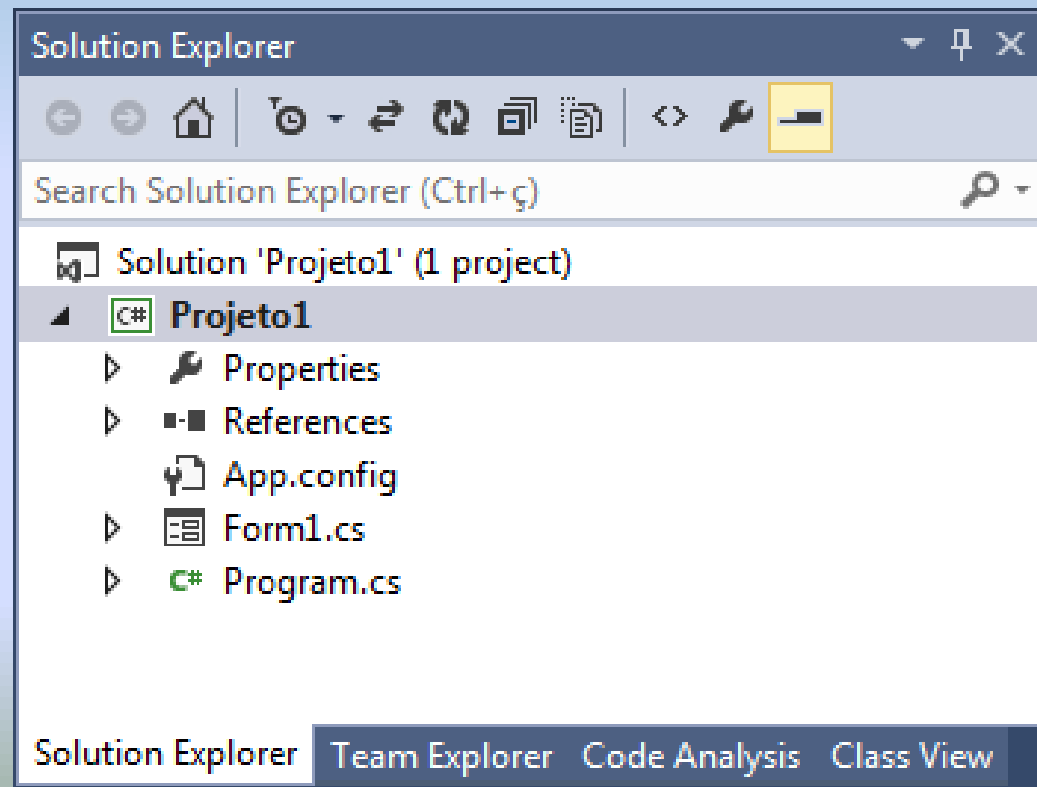
Para aplicações desktop
baseadas em Windows
Presentation Foundation



Para criação de
bibliotecas de classe

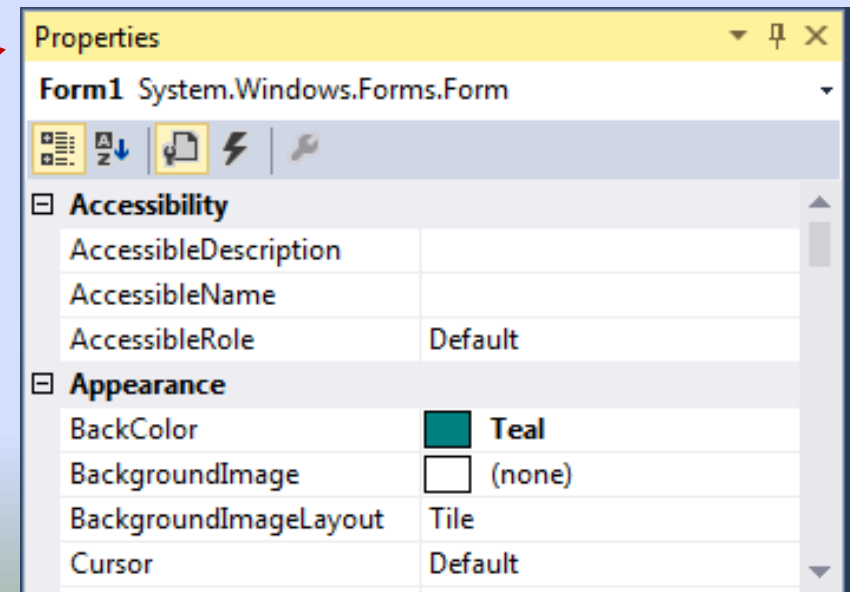
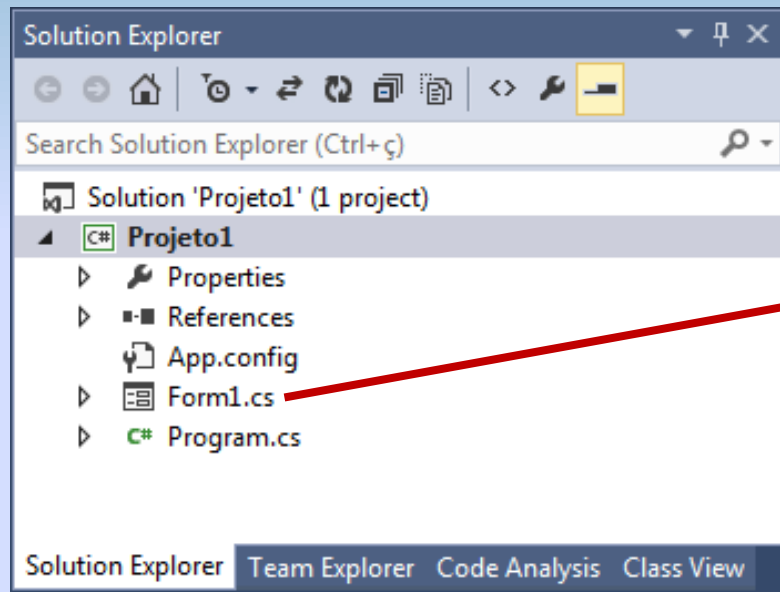
Visual Studio

- Solution Explorer
 - Área que apresenta todas as pastas e itens integrantes da solução e dos projetos abertos pelo usuário.



Visual Studio

- Solution Explorer
 - Cada item exibido no Solution Explorer possui um conjunto de propriedades.



Estrutura Básica de um Programa em C#

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

namespace Exemplo1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Namespaces usados
pelo projeto

Namespace
do projeto

Classe

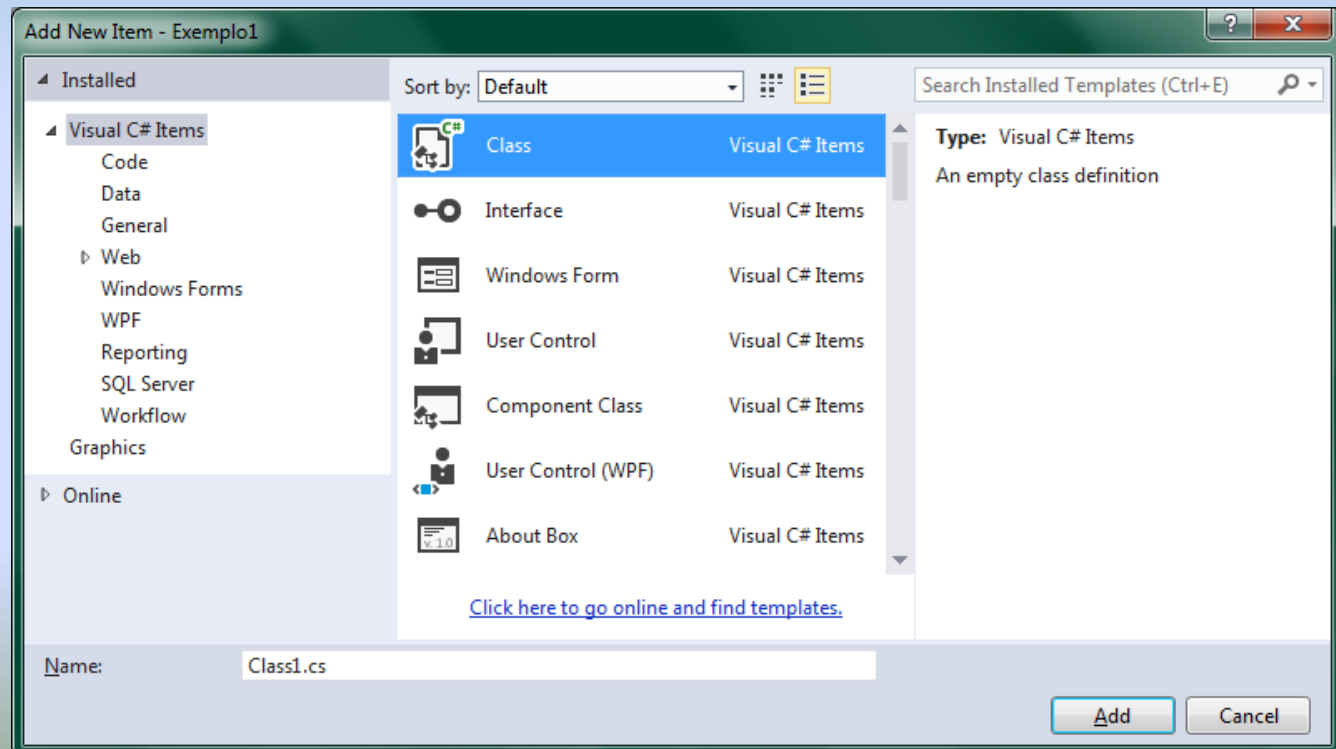
Método Main: Ponto de partida de execução de um programa.

static: Indica que o método pode ser chamado sem a necessidade de se criar uma instância (objeto) para sua classe.

void: Indica que o método não retorna nenhum valor.

Classes

- Os arquivos que implementam as classes possuem extensão CS (acrônimo de C Sharp).
- Para incluir uma nova classe ao projeto, basta acessar o menu **Project >> Add Class**, escolher a opção **Class**, nomear a nova classe e clicar em **Add**.



Declaração de Variáveis e Constantes

- Variáveis

- Uma variável é definida por um nome, um tipo de dado e um conteúdo.

`string` nome = “João da Silva”;

`int` idade = 28;

`bool` estrangeiro = `false`;

- Constantes

- São semelhantes às variáveis, porém seu conteúdo não pode ser alterado durante a execução do programa. Devem ser definidas por meio da palavra-chave **const**.

`const double` icms = 18;

Leitura e Apresentação de Dados

- Leitura de Dados
 - Em aplicações console, a leitura de dados do usuário é feita pelas funções **Read** e **ReadLine**.

Console.Read(); -> Lê apenas o 1º caractere e retorna um número (valor em ASC II) correspondente ao caractere lido.

Console.ReadLine(); -> Lê todos os caracteres e retorna uma string correspondente aos caracteres lidos.

```
int exp1 = Console.Read();  
string exp2 = Console.ReadLine();
```


Leitura e Apresentação de Dados

- Apresentação de Dados
 - Em aplicações console, a apresentação de dados do usuário é feita pelas funções **Write** e **WriteLine**.

Console.Write(); -> Imprime os dados mantendo o cursor do teclado na mesma linha.

Console.WriteLine(); -> Imprime os dados posicionando o cursor do teclado na próxima linha.

```
Console.Write("Testando o comando Write!");  
Console.WriteLine("Testando o comando WriteLine!");
```

Comentários

- São utilizados para documentar o código, sendo ignorados durante a compilação do programa.
- Para comentários de apenas uma linha, deve-se usar duas barras “//” no início do comentário.
- Para comentários de duas ou mais linhas, deve-se usar uma barra e um asterisco “/*” no início do comentário e um asterisco e uma barra no final do comentário “*/”.

```
//Declaração de variáveis
int exp1 = Console.Read();
string exp2 = Console.ReadLine();

/* O comando Write imprime os dados mantendo o cursor na mesma linha,
   e o comando WriteLine imprime os dados posicionando o cursor na próxima linha.*/
Console.Write("Testando o comando Write!");
Console.WriteLine("Testando o comando WriteLine!");
```

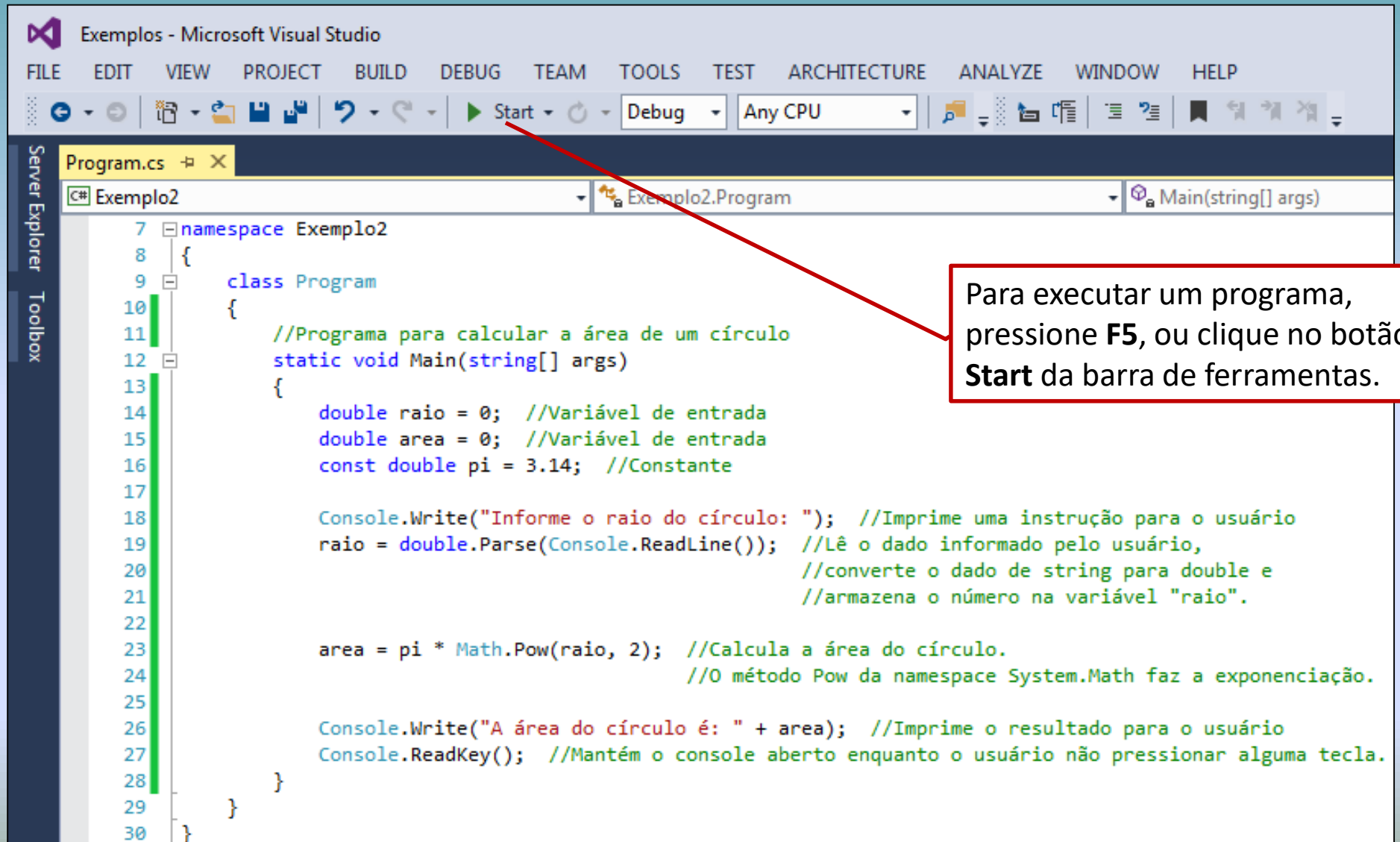
Regras de Nomenclatura

- Nomes de elementos como variáveis, constantes, classes, métodos e outros:
 - Devem iniciar sempre com uma letra de **a** à **z**.
 - Não devem ter qualquer tipo de acento.
 - Não devem conter caracteres especiais como ponto (.), vírgula (,), ponto e vírgula (;), hífen (-), @, \$, # e &.
 - Não devem conter espaços.
 - Devem ser únicos na classe em que estão declarados.

Convenções de Nomenclatura

- Por convenção:
 - Nomes de classes, métodos e propriedades iniciam com letra maiúscula.
 - Nomes de atributos, variáveis e constantes iniciam com letra minúscula.
 - Para separar palavras em classes, métodos, atributos, variáveis e constantes, usa-se as iniciais em maiúsculo a partir da 2ª palavra.
 - **Exemplos:** Depto**P**essoal (classe), Exportar**T**abela (método), total**A**dmitidos (variável) e taxa**I**mposto (constante).

Exemplo de Programa em C#



Exemplos - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP

Start Debug Any CPU

Program.cs

C# Exemplo2 Exemplo2.Program Main(string[] args)

```
7 namespace Exemplo2
8 {
9     class Program
10    {
11        //Programa para calcular a área de um círculo
12        static void Main(string[] args)
13        {
14            double raio = 0; //Variável de entrada
15            double area = 0; //Variável de entrada
16            const double pi = 3.14; //Constante
17
18            Console.WriteLine("Informe o raio do círculo: "); //Imprime uma instrução para o usuário
19            raio = double.Parse(Console.ReadLine()); //Lê o dado informado pelo usuário,
20                                                    //converte o dado de string para double e
21                                                    //armazena o número na variável "raio".
22
23            area = pi * Math.Pow(raio, 2); //Calcula a área do círculo.
24                                           //O método Pow da namespace System.Math faz a exponenciação.
25
26            Console.WriteLine("A área do círculo é: " + area); //Imprime o resultado para o usuário
27            Console.ReadKey(); //Mantém o console aberto enquanto o usuário não pressionar alguma tecla.
28        }
29    }
30 }
```

Para executar um programa, pressione F5, ou clique no botão **Start** da barra de ferramentas.

Referências

- Alfredo Lotar; ASP.NET com C# – Curso Prático. Novatec, 2003.
- Cláudio Vieira Oliveira, Ângela Lühmann e Benedito Petroni; Visual Studio C# – Fundamentos, Programação com ASP.NET, Windows Forms e Web Services. Editora Ciência Moderna, 2015.
- Felipe Cembranelli; ASP.NET – Guia do Desenvolvedor. Novatec, 2003.
- Henrique Loureiro; C# 6.0 com Visual Studio – Curso Completo. FCA, 2015.
- John Sharp; Microsoft Visual C# 2013: Passo a Passo. Bookman, 2014.
- Luiz Henrique Bueno e Sérgio Schaaf; Aplicações Web com Visual Studio .NET, ASP.NET & C#. Alta Books, 2002.
- <http://pt.stackoverflow.com/questions/17501/qual-é-a-diferença-de-api-biblioteca-e-framework>
- <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>
- https://en.wikipedia.org/wiki/.NET_Framework
- <http://tech.just4sharing.com/Pages/tech/NET-Framework-Overview.aspx>
- https://pt.wikipedia.org/wiki/Common_Language_Infrastructure
- [https://msdn.microsoft.com/pt-br/library/gg145045\(v=vs.110\).aspx](https://msdn.microsoft.com/pt-br/library/gg145045(v=vs.110).aspx)