

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Introdução ao C# e .NET

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Conteúdo

- C# e .NET
- Compilação e interpretação
- Estrutura de uma aplicação C# .NET
- Instalação do Visual Studio no Windows
- Primeiro programa em C# no Visual Studio
- Estrutura de um programa C#
- Dicas do Visual Studio

C# e .NET

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

C# e .NET

- C# : uma linguagem de programação (regras sintáticas)
- .NET (2002): uma plataforma de desenvolvimento para se criar diversos tipos de aplicações, podendo usar várias linguagens de programação
 - <https://www.microsoft.com/net/learn/what-is-dotnet>

Linguagens



.NET

- **BCL - Base Class Library**

[https://msdn.microsoft.com/en-us/library/gg145045\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/gg145045(v=vs.110).aspx)

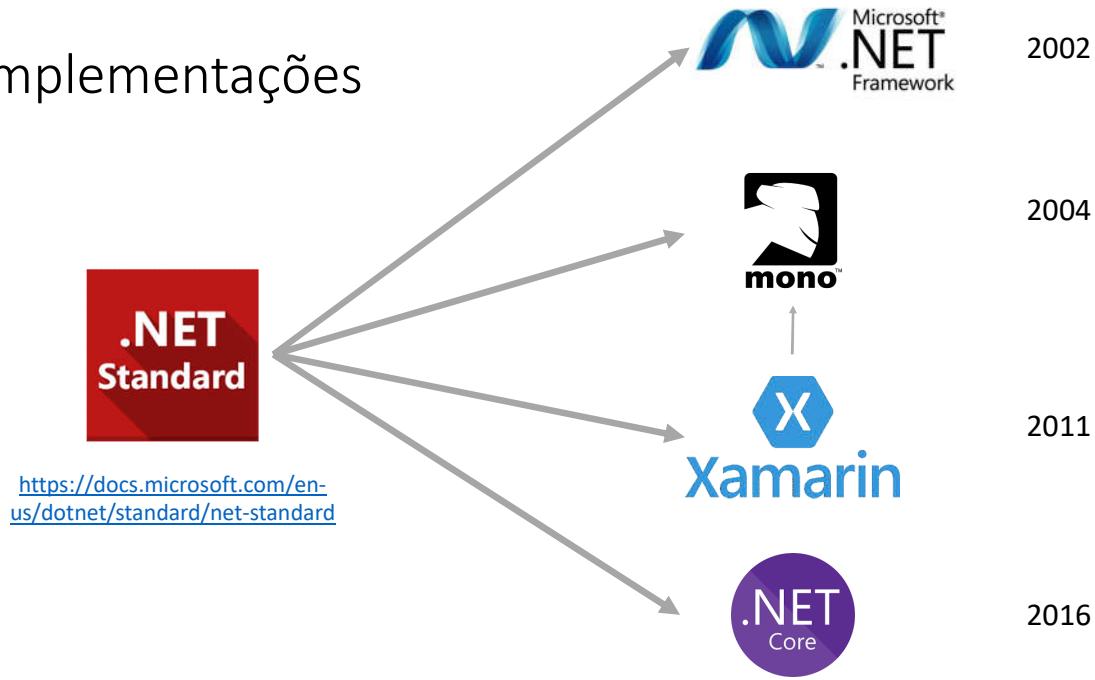
- **CLR - Common Language Runtime (Máquina Virtual)**

- Nota: possui garbage collection (objetos não utilizados são automaticamente desalocados da memória)

<https://www.microsoft.com/net/download>

<https://docs.microsoft.com/pt-br/dotnet/framework/migration-guide/how-to-determine-which-versions-are-installed>

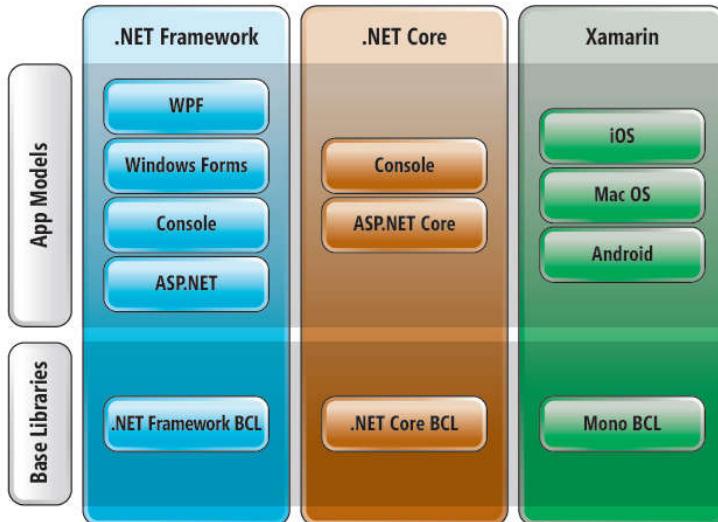
Implementações



	SO	Open Source	Propósito
.NET Standard	-	Sim	Especificação do .NET
.NET Framework	Windows	Não	Usado para criar aplicativos Windows desktop (console e gráfico) e aplicativos web ASP.NET rodando sobre o IIS
Mono	Vários	Sim	Usado para criar aplicativos e jogos multiplataforma
Xamarin	iOS Android Mac OS	Sim	Usado para criar aplicativos móveis híbridos para iOS e Android, e também para aplicações desktop para Mac OS
.NET Core	Windows Linux Mac OS	Sim	Usado para criar aplicativos modo console multiplataforma, e também para criar aplicativos web ASP.NET Core e serviços para nuvem

<https://msdn.microsoft.com/pt-br/magazine/mt842506.aspx>

<https://stackoverflow.com/questions/39649976/is-it-possible-to-make-desktop-gui-application-in-net-core>



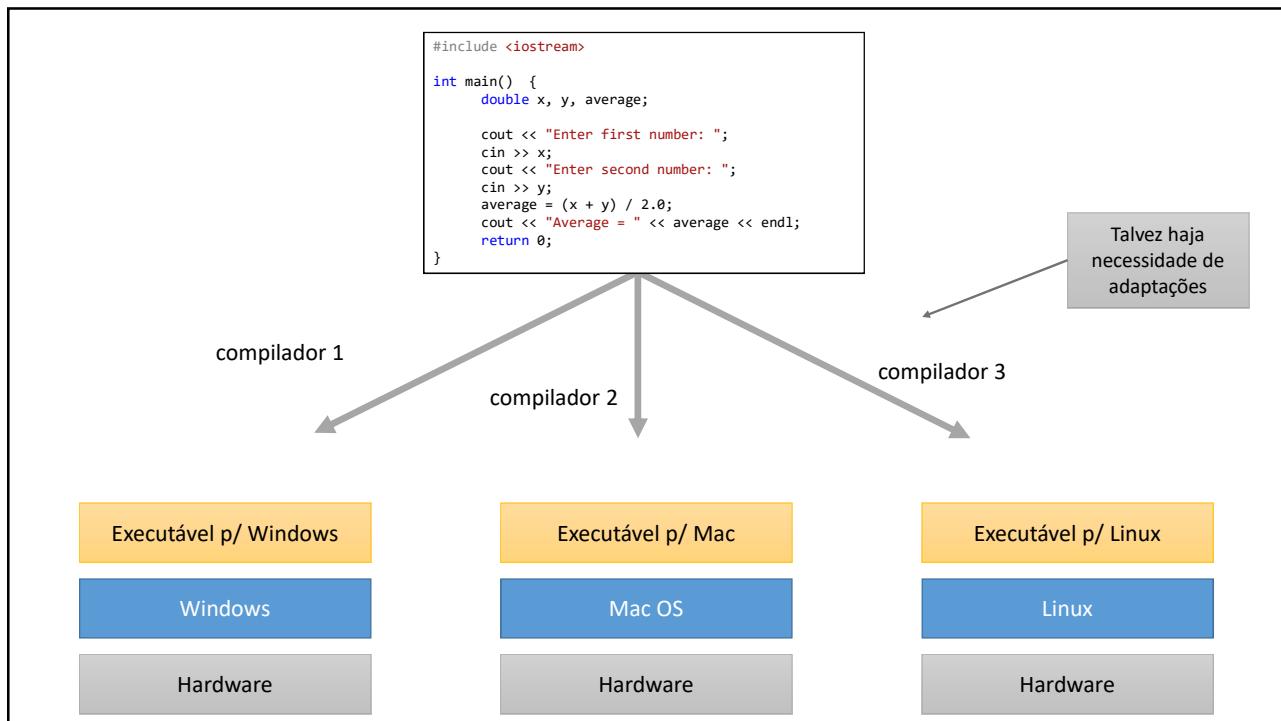
Compilação e interpretação

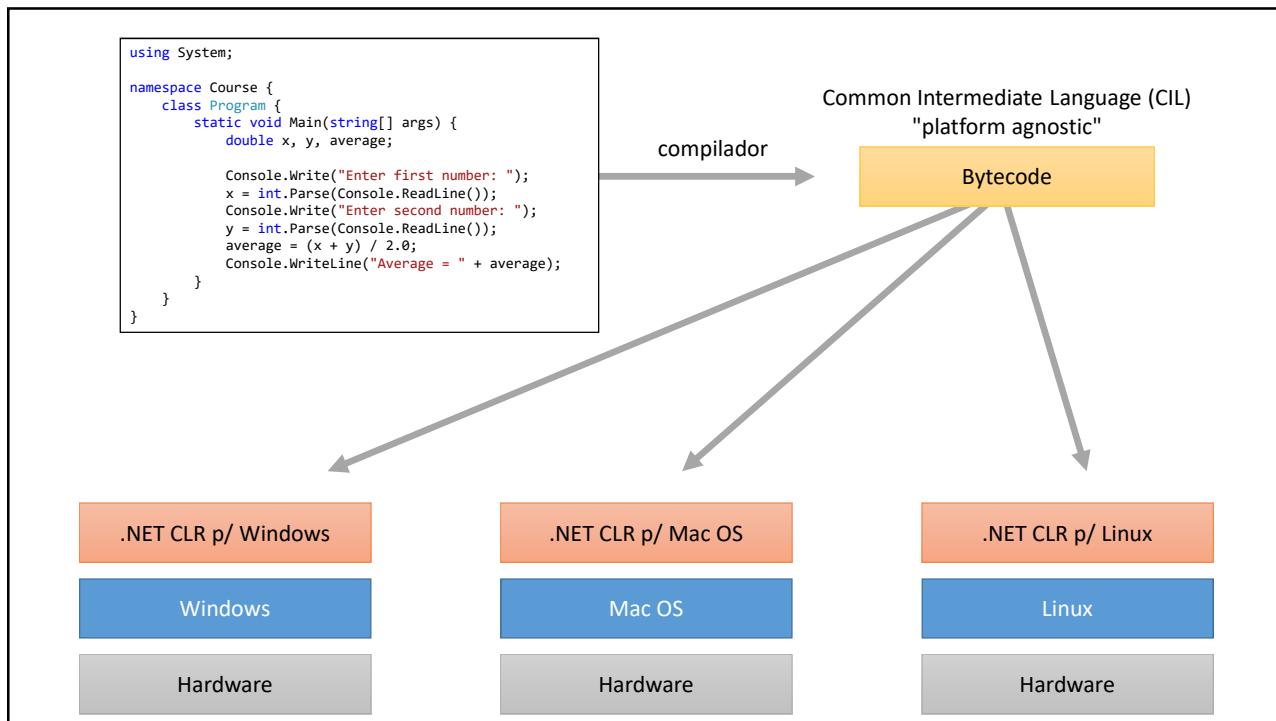
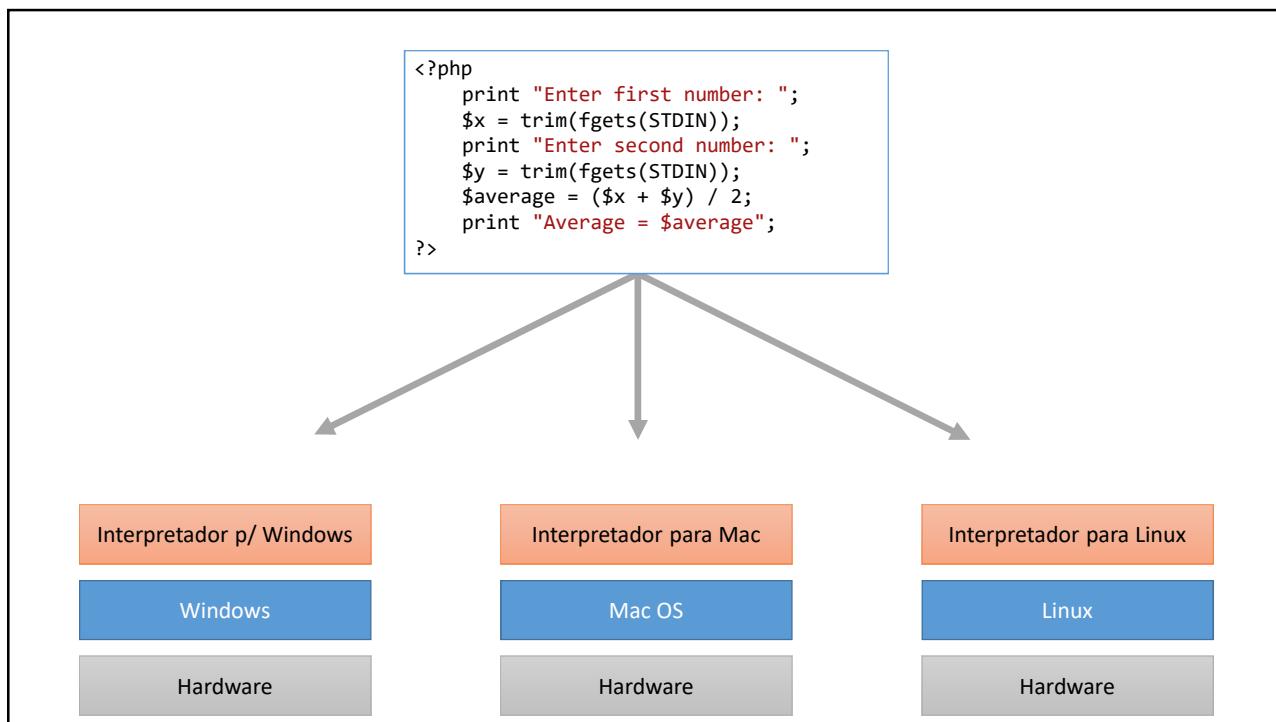
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Compilação e interpretação

- Linguagens **compiladas**: C, C++
- Linguagens **interpretadas**: PHP, JavaScript
- Linguagens **pré-compiladas + máquina virtual**: Java, C#





Modelo de execução

```
using System;
namespace Course {
    class Program {
        static void Main(string[] args) {
            Console.WriteLine("Hello World");
        }
    }
}
```

compilação
compilador →

Common Intermediate Language (CIL)

Compilação just-in-time (JIT)
Muito mais rápido que a interpretação

.NET
Common Language Runtime (CLR) - específica ao SO

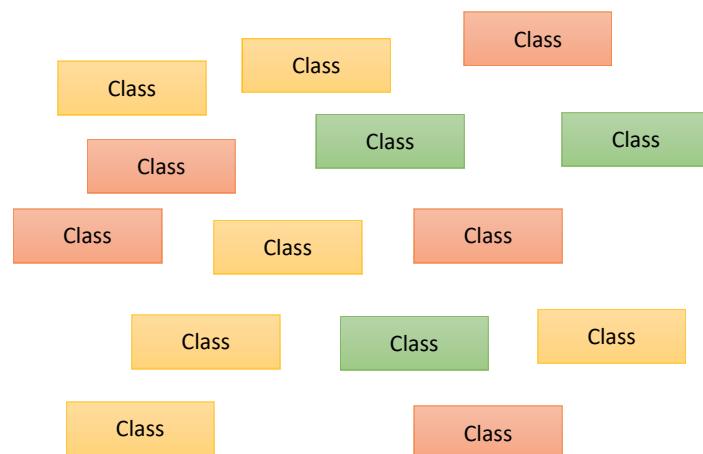
Código de máquina

Estrutura de uma aplicação C# .NET

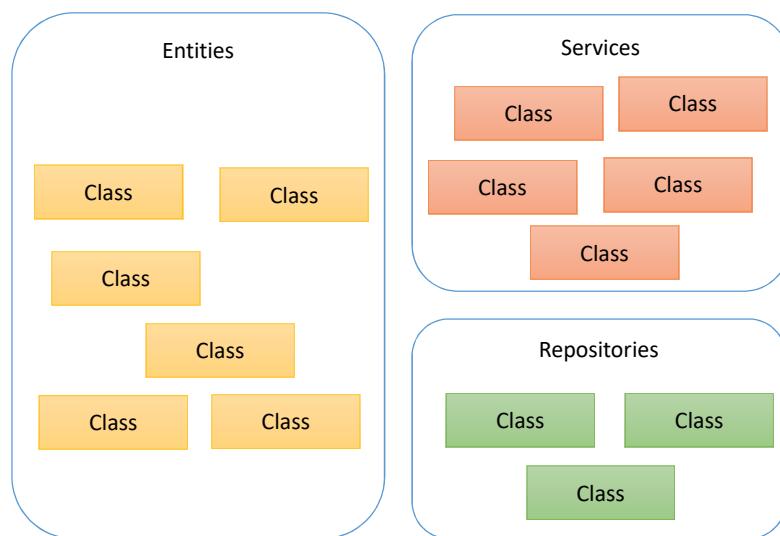
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

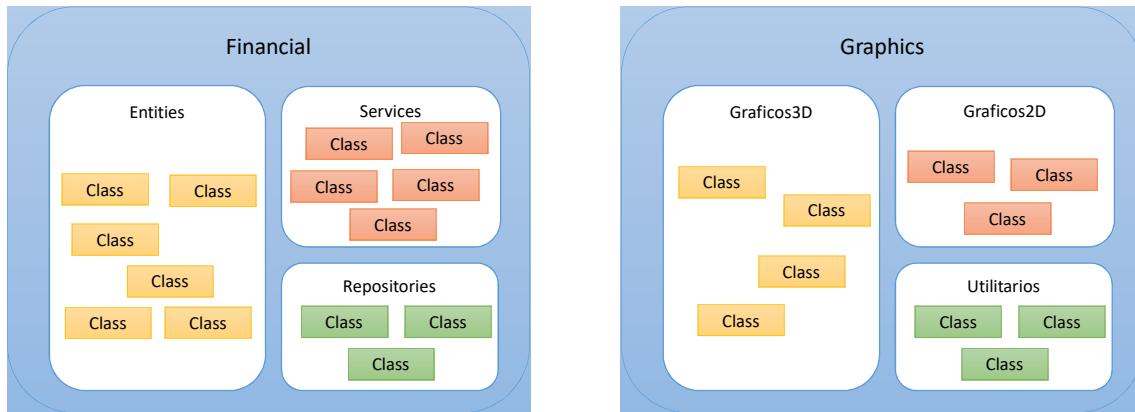
Uma aplicação é composta por classes



Namespace = agrupamento LÓGICO de classes relacionadas

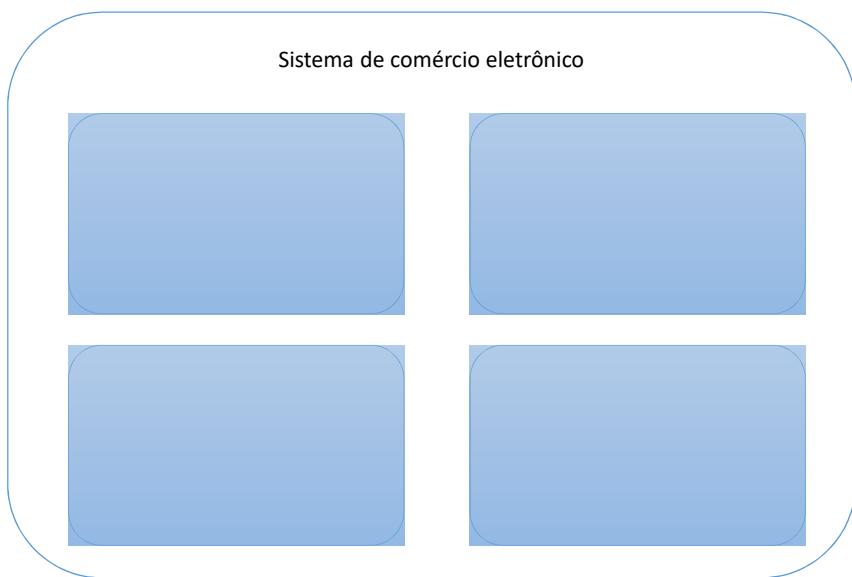


Assembly (DLL ou EXE) = Agrupamento FÍSICO de classes relacionadas (build)



Aplicação = Agrupamento de assemblies relacionados

Aplicação ~ Solution
Assembly ~ Project



Instalação do Visual Studio no Windows

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Certifique-se de que seu Windows esteja devidamente licenciado e atualizado
<https://www.visualstudio.com/pt-br/productinfo/vs2017-system-requirements-vs>
- Google: Visual Studio Community
<https://www.visualstudio.com/pt-br/vs/community/>
- Aba Windows -> Baixar o VS Community 2017
- Rodar o instalador
 - Cargas de Trabalho:
 - Desenvolvimento com a Plataforma Universal do Windows
 - Desenvolvimento para Desktop com .NET
 - ASP.NET e desenvolvimento Web
 - Desenvolvimento de multiplataforma com .NET Core

Primeiro programa em C# no Visual Studio

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Arquivo -> Novo -> Projeto -> Aplicativo de Console (.NET Core)
- File -> New -> Project -> Console App (.NET Core)
- Solution Explorer
 - Solution -> aplicação
 - Project -> assembly
- Executar o programa: CTRL + F5

Estrutura de um programa C#

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Estrutura de arquivos
 - Arquivo .sln
 - Arquivo .csproj
 - Classe principal .cs
 - Subpastas obj e bin
- Program.cs
 - Cláusulas using: referências a outros namespaces
 - Namespace
 - Classe
 - Método
 - static void Main(string[] args)

Dicas do Visual Studio

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Ferramentas -> Opções -> Ambiente -> Configurações Internacionais -> Idioma
 - Tools -> Options -> Environment -> International Settings -> Language
- Como fechar e reabrir o projeto?
 - Abra o arquivo .sln
- Indentação automática: CTRL + K + D
- Quebra de linha nas chaves: Tools -> Options -> Text Editor -> C# -> Code Estiling -> Formatting -> New Lines

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Recapitação de Lógica de Programação usando C#
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Construir programas básicos, aplicando o básico de Lógica de Programação, usando C# como linguagem

Objetivo:

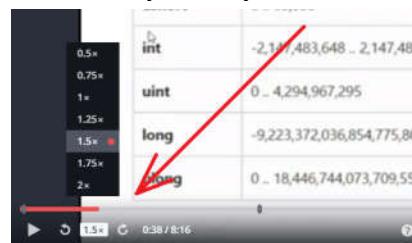
- Tipos de dados básicos em C#
- Estrutura sequencial (*entrada, processamento, saída*)
- Operadores (*aritméticos, comparativos, lógicos*)
- Estruturas de controle (*if-else, while, for*)

Dica para iniciantes



C# primeiros passos: Lógica de Programação e Algoritmos

Dica para experientes



int	-2,147,483,648 .. 2,147,483
uint	0 .. 4,294,967,295
long	-9,223,372,036,854,775,8
ulong	0 .. 18,446,744,073,709,55

Tipos básicos em C#

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/built-in-types-table>

- C# é uma linguagem **estaticamente tipada**
- Tipos valor pré-definidos em C#
- Tipos referência pré-definidos em C#
- Variável não atribuída
- Overflow
- Padrão para float: sufixo "f"
- Padrão para char: aspas simples
- Padrão para string: aspas duplas
- Padrão para bool: true, false
- Opção: inferência de tipos com palavra "var" (dentro de métodos)

C# built-in data types (tipos valor)

C# Type	.Net Framework Type	Signed	Bytes	Possible Values
sbyte	System.Sbyte	Yes	1	-128 to 127
short	System.Int16	Yes	2	-32768 to 32767
int	System.Int32	Yes	4	-2 ³¹ to 2 ³¹ - 1
long	System.Int64	Yes	8	-2 ⁶³ to 2 ⁶³ - 1
byte	System.Byte	No	1	0 to 255
ushort	System.UInt16	No	2	0 to 65535
uint	System.UInt32	No	4	0 to 2 ³² - 1
ulong	System.UInt64	No	8	0 to 2 ⁶⁴ - 1
float	System.Single	Yes	4	±1.5 x 10 ⁻⁴⁵ to ±3.4 x 10 ³⁸ with 7 significant figures
double	System.Double	Yes	8	±5.0 x 10 ⁻³²⁴ to ±1.7 x 10 ³⁰⁸ with 15 or 16 significant figures
decimal	System.Decimal	Yes	12	±1.0 x 10 ⁻²⁸ to ±7.9 x 10 ²⁸ with 28 or 29 significant figures
char	System.Char	N/A	2	Any Unicode character
bool	System.Boolean	N/A	1/2	true or false

C# built-in data types (tipos referência)

Tipo C#	Tipo .NET	Descrição
string	System.String	Uma cadeia de caracteres Unicode IMUTÁVEL (<i>segurança, simplicidade, thread safe</i>)
object	System.Object	Um objeto genérico (toda classe em C# é subclasse de object) GetType Equals GetHashCode ToString

Demo

```
bool completo = false;
char genero = 'F';
char letra = '\u0041';
byte n1 = 126;
int n2 = 1000;
int n3 = 2147483647;
long n4 = 2147483648L;
float n5 = 4.5f;
double n6 = 4.5;
String nome = "Maria Green";
Object obj1 = "Alex Brown";
Object obj2 = 4.5f;

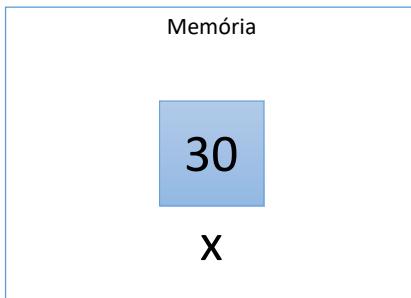
Console.WriteLine(completo);
Console.WriteLine(genero);
Console.WriteLine(letra);
Console.WriteLine(n1);
Console.WriteLine(n2);
Console.WriteLine(n3);
Console.WriteLine(n4);
Console.WriteLine(n5);
Console.WriteLine(n6);
Console.WriteLine(nome);
Console.WriteLine(obj1);
Console.WriteLine(obj2);
```

Funções para valores mínimos e máximos

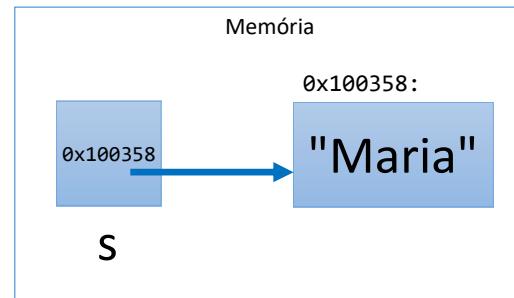
- int.MinValue
- int.MaxValue
- sbyte.MaxValue
- long.MaxValue
- decimal.MaxValue
- etc...

Tipo valor vs. tipo referência

```
int x = 30;
```



```
string s = "Maria";
```



Restrições e convenções para nomes

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Restrições para nomes de variáveis

- Não pode começar com dígito: use uma letra ou _
- Não usar acentos ou til
- Não pode ter espaço em branco
- Sugestão: use nomes que tenham um significado

Errado:

```
int 5minutos;  
int salário;  
int salario do funcionario;
```

Correto:

```
int _5minutos;  
int salario;  
int salarioDoFuncionario;
```

Convenções

- Camel Case: lastName (parâmetros de métodos, variáveis dentro de métodos)
- Pascal Case: LastName (namespaces, classe, properties e métodos)
- Padrão _lastName (atributos "internos" da classe)

```
namespace Curso
{
    class ContaBancaria
    {
        public string Titular { get; set; }
        private double _saldo;

        public void Deposito(double quantia)
        {
            _saldo += quantia;
        }

        public double GetSaldo()
        {
            return _saldo;
        }
    }
}
```

Conversão implícita e casting

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Conversão implícita entre tipos
<https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/keywords/implicit-numeric-conversions-table>
- Casting: conversão explícita entre tipos COMPATÍVEIS

Exemplo 1

```
double a;  
float b;  
  
a = 5.1;  
b = (float)a;  
  
Console.WriteLine(b);
```

Exemplo 2

```
double a;  
int b;  
  
a = 5.1;  
b = (int)a;  
  
Console.WriteLine(b);
```

Exemplo 3

```
int a = 5;  
int b = 2;  
  
double resultado = (double) a / b;  
  
Console.WriteLine(resultado);
```

Saída de dados em C#

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Imprimir na saída padrão (console)

- Comandos
 - `Console.WriteLine(valor);`
 - `Console.Write(valor);`

Demo

```
using System;
using System.Globalization;

namespace PrimeiroProjeto {
    class Program {
        static void Main(string[] args) {

            char genero = 'F';
            int idade = 32;
            double saldo = 10.35784;
            String nome = "Maria";

            Console.WriteLine("Bom dia!");
            Console.WriteLine("Boa tarde!");
            Console.WriteLine("Boa noite!");
            Console.WriteLine("-----");
            Console.WriteLine(genero);
            Console.WriteLine(idade);
            Console.WriteLine(saldo);
            Console.WriteLine(nome);
            Console.WriteLine(saldo.ToString("F2"));
            Console.WriteLine(saldo.ToString("F4"));
            Console.WriteLine(saldo.ToString("F4", CultureInfo.InvariantCulture));
        }
    }
}
```

Placeholders, concatenação e interpolação

Demo:

```
int idade = 32;
double saldo = 10.35784;
String nome = "Maria";

Console.WriteLine("{0} tem {1} anos e tem saldo igual a {2:F2} reais", nome, idade, saldo);

Console.WriteLine($"{nome} tem {idade} anos e tem saldo igual a {saldo:F2} reais");

Console.WriteLine(nome + " tem " + idade + " anos e tem saldo igual a "
+ saldo.ToString("F2", CultureInfo.InvariantCulture) + " reais");
```

Exercício de fixação

Em um novo programa, inicie as seguintes variáveis:

```
string produto1 = "Computador";
string produto2 = "Mesa de escritório";

byte idade = 30;
int codigo = 5290;
char genero = 'M';

double preco1 = 2100.0;
double preco2 = 650.50;
double medida = 53.234567;
```

Em seguida, usando os valores das variáveis, produza a seguinte saída na tela do console:

Produtos:
Computador, cujo preço é \$ 2100,00
Mesa de escritório, cujo preço é \$ 650,50

Registro: 30 anos de idade, código 5290 e gênero: M

Medida com oito casas decimais: 53,23456700
Arredondado (três casas decimais): 53,235
Separador decimal invariant culture: 53.235

(correção na próxima página)

```
using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {
            string produto1 = "Computador";
            string produto2 = "Mesa de escritório";

            byte idade = 30;
            int codigo = 5290;
            char genero = 'M';

            double preco1 = 2100.0;
            double preco2 = 650.50;
            double medida = 53.234567;

            Console.WriteLine("Produtos:");
            Console.WriteLine("{0}, cujo preço é $ {1:F2}", produto1, preco1);
            Console.WriteLine("{0}, cujo preço é $ {1:F2}", produto2, preco2);
            Console.WriteLine();
            Console.WriteLine("Registro: {0} anos de idade, código {1} e gênero: {2}", idade, codigo, genero);
            Console.WriteLine();
            Console.WriteLine("Medida com oito casas decimais: {0:F8}", medida);
            Console.WriteLine("Arredondado (três casas decimais): {0:F3}", medida);
            Console.WriteLine("Separador decimal invariant culture: " + medida.ToString("F3", CultureInfo.InvariantCulture));
        }
    }
}
```

Operadores aritméticos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Operadores aritméticos

Operador	Significado
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão

NOTAS:

1) * / % tem precedência maior que + -

2) Exemplos:

$3 + 4 * 2 \longrightarrow$ Resultado: 11

$(3 + 4) * 2 \longrightarrow$ Resultado: 14

3) Pode-se usar parêntesis à vontade

4) Exemplo com mod:

$17 \% 3 \longrightarrow$ Resultado: 2

$$\begin{array}{r} 17 \\ \hline 3 \\ 2 \end{array}$$

Demo

```
int n1 = 3 + 4 * 2;
int n2 = (3 + 4) * 2;
int n3 = 17 % 3;
double n4 = 10.0 / 8.0;

double a = 1.0, b = -3.0, c = -4.0;

double delta = Math.Pow(b, 2.0) - 4.0 * a * c;

double x1 = (-b + Math.Sqrt(delta)) / (2.0 * a);
double x2 = (-b - Math.Sqrt(delta)) / (2.0 * a);

Console.WriteLine(n1);
Console.WriteLine(n2);
Console.WriteLine(n3);
Console.WriteLine(n4);
Console.WriteLine(delta);
Console.WriteLine(x1);
Console.WriteLine(x2);
```

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Operadores de atribuição

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Operadores de atribuição

Operador	Exemplo	Significado
=	a = 10;	a RECEBE 10
+=	a += 2;	a RECEBE a + 2;
-=	a -= 2;	a RECEBE a - 2;
*=	a *= 2;	a RECEBE a * 2;
/=	a /= 2;	a RECEBE a / 2;
%=	a %= 3;	a RECEBE a % 3;

Demo

```
int a = 10;
Console.WriteLine(a);

a += 2;
Console.WriteLine(a);

a *= 3;
Console.WriteLine(a);

string s = "ABC";
Console.WriteLine(s);

s += "DEF";
Console.WriteLine(s);
```

Operadores aritméticos / atribuição

Operador	Exemplo	Significado
++	a++; ou ++a;	a = a + 1;
--	a--; ou --a;	a = a - 1;

```
int a = 10;  
a++;  
Console.WriteLine(a);  
  
SAÍDA:  
11
```

```
int a = 10;  
int b = a++;  
Console.WriteLine(a);  
Console.WriteLine(b);  
  
SAÍDA:  
11  
10
```

```
int a = 10;  
int b = ++a;  
Console.WriteLine(a);  
Console.WriteLine(b);  
  
SAÍDA:  
11  
11
```

Entrada de dados em C# - Parte 1

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Comando

```
Console.ReadLine();
```

- Lê da entrada padrão **até a quebra de linha**.
- Retorna os dados lidos **na forma de string**.

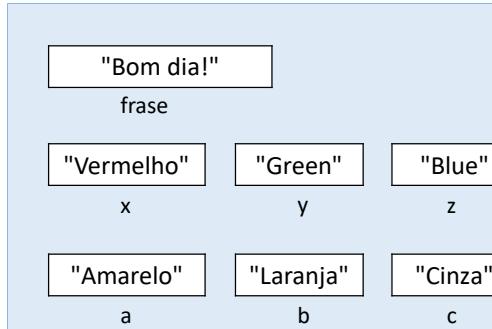
Checklist

- Ler um texto até a quebra de linha e armazenar em uma variável
- Ler três palavras, uma em cada linha, armazenando cada uma em uma variável
- Ler três palavras na mesma linha, separadas por espaço, armazenando cada uma em uma variável

Console:

```
Bom dia!  
Vermelho  
Verde  
Azul  
Amarelo Laranja Cinza
```

Memória:



Split

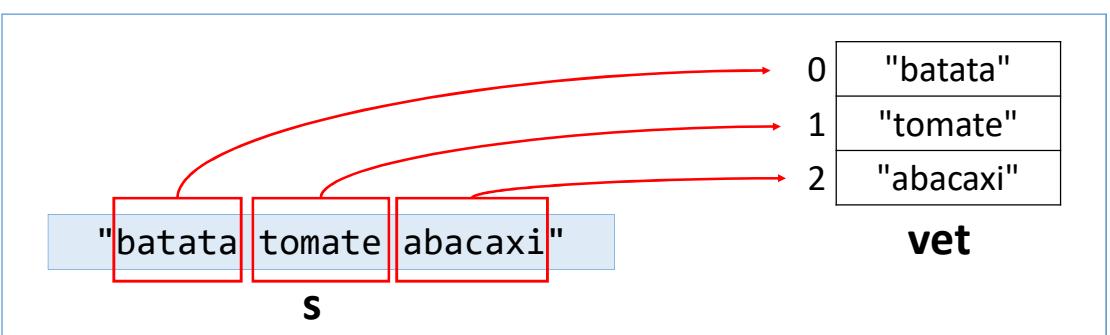
```
string s = Console.ReadLine();

string[] vet = s.Split(' ');
string p1 = vet[0];
string p2 = vet[1];
string p3 = vet[2];
```

Console:

```
batata tomate abacaxi
```

"batata"	"tomate"	"abacaxi"
p1	p2	p3



Demo

```
string frase = Console.ReadLine();
string x = Console.ReadLine();
string y = Console.ReadLine();
string z = Console.ReadLine();

string[] v = Console.ReadLine().Split(' ');
string a = v[0];
string b = v[1];
string c = v[2];

Console.WriteLine("Você digitou: ");
Console.WriteLine(frase);
Console.WriteLine(x);
Console.WriteLine(y);
Console.WriteLine(z);
Console.WriteLine(a);
Console.WriteLine(b);
Console.WriteLine(c);
```

Entrada de dados em C# - Parte 2

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

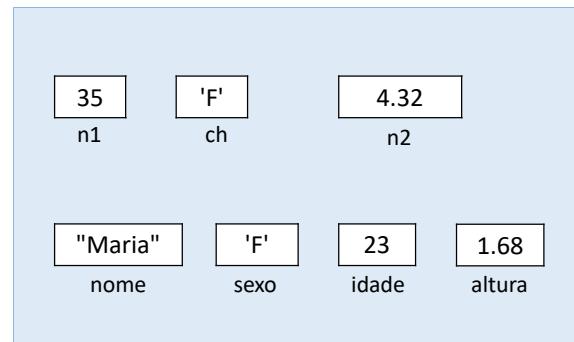
Checklist

- Ler um número inteiro
- Ler um caractere
- Ler um número double
- Ler um nome (única palavra), sexo (caractere F ou M), idade (inteiro) e altura (double) na mesma linha, armazenando-os em quatro variáveis com os devidos tipos

Console:

```
35
F
4.32
Maria F 23 1.68
```

Memória:



Demo

```
int n1 = int.Parse(Console.ReadLine());
char ch = char.Parse(Console.ReadLine());
double n2 = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

string[] vet = Console.ReadLine().Split(' ');
string nome = vet[0];
char sexo = char.Parse(vet[1]);
int idade = int.Parse(vet[2]);
double altura = double.Parse(vet[3], CultureInfo.InvariantCulture);

Console.WriteLine("Você digitou:");
Console.WriteLine(n1);
Console.WriteLine(ch);
Console.WriteLine(n2.ToString("F2", CultureInfo.InvariantCulture));
Console.WriteLine(nome);
Console.WriteLine(sexo);
Console.WriteLine(idade);
Console.WriteLine(altura.ToString("F2", CultureInfo.InvariantCulture));
```

Exercício de fixação

Fazer um programa para executar a seguinte interação com o usuário, lendo os valores destacados em vermelho, e depois mostrar os dados na tela:

Entre com seu nome completo:

Alex Green

Quantos quartos tem na sua casa?

3

Entre com o preço de um produto:

500.50

Entre seu último nome, idade e altura (mesma linha):

Green 21 1.73

SAÍDA ESPERADA (NÚMEROS REAIS COM DUAS CASAS DECIMAIS):

Alex Green

3

500.50

Green

21

1.73

(correção na próxima página)

```
using System;
using System.Globalization;
namespace Course {
    class Program {
        static void Main(string[] args) {

            Console.WriteLine("Entre com seu nome completo:");
            string fullName = Console.ReadLine();
            Console.WriteLine("Quantos quartos tem na sua casa?");
            int bedrooms = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter product price:");
            double price = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            Console.WriteLine("Entre seu último nome, idade e altura (mesma linha):");
            string[] vect = Console.ReadLine().Split(' ');
            string lastName = vect[0];
            int age = int.Parse(vect[1]);
            double height = double.Parse(vect[2], CultureInfo.InvariantCulture);

            Console.WriteLine(fullName);
            Console.WriteLine(bedrooms);
            Console.WriteLine(price.ToString("F2", CultureInfo.InvariantCulture));
            Console.WriteLine(lastName);
            Console.WriteLine(age);
            Console.WriteLine(height.ToString("F2", CultureInfo.InvariantCulture));
        }
    }
}
```

Exercícios propostos - PARTE 1

Estrutura sequencial (entrada, processamento, saída)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Operadores comparativos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Operadores comparativos

Operador	Significado
>	maior
<	menor
\geq	maior ou igual
\leq	menor ou igual
$=$	igual
\neq	diferente

Demo

```
int a = 10;
bool c1 = a < 10;
bool c2 = a < 20;
bool c3 = a > 10;
bool c4 = a > 5;
Console.WriteLine(c1);
Console.WriteLine(c2);
Console.WriteLine(c3);
Console.WriteLine(c4);
Console.WriteLine("-----");

bool c5 = a <= 10;
bool c6 = a >= 10;
bool c7 = a == 10;
bool c8 = a != 10;
Console.WriteLine(c5);
Console.WriteLine(c6);
Console.WriteLine(c7);
Console.WriteLine(c8);
```

Operadores lógicos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Operadores lógicos

Operador	Significado
&&	E
	OU
!	NÃO

C1	C2	C1 E C2
F	F	F
F	V	F
V	F	F
V	V	V

C1	C2	C1 OU C2
F	F	F
F	V	V
V	F	V
V	V	V

NOTAS:

- 1) Precedência: ! > && > ||
- 2) Pode-se usar parêntesis à vontade
- 3) Exemplos:
2 > 3 || 4 != 5 → Resultado: true
!(2>3) && 4 != 5 → Resultado: true

Demo

```
bool c1 = 2 > 3 || 4 != 5; // true
bool c2 = !(2 > 3) && 4 != 5; // true
Console.WriteLine(c1);
Console.WriteLine(c2);

Console.WriteLine("-----");

bool c3 = 10 < 5; // false
bool c4 = c1 || c2 && c3; // true
Console.WriteLine(c3);
Console.WriteLine(c4);
```

Estrutura condicional (if-else)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura condicional

Simples

```
if ( condição ) {  
    comando 1  
    comando 2  
}
```

Composta

```
if ( condição ) {  
    comando 1  
    comando 2  
}  
else {  
    comando 3  
    comando 4  
}
```

Encadeamentos

```
if ( condição 1 ) {  
    comando 1  
    comando 2  
}  
else if ( condição 2 ) {  
    comando 3  
    comando 4  
}  
else if ( condição 3 ) {  
    comando 5  
    comando 6  
}  
else {  
    comando 7  
    comando 8  
}
```

Nota: se o bloco de comandos possuir apenas um comando, as chaves são opcionais.

Demo #1

Entre com um número inteiro:

10

Par!

Entre com um número ímpar?

15

Ímpar!

```
using System;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Console.WriteLine("Entre com um número inteiro:");
            int x = int.Parse(Console.ReadLine());

            if (x % 2 == 0) {
                Console.WriteLine("Par!");
            }
            else {
                Console.WriteLine("Ímpar");
            }
        }
    }
}
```

Demo #2

Qual a hora atual?
10
Bom dia!

hora < 12

Qual a hora atual?
14
Boa tarde!

12 <= hora < 18

Qual a hora atual?
19
Boa noite!

hora >= 18

```
using System;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Console.WriteLine("Qual a hora atual?");
            int hora = int.Parse(Console.ReadLine());

            if (hora < 12) {
                Console.WriteLine("Bom dia!");
            }
            else if (hora < 18) {
                Console.WriteLine("Boa tarde!");
            }
            else {
                Console.WriteLine("Boa noite!");
            }
        }
    }
}
```

Escopo e inicialização

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Escopo de uma variável: é a região do programa onde a variável é válida, ou seja, onde ela pode ser referenciada.
- Uma variável não pode ser usada se não for iniciada.
- Falaremos de escopo de métodos no Capítulo "Comportamento de memória, arrays, listas"

Demo

```
double preco = double.Parse(Console.ReadLine());  
  
if (preco > 100.0) {  
    double desconto = preco * 0.1;  
}  
  
Console.WriteLine(desconto);
```

Exercícios propostos - PARTE 2 Estrutura condicional (if-else)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Funções (sintaxe)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Funções

- Representam um processamento que possui um significado
 - Math.Sqrt(double)
 - Console.WriteLine(string)
- Principais vantagens: modularização, delegação e reaproveitamento
- Dados de entrada e saída
 - Funções podem receber dados de entrada (parâmetros ou argumentos)
 - Funções podem ou não retornar uma saída
- Em orientação a objetos, funções em classes recebem o nome de "métodos"

Problema exemplo

Fazer um programa para ler três números inteiros e mostrar na tela o maior deles.

Exemplo:

```
Digite três números:  
5  
8  
3  
Maior = 8
```

```
using System;  
  
namespace Course {  
    class Program {  
        static void Main(string[] args) {  
  
            Console.WriteLine("Digite três números:");  
            int n1 = int.Parse(Console.ReadLine());  
            int n2 = int.Parse(Console.ReadLine());  
            int n3 = int.Parse(Console.ReadLine());  
  
            double resultado = Maior(n1, n2, n3);  
  
            Console.WriteLine("Maior = " + resultado);  
        }  
  
        static int Maior(int a, int b, int c) {  
            int m;  
            if (a > b && a > c) {  
                m = a;  
            }  
            else if (b > c) {  
                m = b;  
            }  
            else {  
                m = c;  
            }  
            return m;  
        }  
    }  
}
```

Debugging com Visual Studio

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Tópicos

- Teclas
 - F9 - marcar/desmarcar breakpoint
 - F5 - iniciar/continuar o debug
 - F10 - executar um passo (pula função)
 - F11 - executar um passo (entra na função)
 - SHIFT+F11 - sair do método em execução
 - SHIFT+F5 - parar debug
- Janelas
 - Watch (expressões personalizadas)
 - Autos (expressões "interessantes" detectadas pelo Visual Studio)
 - Locals (variáveis locais)

Estrutura repetitiva while

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura "enquanto"

```
while ( condição ) {  
    comando 1  
    comando 2  
}
```

Regra:

V: executa e volta
F: pula fora

Problema exemplo

Digitar um número e mostrar sua raiz quadrada com três casas decimais, depois repetir o procedimento. Quando o usuário digitar um número negativo (podendo inclusive ser na primeira vez), mostrar uma mensagem "Número negativo" e terminar o programa.

```
Digite um número: 25
5.000
Digite outro número: 10
3.162
Digite outro número : 9
3.000
Digite outro número : -4
Número negativo!
```

```
using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Console.Write("Digite um número: ");
            double x = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

            while (x >= 0.0) {
                double raiz = Math.Sqrt(x);
                Console.WriteLine(raiz.ToString("F3", CultureInfo.InvariantCulture));
                Console.Write("Digite outro número: ");
                x = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            }

            Console.WriteLine("Número negativo!");
        }
    }
}
```

Exercícios propostos - PARTE 3

Estrutura while

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura repetitiva for

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura "para"

Executa somente na primeira vez V: executa e volta
F: pula fora Executa toda vez depois de voltar

```
for ( início ; condição ; incremento ) {  
    comando 1  
    comando 2  
}
```

Demo

Digitar um número N e depois N valores inteiros. Mostrar a soma dos N valores digitados.

```
Quantos números inteiros você vai digitar? 3  
Valor #1: 10  
Valor #2: 7  
Valor #3: 8  
Soma = 25
```

```
using System;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Console.WriteLine("Quantos números inteiros você vai digitar? ");
            int N = int.Parse(Console.ReadLine());

            int soma = 0;
            for (int i = 1; i <= N; i++) {
                Console.Write("Valor #{0}: ", i);
                int valor = int.Parse(Console.ReadLine());
                soma += valor;
            }

            Console.WriteLine("Soma = " + soma);
        }
    }
}
```

Exercícios propostos - PARTE 4 Estrutura for

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Classes, atributos, métodos, membros estáticos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Resolvendo um problema sem orientação a objetos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema exemplo

Fazer um programa para ler as medidas dos lados de dois triângulos X e Y (suponha medidas válidas). Em seguida, mostrar o valor das áreas dos dois triângulos e dizer qual dos dois triângulos possui a maior área.

A fórmula para calcular a área de um triângulo a partir das medidas de seus lados a, b e c é a seguinte (fórmula de Heron):

$$area = \sqrt{p(p-a)(p-b)(p-c)} \quad \text{onde} \quad p = \frac{a+b+c}{2}$$

Exemplo:

Entre com as medidas do triângulo X:

3.00
4.00
5.00

Entre com as medidas do triângulo Y:

7.50
4.50
4.02

Área de X = 6.0000

Área de Y = 7.5638

Maior área: Y

```
using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {

            double xA, xB, xC, yA, yB, yC;

            Console.WriteLine("Entre com as medidas do triângulo X:");
            xA = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            xB = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            xC = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

            Console.WriteLine("Entre com as medidas do triângulo Y:");
            yA = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            yB = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            yC = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

            double p = (xA + xB + xC) / 2.0;
            double areaX = Math.Sqrt(p * (p - xA) * (p - xB) * (p - xC));

            p = (yA + yB + yC) / 2.0;
            double areaY = Math.Sqrt(p * (p - yA) * (p - yB) * (p - yC));

            Console.WriteLine("Área de X = " + areaX.ToString("F4", CultureInfo.InvariantCulture));
            Console.WriteLine("Área de Y = " + areaY.ToString("F4", CultureInfo.InvariantCulture));

            if (areaX > areaY) {
                Console.WriteLine("Maior área: X");
            }
            else {
                Console.WriteLine("Maior área: Y");
            }
        }
    }
}
```

Criando uma classe com três atributos para representar melhor o triângulo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Discussão

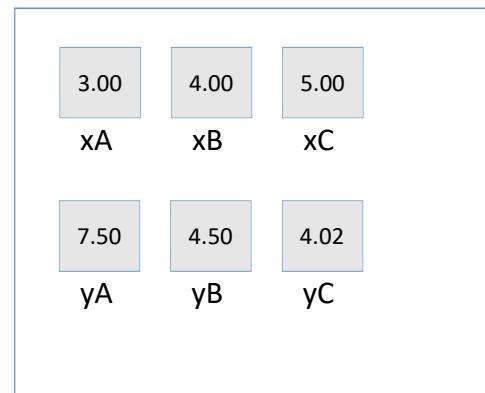
Triângulo é uma entidade com três atributos: a, b, c.

Estamos usando três variáveis distintas para representar cada triângulo:

double xA, xB, xC, yA, yB, yC;

Para melhorar isso, vamos usar uma CLASSE para representar um triângulo.

Memória:

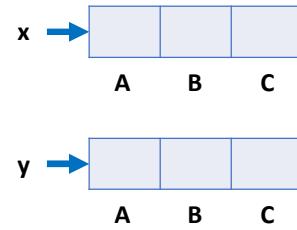
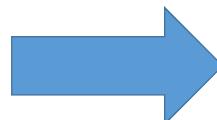


Classe

- É um tipo estruturado que pode conter (membros):
 - Atributos (dados / campos)
 - Métodos (funções / operações)
- A classe também pode prover muitos outros recursos, tais como:
 - Construtores
 - Sobrecarga
 - Encapsulamento
 - Herança
 - Polimorfismo
- Exemplos:
 - Entidades: Produto, Cliente, Triangulo
 - Serviços: ProdutoService, ClienteService, EmailService, StorageService
 - Controladores: ProdutoController, ClienteController
 - Utilitários: Calculadora, Compactador
 - Outros (views, repositórios, gerenciadores, etc.)

```
namespace Course {  
    class Triangulo {  
  
        public double A;  
        public double B;  
        public double C;  
    }  
  
    double xA, xB, xC, yA, yB, yC;  
  
      
      
      
  
      
      
    
```

```
Triangulo x, y;  
x = new Triangulo();  
y = new Triangulo();
```

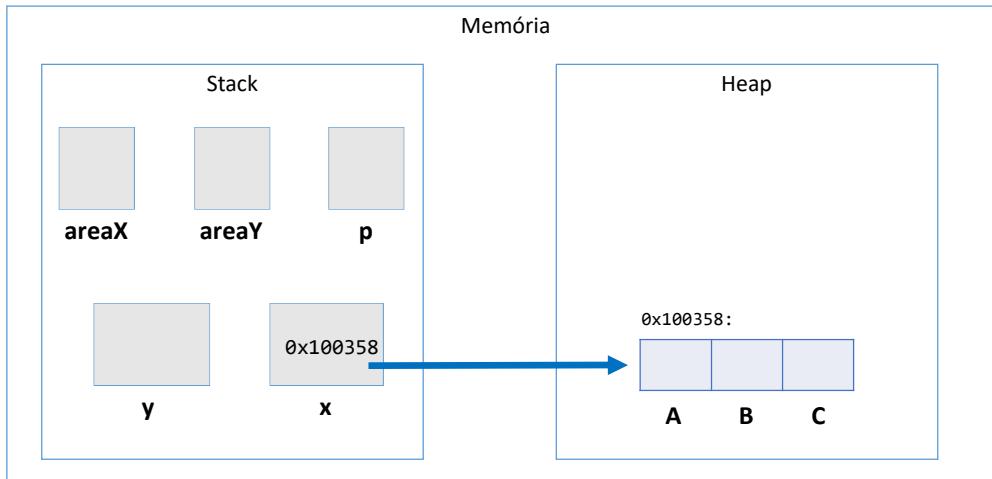


Instanciação

(alocação dinâmica de memória)

```
double areaX, areaY, p;  
Triangulo x, y;
```

```
x = new Triangulo();
```



```
using System;  
using System.Globalization;  
  
namespace Course {  
    class Program {  
        static void Main(string[] args) {  
  
            Triangulo x, y;  
  
            x = new Triangulo();  
            y = new Triangulo();  
  
            Console.WriteLine("Entre com as medidas do triângulo X:");  
            x.A = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);  
            x.B = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);  
            x.C = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);  
  
            Console.WriteLine("Entre com as medidas do triângulo Y:");  
            y.A = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);  
            y.B = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);  
            y.C = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);  
  
            double p = (x.A + x.B + x.C) / 2.0;  
            double areaX = Math.Sqrt(p * (p - x.A) * (p - x.B) * (p - x.C));  
  
            p = (y.A + y.B + y.C) / 2.0;  
            double areaY = Math.Sqrt(p * (p - y.A) * (p - y.B) * (p - y.C));  
  
            Console.WriteLine("Área de X = " + areaX.ToString("F4", CultureInfo.InvariantCulture));  
            Console.WriteLine("Área de Y = " + areaY.ToString("F4", CultureInfo.InvariantCulture));  
  
            if (areaX > areaY) {  
                Console.WriteLine("Maior área: X");  
            }  
            else {  
                Console.WriteLine("Maior área: Y");  
            }  
        }  
    }  
}
```

Classes, objetos, atributos

- Classe: é a definição do tipo

```
namespace Course {  
    class Triangulo {  
  
        public double A;  
        public double B;  
        public double C;  
    }  
}
```

- Objetos: são instâncias da classe



Primeiros exercícios (classes, objetos e atributos)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Favor fazer os exercícios do arquivo "**primeiros-exercícios.pdf**"
deste capítulo.

Criando um método para
obtermos os benefícios de
reaproveitamento e delegação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

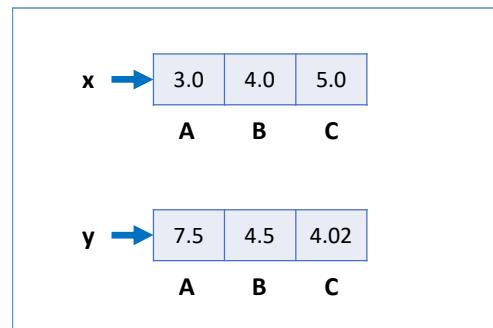
Discussão

Com o uso de CLASSE, agora nós temos uma variável composta do tipo "Triangulo" para representar cada triângulo:

```
Triangulo x, y;  
x = new Triangulo();  
y = new Triangulo();
```

Agora vamos melhorar nossa CLASSE, acrescentando nela um MÉTODO para calcular a área.

Memória:



```

using System;

namespace Course {
    class Triangulo {

        public double A;
        public double B;
        public double C;

        public double Area() {
            double p = (A + B + C) / 2.0;
            return Math.Sqrt(p * (p - A) * (p - B) * (p - C));
        }
    }
}

```

```

using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Triangulo x, y;

            x = new Triangulo();
            y = new Triangulo();

            Console.WriteLine("Entre com as medidas do triângulo X:");
            x.A = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            x.B = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            x.C = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

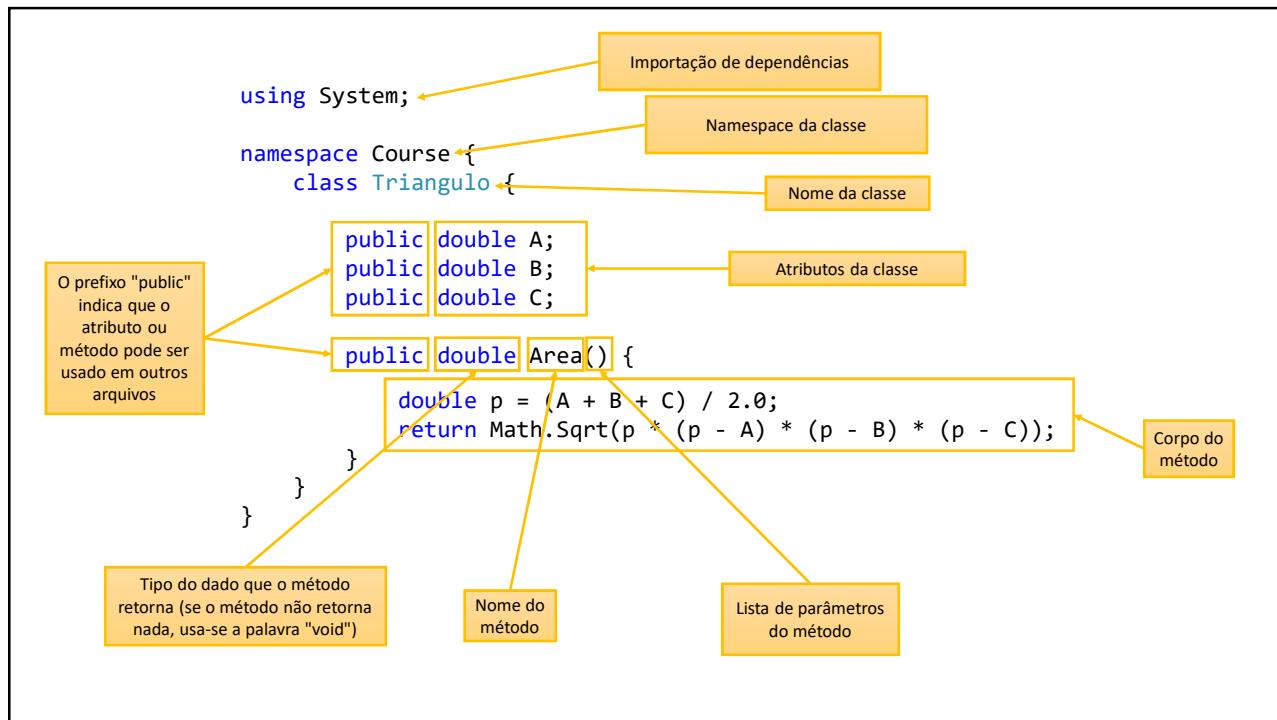
            Console.WriteLine("Entre com as medidas do triângulo Y:");
            y.A = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            y.B = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            y.C = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

            double areaX = x.Area();
            double areaY = y.Area();

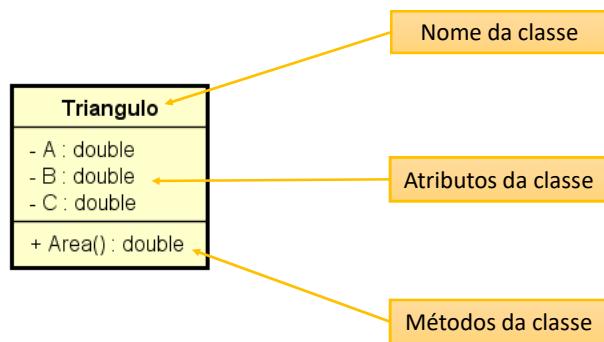
            Console.WriteLine("Área de X = " + areaX.ToString("F4", CultureInfo.InvariantCulture));
            Console.WriteLine("Área de Y = " + areaY.ToString("F4", CultureInfo.InvariantCulture));

            if (areaX > areaY) {
                Console.WriteLine("Maior área: X");
            }
            else {
                Console.WriteLine("Maior área: Y");
            }
        }
    }
}

```



Projeto da classe (UML)



Discussão

Quais são os benefícios de se calcular a área de um triângulo por meio de um MÉTODO dentro da CLASSE Triangulo?

- 1) Reaproveitamento de código:** nós eliminamos o código repetido (cálculo das áreas dos triângulos x e y) no programa principal.
- 2) Delegação de responsabilidades:** quem deve ser responsável por saber como calcular a área de um triângulo é o próprio triângulo. A lógica do cálculo da área não deve estar em outro lugar.

Começando a resolver um segundo problema exemplo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema:

Fazer um programa para ler os dados de um produto em estoque (nome, preço e quantidade no estoque). Em seguida:

- Mostrar os dados do produto (nome, preço, quantidade no estoque, valor total no estoque)
- Realizar uma entrada no estoque e mostrar novamente os dados do produto
- Realizar uma saída no estoque e mostrar novamente os dados do produto

Para resolver este problema, você deve criar uma CLASSE conforme projeto ao lado:

(veja exemplo na próxima página)

Produto
- Nome : string
- Preco : double
- Quantidade : int
+ ValorTotalEmEstoque() : double
+ AdicionarProdutos(quantidade : int) : void
+ RemoverProdutos(quantidade : int) : void

Exemplo:

Entre os dados do produto:

Nome: **TV**

Preço: **900.00**

Quantidade no estoque: **10**

Dados do produto: TV, \$ 900.00, 10 unidades, Total: \$ 9000.00

Digite o número de produtos a ser adicionado ao estoque: **5**

Dados atualizados: TV, \$ 900.00, 15 unidades, Total: \$ 13500.00

Digite o número de produtos a ser removido do estoque: **3**

Dados atualizados: TV, \$ 900.00, 12 unidades, Total: \$ 10800.00

Produto
- Nome : string
- Preco : double
- Quantidade : int
+ ValorTotalEmEstoque() : double
+ AdicionarProdutos(quantidade : int) : void
+ RemoverProdutos(quantidade : int) : void

Object e ToString

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Discussão

- Toda classe em C# é uma subclasse da classe Object
- Object possui os seguintes métodos:
 - GetType - retorna o tipo do objeto
 - Equals - compara se o objeto é igual a outro
 - GetHashCode - retorna um código hash do objeto
 - ToString - converte o objeto para string
- Demo

```
using System.Globalization;

namespace Course {
    class Produto {

        public string Nome;
        public double Preco;
        public int Quantidade;

        public double ValorTotalEmEstoque() {
            return Preco * Quantidade;
        }

        public override string ToString() {
            return Nome
                + ", $ "
                + Preco.ToString("F2", CultureInfo.InvariantCulture)
                + ","
                + Quantidade
                + " unidades, Total: $ "
                + ValorTotalEmEstoque().ToString("F2", CultureInfo.InvariantCulture);
        }
    }
}
```

Finalizando o programa

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```

using System.Globalization;

namespace Course {
    class Produto {

        public string Nome;
        public double Preco;
        public int Quantidade;

        public double ValorTotalEmEstoque() {
            return Preco * Quantidade;
        }

        public void AdicionarProdutos(int quantidade) {
            Quantidade += quantidade;
        }

        public void RemoverProdutos(int quantidade) {
            Quantidade -= quantidade;
        }

        public override string ToString() {
            return Nome
                + ", $ "
                + Preco.ToString("F2", CultureInfo.InvariantCulture)
                + " "
                + Quantidade
                + " unidades, Total: $ "
                + ValorTotalEmEstoque().ToString("F2", CultureInfo.InvariantCulture);
        }
    }
}

```

```

using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Produto p = new Produto();

            Console.WriteLine("Entre os dados do produto:");
            Console.Write("Nome: ");
            p.Nome = Console.ReadLine();
            Console.Write("Preço: ");
            p.Preco = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            Console.Write("Quantidade no estoque: ");
            p.Quantidade = int.Parse(Console.ReadLine());

            Console.WriteLine();
            Console.WriteLine("Dados do produto: " + p);

            Console.WriteLine();
            Console.Write("Digite o número de produtos a ser adicionado ao estoque: ");
            int qte = int.Parse(Console.ReadLine());
            p.AdicionarProdutos(qte);

            Console.WriteLine();
            Console.WriteLine("Dados atualizados: " + p);

            Console.WriteLine();
            Console.Write("Digite o número de produtos a ser removido do estoque: ");
            qte = int.Parse(Console.ReadLine());
            p.RemoverProdutos(qte);

            Console.WriteLine();
            Console.WriteLine("Dados atualizados: " + p);
        }
    }
}

```

Exercícios de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Exercício 1

Fazer um programa para ler os valores da largura e altura de um retângulo. Em seguida, mostrar na tela o valor de sua área, perímetro e diagonal. Usar uma classe como mostrado no projeto ao lado.

Retangulo
- Largura : double
- Altura : double

+ Área() : double
+ Perímetro() : double
+ Diagonal() : double

Exemplo:

```
Entre a largura e altura do retângulo:  
3.00  
4.00  
AREA = 12.00  
PERÍMETRO = 14.00  
DIAGONAL = 5.00
```

Correção: arquivo **correcao-exercicios-fixacao.pdf**

Exercício 2

Fazer um programa para ler os dados de um funcionário (nome, salário bruto e imposto). Em seguida, mostrar os dados do funcionário (nome e salário líquido). Em seguida, aumentar o salário do funcionário com base em uma porcentagem dada (somente o salário bruto é afetado pela porcentagem) e mostrar novamente os dados do funcionário. Use a classe projetada abaixo.

Exemplo:

Nome: **Joao Silva**
Salário bruto: **6000.00**
Imposto: **1000.00**

Funcionário: Joao Silva, \$ 5000.00

Digite a porcentagem para aumentar o salário: **10.0**

Dados atualizados: Joao Silva, \$ 5600.00

Funcionario

```
- Nome : string  
- SalarioBruto : double  
- Imposto : double  
  
+ SalarioLiquido() : double  
+ AumentarSalario(porcentagem : double) : void
```

Correção: arquivo **correcao-exercicios-fixacao.pdf**

Exercício 3

Fazer um programa para ler o nome de um aluno e as três notas que ele obteve nos três trimestres do ano (primeiro trimestre vale 30 e o segundo e terceiro valem 35 cada). Ao final, mostrar qual a nota final do aluno no ano. Dizer também se o aluno está APROVADO ou REPROVADO e, em caso negativo, quantos pontos faltam para o aluno obter o mínimo para ser aprovado (que é 60 pontos). Você deve criar uma classe **Aluno** para resolver este problema.

Exemplo 1:

Nome do aluno: **Alex Green**
Digite as três notas do aluno:
27.00
31.00
32.00
NOTA FINAL = **90.00**
APROVADO

Exemplo 2:

Nome do aluno: **Alex Green**
Digite as três notas do aluno:
17.00
20.00
15.00
NOTA FINAL = **52.00**
REPROVADO
FALTARAM **8.00 PONTOS**

Correção: arquivo **correcao-exercicios-fixacao.pdf**

Membros estáticos - PARTE 1

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Produto
- Nome : string
- Preco : double
- Quantidade : int
+ ValorTotalEmEstoque() : double
+ AdicionarProdutos(quantity : int) : void
+ RemoverProdutos(quantity : int) : void

membros
= atributos e métodos

Membros estáticos

- Também chamados membros de classe
 - Em oposição a membros e instâncias
- São membros que fazem sentido independentemente de objetos. Não precisam de objeto para serem chamados. São chamados a partir do próprio nome da classe.
- Aplicações comuns:
 - Classes utilitárias  **Math.Sqrt(double)**
 - Declaração de constantes
- Uma classe que possui somente membros estáticos, pode ser uma classe estática também. Esta classe não poderá ser instanciada.

```
Triangulo x, y;  
x = new Triangulo();  
y = new Triangulo();
```



x.Area() → 6.0



y.Area() → 7.5638

Problema exemplo

Fazer um programa para ler um valor numérico qualquer, e daí mostrar quanto seria o valor de uma circunferência e do volume de uma esfera para um raio daquele valor. Informar também o valor de PI com duas casas decimais.

Exemplo:

```
Entre o valor do raio: 3.0
Circunferência: 18.84
Volume: 113.04
Valor de PI: 3.14
```

Checklist

- Versão 1: métodos na própria classe do programa
 - Nota: dentro de um método estático você não pode chamar membros de instância da mesma classe.
- Versão 2: classe Calculadora com membros de instância
- Versão 3: classe Calculadora com método estático

```

using System;
using System.Globalization;

namespace Course {
    class Program {

        static double Pi = 3.14;

        static void Main(string[] args) {
            Console.Write("Entre o valor do raio: ");
            double raio = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

            double circ = Circunferencia(raio);
            double volume = Volume(raio);

            Console.WriteLine("Circunferência: " + circ.ToString("F2", CultureInfo.InvariantCulture));
            Console.WriteLine("Volume: " + volume.ToString("F2", CultureInfo.InvariantCulture));
            Console.WriteLine("Valor de PI: " + Pi.ToString("F2", CultureInfo.InvariantCulture));
        }

        static double Circunferencia(double r) {
            return 2.0 * Pi * r;
        }

        static double Volume(double r) {
            return 4.0 / 3.0 * Pi * r * r * r;
        }
    }
}

```

VERSAO 1

Membros estáticos - PARTE 2

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```
namespace Course {
    class Calculadora {

        public double Pi = 3.14;

        public double Circunferencia(double r) {
            return 2.0 * Pi * r;
        }

        public double Volume(double r) {
            return 4.0 / 3.0 * Pi * r * r * r;
        }
    }
}
```

VERSÃO 2

```
Calculadora calc = new Calculadora();

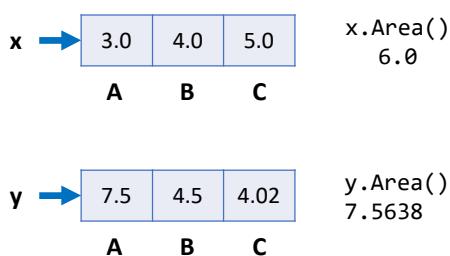
Console.Write("Entre o valor do raio: ");
double raio = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

double circ = calc.Circunferencia(raio);
double volume = calc.Volume(raio);

Console.WriteLine("Circunferência: " + circ.ToString("F2", CultureInfo.InvariantCulture));
Console.WriteLine("Volume: " + volume.ToString("F2", CultureInfo.InvariantCulture));
Console.WriteLine("Valor de PI: " + calc.Pi.ToString("F2", CultureInfo.InvariantCulture));
```

Discussão

- No problema dos triângulos, cada triângulo possui sua área.
 - `Area()` é uma operação concernente ao objeto: cada triângulo possui sua área.
 - Já no caso da calculadora, os valores dos cálculos não mudam para calculadoras diferentes, ou seja, são cálculos estáticos. O valor de Pi também é estático.



```
Calculadora calc1 = new Calculadora();
Calculadora calc2 = new Calculadora();

calc1 → 

|      |
|------|
| 3.14 |
| Pi   |


          calc1.Pi
          3.14
          calc1.Circunferencia(3.0)
          18.85

calc2 → 

|      |
|------|
| 3.14 |
| Pi   |


          calc2.Pi
          3.14
          calc2.Circunferencia(3.0)
          18.85
```

```
namespace Course {
    class Calculadora {

        public static double Pi = 3.14;

        public static double Circunferencia(double r) {
            return 2.0 * Pi * r;
        }

        public static double Volume(double r) {
            return 4.0 / 3.0 * Pi * r * r * r;
        }
    }
}
```

VERSAO 3

```
Console.WriteLine("Entre o valor do raio: ");
double raio = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

double circ = Calculadora.Circunferencia(raio);
double volume = Calculadora.Volume(raio);

Console.WriteLine("Circunferênci: " + circ.ToString("F2", CultureInfo.InvariantCulture));
Console.WriteLine("Volume: " + volume.ToString("F2", CultureInfo.InvariantCulture));
Console.WriteLine("Valor de PI: " + Calculadora.Pi.ToString("F2",
CultureInfo.InvariantCulture));
```

Exercício de fixação (membros estáticos)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Exercício de fixação

Faça um programa para ler a cotação do dólar, e depois um valor em dólares a ser comprado por uma pessoa em reais. Informar quantos reais a pessoa vai pagar pelos dólares, considerando ainda que a pessoa terá que pagar 6% de IOF sobre o valor em dólar. Criar uma classe **ConversorDeMoeda** para ser responsável pelos cálculos.

Exemplo:

```
Qual é a cotação do dólar? 3.10
Quantos dólares você vai comprar? 200.00
Valor a ser pago em reais = 657.20
```

(correção na próxima página)

```
using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {
            Console.Write("Qual é a cotação do dólar? ");
            double cotacao = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

            Console.Write("Quantos dólares você vai comprar? ");
            double quantia = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

            double result = ConversorDeMoeda.DolarParaReal(quantia, cotacao);

            Console.WriteLine("Valor a ser pago em reais = " + result.ToString("F2", CultureInfo.InvariantCulture));
        }
    }
}
```

```
namespace Course {
    class ConversorDeMoeda {

        public static double Iof = 6.0;

        public static double DolarParaReal(double quantia, double cotacao) {
            double total = quantia * cotacao;
            return total + total * Iof / 100.0;
        }
    }
}
```

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Construtores, palavra this, sobrecarga, encapsulamento

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Construtores

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Construtor

- É uma operação especial da classe, que executa no momento da instânciação do objeto
- Usos comuns:
 - Iniciar valores dos atributos
 - Permitir ou obrigar que o objeto receba dados / dependências no momento de sua instância (injeção de dependência)
- Se um construtor customizado não for especificado, a classe disponibiliza o construtor padrão:
 - Produto p = new Produto();
- É possível especificar mais de um construtor na mesma classe (sobrecarga)

Exemplo:

Entre os dados do produto:

Nome: **TV**

Preço: **900.00**

Quantidade no estoque: **10**

Dados do produto: TV, \$ 900.00, 10 unidades, Total: \$ 9000.00

Digite o número de produtos a ser adicionado ao estoque: **5**

Dados atualizados: TV, \$ 900.00, 15 unidades, Total: \$ 13500.00

Digite o número de produtos a ser removido do estoque: **3**

Dados atualizados: TV, \$ 900.00, 12 unidades, Total: \$ 10800.00

Produto
- Nome : string
- Preco : double
- Quantidade : int
+ ValorTotalEmEstoque() : double
+ AdicionarProdutos(quantidade : int) : void
+ RemoverProdutos(quantidade : int) : void

```

using System.Globalization;

namespace Course {
    class Produto {

        public string Nome;
        public double Preco;
        public int Quantidade;

        public double ValorTotalEmEstoque() {
            return Preco * Quantidade;
        }

        public void AdicionarProdutos(int quantidade) {
            Quantidade += quantidade;
        }

        public void RemoverProdutos(int quantidade) {
            Quantidade -= quantidade;
        }

        public override string ToString() {
            return Nome
                + ", $ "
                + Preco.ToString("F2", CultureInfo.InvariantCulture)
                + " "
                + Quantidade
                + " unidades, Total: $ "
                + ValorTotalEmEstoque().ToString("F2", CultureInfo.InvariantCulture);
        }
    }
}

```

```

using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Produto p = new Produto();

            Console.WriteLine("Entre os dados do produto:");
            Console.Write("Nome: ");
            p.Nome = Console.ReadLine();
            Console.Write("Preço: ");
            p.Preco = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            Console.Write("Quantidade no estoque: ");
            p.Quantidade = int.Parse(Console.ReadLine());

            Console.WriteLine();
            Console.WriteLine("Dados do produto: " + p);

            Console.WriteLine();
            Console.Write("Digite o número de produtos a ser adicionado ao estoque: ");
            int qte = int.Parse(Console.ReadLine());
            p.AdicionarProdutos(qte);

            Console.WriteLine();
            Console.WriteLine("Dados atualizados: " + p);

            Console.WriteLine();
            Console.Write("Digite o número de produtos a ser removido do estoque: ");
            qte = int.Parse(Console.ReadLine());
            p.RemoverProdutos(qte);

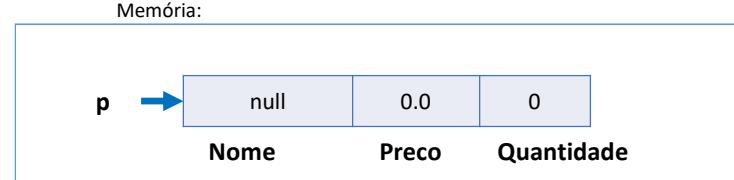
            Console.WriteLine();
            Console.WriteLine("Dados atualizados: " + p);
        }
    }
}

```

Proposta de melhoria

Quando executamos o comando abaixo, instanciamos um produto "p" com seus atributos "vazios":

```
p = new Produto();
```



Entretanto, faz sentido um produto que não tem nome? Faz sentido um produto que não tem preço?

Com o intuito de evitar a existência de produtos sem nome e sem preço, é possível fazer com que seja "obrigatória" a iniciação desses valores?

```
using System.Globalization;

namespace Course {
    class Produto {

        public string Nome;
        public double Preco;
        public int Quantidade;

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

        public double ValorTotalEmEstoque() {
            return Preco * Quantidade;
        }

        public void AdicionarProdutos(int quantidade) {
            Quantidade += quantidade;
        }

        public void RemoverProdutos(int quantidade) {
            Quantidade -= quantidade;
        }

        public override string ToString() {
            return Nome
                + ", $ "
                + Preco.ToString("F2", CultureInfo.InvariantCulture)
                + ", "
                + Quantidade
                + " unidades, Total: $ "
                + ValorTotalEmEstoque().ToString("F2", CultureInfo.InvariantCulture);
        }
    }
}
```

```
using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Console.WriteLine("Entre os dados do produto:");
            Console.Write("Nome: ");
            string nome = Console.ReadLine();
            Console.Write("Preço: ");
            double preco = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            Console.WriteLine("Quantidade no estoque: ");
            int quantidade = int.Parse(Console.ReadLine());

            Produto p = new Produto(nome, preco, quantidade);

            Console.WriteLine();
            Console.WriteLine("Dados do produto: " + p);

            Console.WriteLine();
            Console.WriteLine("Digite o número de produtos a ser adicionado ao estoque: ");
            int qte = int.Parse(Console.ReadLine());
            p.AdicionarProdutos(qte);

            Console.WriteLine();
            Console.WriteLine("Dados atualizados: " + p);

            Console.WriteLine();
            Console.WriteLine("Digite o número de produtos a ser removido do estoque: ");
            qte = int.Parse(Console.ReadLine());
            p.RemoverProdutos(qte);

            Console.WriteLine();
            Console.WriteLine("Dados atualizados: " + p);
        }
    }
}
```

Sobrecarga

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Sobrecarga

- É um recurso que uma classe possui de oferecer mais de uma operação com o mesmo nome, porém com diferentes listas de parâmetros.

Proposta de melhoria

- Vamos criar um construtor opcional, o qual recebe apenas nome e preço do produto. A quantidade em estoque deste novo produto, por padrão, deverá então ser iniciada com o valor zero.
- Nota: é possível também incluir um **construtor padrão** (sem parâmetros)

```
public Produto() {  
}  
  
public Produto(string nome, double preco, int quantidade) {  
    Nome = nome;  
    Preco = preco;  
    Quantidade = quantidade;  
}  
  
public Produto(string nome, double preco) {  
    Nome = nome;  
    Preco = preco;  
    Quantidade = 0;  
}
```

Sintaxe alternativa para inicializar valores

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```
using System.Globalization;

namespace Course {
    class Produto {

        public string Nome;
        public double Preco;
        public int Quantidade;

        public Produto() {
        }

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

        (...)
```

```
Produto p = new Produto("TV", 900.00, 10);
```

```
Produto p = new Produto {
    Nome = "TV",
    Preco = 900.0,
    Quantidade = 0
};

Produto p2 = new Produto() {
    Nome = "TV",
    Preco = 900.0,
    Quantidade = 0
};
```

Isso funciona mesmo se a classe não possuir construtores implementados

Palavra this

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

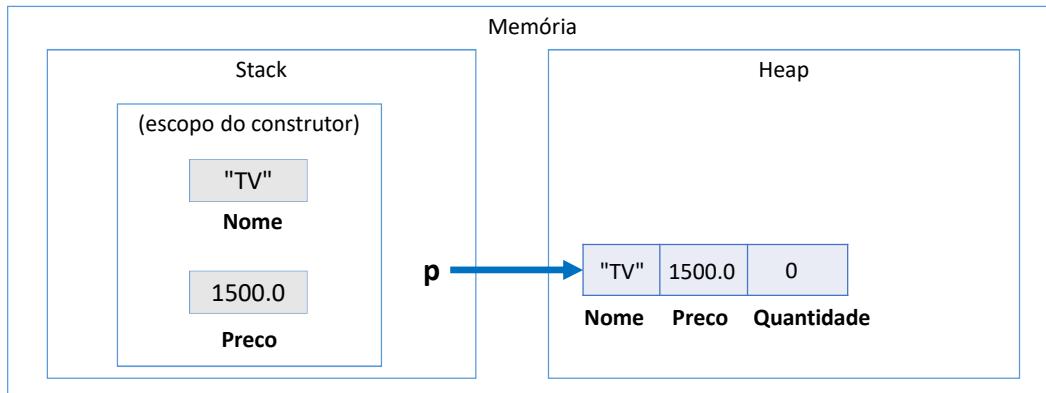
Palavra this

- É uma referência para o próprio objeto
- Usos comuns:
 - Diferenciar atributos de variáveis locais (Java)
 - Referenciar outro construtor em um construtor
 - Passar o próprio objeto como argumento na chamada de um método ou construtor

Diferenciar atributos de variáveis locais

```
Produto p = new Produto("TV", 1500.0);
```

```
public Produto(string Nome, double Preco) {  
    this.Nome = Nome;  
    this.Preco = Preco;  
    Quantidade = 0;  
}
```



Referenciar outro construtor em um construtor

```
using System.Globalization;  
  
namespace Course {  
    class Produto {  
  
        public string Nome;  
        public double Preco;  
        public int Quantidade;  
  
        public Produto() {  
            Quantidade = 0;  
        }  
  
        public Produto(string nome, double preco) : this() {  
            Nome = nome;  
            Preco = preco;  
        }  
  
        public Produto(string nome, double preco, int quantidade) : this(nome, preco) {  
            Quantidade = quantidade;  
        }  
  
        (...)
```

Passar o próprio objeto como argumento na chamada de um método ou construtor

```
class ChessMatch {  
    (...)  
    PlaceNewPiece('e', 1, new King(board, Color.White, this));  
    (...)
```

Encapsulamento

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Encapsulamento

- É um princípio que consiste em esconder detalhes de implementação de um componente, expondo apenas operações seguras e que o mantenham em um estado consistente.
- Regra de ouro: o objeto deve sempre estar em um estado consistente, e a própria classe deve garantir isso.

Analogia:



Opção 1: implementação manual

- Todo atributo é definido como `private`
- Implementa-se métodos `Get` e `Set` para cada atributo, conforme regras de negócio
- Nota: não é usual na plataforma C#

```
using System.Globalization;

namespace Course {
    class Produto {

        private string _nome;
        private double _preco;
        private int _quantidade;

        public Produto() {}

        public Produto(string nome, double preco, int quantidade) {
            _nome = nome;
            _preco = preco;
            _quantidade = quantidade;
        }

        public string GetNome() {
            return _nome;
        }

        public void SetNome(string nome) {
            if (nome != null && nome.Length > 1) {
                _nome = nome;
            }
        }

        public double GetPreco() {
            return _preco;
        }
    }
}
```

```
public int GetQuantidade() {
    return _quantidade;
}

public double ValorTotalEmEstoque() {
    return _preco * _quantidade;
}

public void AdicionarProdutos(int quantidade) {
    _quantidade += quantidade;
}

public void RemoverProdutos(int quantidade) {
    _quantidade -= quantidade;
}

public override string ToString() {
    return _nome
        + ", $ "
        + _preco.ToString("F2", CultureInfo.InvariantCulture)
        + ","
        + _quantidade
        + " unidades, Total: $ "
        + ValorTotalEmEstoque().ToString("F2", CultureInfo.InvariantCulture);
}
}
```

Properties

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Propriedades

- São definições de métodos encapsulados, porém expondo uma sintaxe similar à de atributos e não de métodos
- <https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/properties>
 - Uma propriedade é um membro que oferece um mecanismo flexível para ler, gravar ou calcular o valor de um campo particular. As propriedades podem ser usadas como se fossem atributos públicos, mas na verdade elas são métodos especiais chamados "acessadores". Isso permite que os dados sejam acessados facilmente e ainda ajuda a promover a segurança e a flexibilidade dos métodos.

```
using System.Globalization;

namespace Course {
    class Produto {

        private string _nome;
        private double _preco;
        private int _quantidade;

        public Produto() {
        }

        public Produto(string nome, double preco, int quantidade) {
            _nome = nome;
            _preco = preco;
            _quantidade = quantidade;
        }

        public string Nome {
            get { return _nome; }
            set {
                if (value != null && value.Length > 1) {
                    _nome = value;
                }
            }
        }

        public double Preco {
            get { return _preco; }
        }

        public int Quantidade {
            get { return _quantidade; }
        }
    }
}
```

```
public double ValorTotalEmEstoque {
    get { return _preco * _quantidade; }
}

public void AdicionarProdutos(int quantidade) {
    _quantidade += quantidade;
}

public void RemoverProdutos(int quantidade) {
    _quantidade -= quantidade;
}

public override string ToString() {
    return _nome
        + ", $ "
        + _preco.ToString("F2", CultureInfo.InvariantCulture)
        + ", "
        + _quantidade
        + " unidades, Total: $ "
        + ValorTotalEmEstoque.ToString("F2", CultureInfo.InvariantCulture);
}
}
```

Auto Properties

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Propriedades autoimplementadas

- É uma forma simplificada de se declarar propriedades que não necessitam lógicas particulares para as operações get e set.

```
public double Preco { get; private set; }
```

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/auto-implemented-properties>

```
using System.Globalization;

namespace Course {
    class Produto {

        private string _nome;
        public double Preco { get; private set; }
        public double Quantidade { get; set; }

        public Produto() {
        }

        public Produto(string nome, double preco, int quantidade) {
            _nome = nome;
            Preco = preco;
            Quantidade = quantidade;
        }

        public string Nome {
            get { return _nome; }
            set {
                if (value != null && value.Length > 1) {
                    _nome = value;
                }
            }
        }
    }
}
```

```
public double ValorTotalEmEstoque {
    get { return Preco * Quantidade; }
}

public void AdicionarProdutos(int quantidade) {
    Quantidade += quantidade;
}

public void RemoverProdutos(int quantidade) {
    Quantidade -= quantidade;
}

public override string ToString() {
    return _nome
        + ", $ "
        + Preco.ToString("F2", CultureInfo.InvariantCulture)
        + ","
        + Quantidade
        + " unidades, Total: $ "
        + ValorTotalEmEstoque.ToString("F2", CultureInfo.InvariantCulture);
}
}
```

Ordem sugerida para implementação de membros

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Ordem sugerida

- Atributos privados
- Propriedades autoimplementadas
- Construtores
- Propriedades customizadas
- Outros métodos da classe

Modificadores e acesso

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Modificadores de acesso

- <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/access-modifiers>

Membros

	própria classe	subclasses no assembly	classes do assembly	subclasses fora do assembly	classes fora do assembly
public	x	x	x	x	x
protected internal	x	x	x	x	
internal	x	x	x		
protected	x	x		x	
private protected	x	x			
private	x				

Classes

- Acesso por qualquer classe
 - public class Product
- Acesso somente dentro do assembly
 - internal class Product
 - class Product
- Acesso somente pela classe-mãe
 - private class Product
 - Nota: classe aninhada, por padrão, é private

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Em um banco, para se cadastrar uma conta bancária, é necessário informar o número da conta, o nome do titular da conta, e o valor de depósito inicial que o titular depositou ao abrir a conta. Este valor de depósito inicial, entretanto, é opcional, ou seja: se o titular não tiver dinheiro a depositar no momento de abrir sua conta, o depósito inicial não será feito e o saldo inicial da conta será, naturalmente, zero.

Importante: uma vez que uma conta bancária foi aberta, o número da conta nunca poderá ser alterado. Já o nome do titular pode ser alterado (pois uma pessoa pode mudar de nome por ocasião de casamento, por exemplo).

Por fim, o saldo da conta não pode ser alterado livremente. É preciso haver um mecanismo para proteger isso. O saldo só aumenta por meio de depósitos, e só diminui por meio de saques. Para cada saque realizado, o banco cobra uma taxa de \$ 5.00. Nota: a conta pode ficar com saldo negativo se o saldo não for suficiente para realizar o saque e/ou pagar a taxa.

Você deve fazer um programa que realize o cadastro de uma conta, dando opção para que seja ou não informado o valor de depósito inicial. Em seguida, realizar um depósito e depois um saque, sempre mostrando os dados da conta após cada operação.

(exemplos nas próximas páginas)

EXEMPLO 1

```
Entre o número da conta: 8532
Entre o titular da conta: Alex Green
Haverá depósito inicial (s/n)? s
Entre o valor de depósito inicial: 500.00

Dados da conta:
Conta 8532, Titular: Alex Green, Saldo: $ 500.00

Entre um valor para depósito: 200.00
Dados da conta atualizados:
Conta 8532, Titular: Alex Green, Saldo: $ 700.00

Entre um valor para saque: 300.00
Dados da conta atualizados:
Conta 8532, Titular: Alex Green, Saldo: $ 395.00
```

EXEMPLO 2

```
Entre o número da conta: 7801
Entre o titular da conta: Maria Brown
Haverá depósito inicial (s/n)? n

Dados da conta:
Conta 7801, Titular: Maria Brown, Saldo: $ 0.00

Entre um valor para depósito: 200.00
Dados da conta atualizados:
Conta 7801, Titular: Maria Brown, Saldo: $ 200.00

Entre um valor para saque: 198.00
Dados da conta atualizados:
Conta 7801, Titular: Maria Brown, Saldo: $ -3.00
```

Correção do exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Código fonte no Github

<https://github.com/acenelio/encapsulamento1-csharp>

ContaBancaria
- Numero : Integer
- Titular : String
- Saldo : Double
+ Deposito(quantia : double) : void
+ Saque(quantia : double) : void

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Comportamento de memória, arrays, listas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Tipos referência vs. tipos valor

<http://educandoweb.com.br>

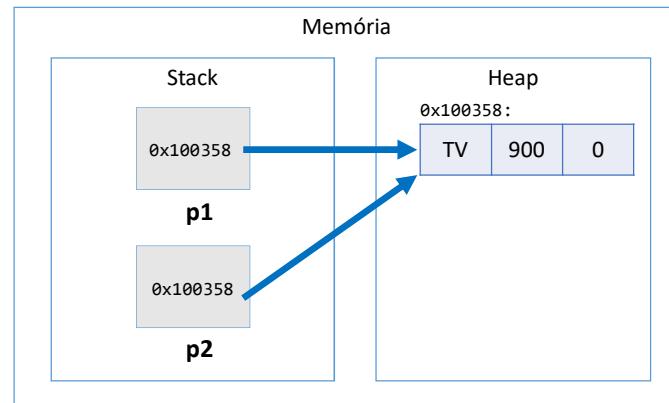
Prof. Dr. Nelio Alves

Classes são tipos referência

Variáveis cujo tipo são classes não devem ser entendidas como caixas, mas sim “tentáculos” (ponteiros) para caixas

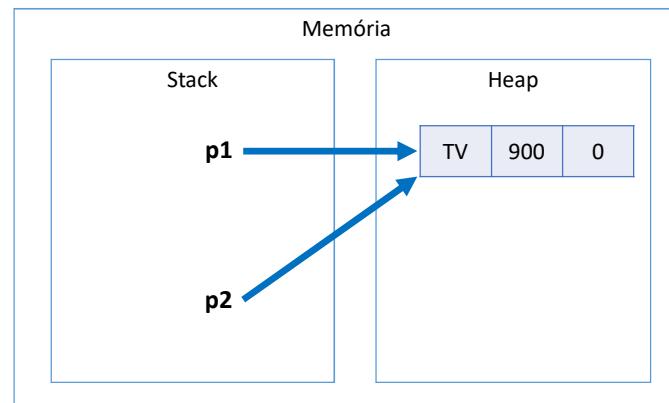
```
Product p1, p2;  
  
p1 = new Product("TV", 900.00, 0);  
  
p2 = p1;
```

p2 = p1;
"p2 passa a apontar para onde
p1 aponta"



Desenho simplificado

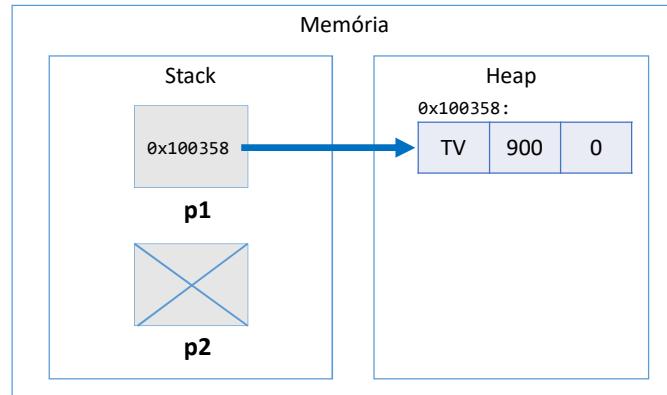
```
Product p1, p2;  
  
p1 = new Product("TV", 900.00, 0);  
  
p2 = p1;
```



Valor "null"

Tipos referência aceitam o valor "null", que indica que a variável aponta pra ninguém.

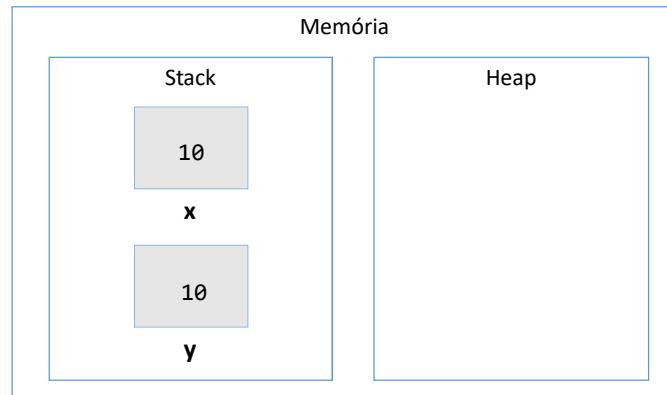
```
Product p1, p2;  
  
p1 = new Product("TV", 900.00, 0);  
  
p2 = null;
```



Structs são tipos valor

A linguagem C# possui também tipos valor, que são os "structs". Structs são CAIXAS e não ponteiros.

```
double x, y;  
  
x = 10;  
  
y = x;  
  
y = x;  
"y recebe uma CÓPIA de x"
```



C# Type	.Net Framework Type	Signed	Bytes	Possible Values
sbyte	System.Sbyte	Yes	1	-128 to 127
short	System.Int16	Yes	2	-32768 to 32767
int	System.Int32	Yes	4	- 2^{31} to $2^{31} - 1$
long	System.Int64	Yes	8	- 2^{63} to $2^{63} - 1$
byte	System.Byte	No	1	0 to 255
ushort	System.UInt16	No	2	0 to 65535
uint	System.UInt32	No	4	0 to $2^{32} - 1$
ulong	System.UInt64	No	8	0 to $2^{64} - 1$
float	System.Single	Yes	4	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ with 7 significant figures
double	System.Double	Yes	8	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ with 15 or 16 significant figures
decimal	System.Decimal	Yes	12	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$ with 28 or 29 significant figures
char	System.Char	N/A	2	Any Unicode character
bool	System.Boolean	N/A	1/2	true or false

Outros structs importantes: DateTime, TimeSpan

É possível criar seus próprios structs

```
namespace Course {
    struct Point {

        public double X, Y;

        public override string ToString() {
            return "(" + X + "," + Y + ")";
        }
    }
}
```

Structs e inicialização

- Demo:

```
Point p;
Console.WriteLine(p); // erro: variável não atribuída
p.X = 10;
p.Y = 20;
Console.WriteLine(p);
p = new Point();
Console.WriteLine(p);
```

Valores padrão

- Quando alocamos (new) qualquer tipo estruturado (classe, struct, array), são atribuídos valores padrão aos seus elementos
 - números: 0
 - bool: False
 - char: caractere código 0
 - objeto: null
- Lembrando: uma variável apenas declarada, mas não instanciada, inicia em estado "não atribuída", e o próprio compilador não permite que ela seja acessada.

Tipos referência vs. tipos valor

CLASSE	STRUCT
Vantagem: usufrui de todos recursos OO	Vantagem: é mais simples e mais performático
Variáveis são ponteiros	Variáveis são caixas
Objetos precisam ser instanciadas usando new, ou apontar para um objeto já existente.	Não é preciso instanciar usando new, mas é possível
Aceita valor null	Não aceita valor null
Suporte a herança	Não tem suporte a herança (mas pode implementar interfaces)
Y = X; "Y passa a apontar para onde X aponta"	Y = X; "Y recebe uma cópia de X"
Objetos instanciados no heap	Objetos instanciados no stack
Objetos não utilizados são desalocados em um momento próximo pelo garbage collector	"Objetos" são desalocados imediatamente quando seu escopo de execução é finalizado

Desalocação de memória - garbage collector e escopo local

<http://educandoweb.com.br>

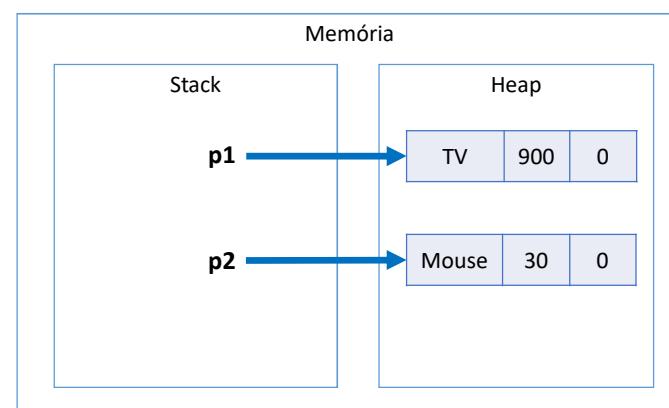
Prof. Dr. Nelio Alves

Garbage collector

- É um processo que automatiza o gerenciamento de memória de um programa em execução
- O garbage collector monitora os objetos alocados dinamicamente pelo programa (no heap), desalocando aqueles que não estão mais sendo utilizados.

Desalocação por garbage collector

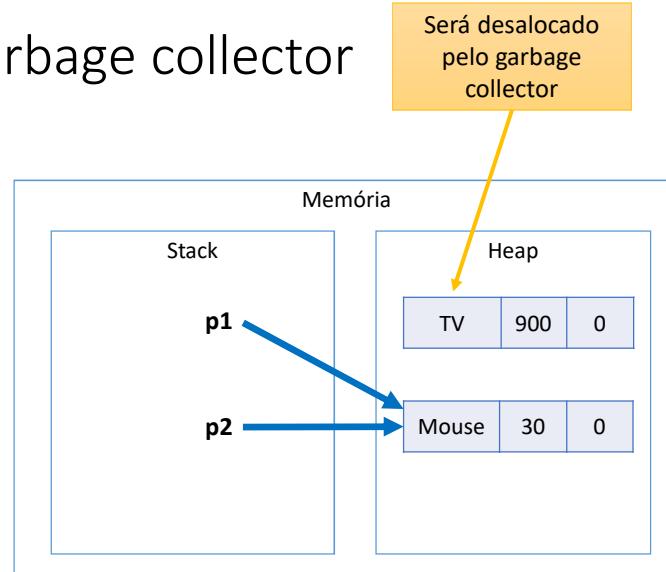
```
Product p1, p2;  
p1 = new Product("TV", 900.00, 0);  
p2 = new Product("Mouse", 30.00, 0);
```



Desalocação por garbage collector

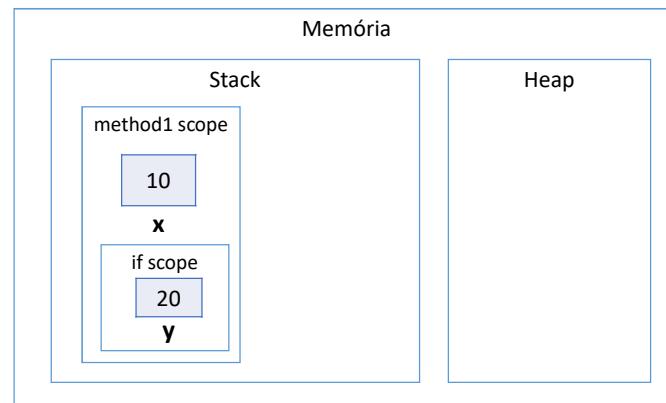
```
Product p1, p2;  
  
p1 = new Product("TV", 900.00, 0);  
  
p2 = new Product("Mouse", 30.00, 0);
```

```
p1 = p2;
```



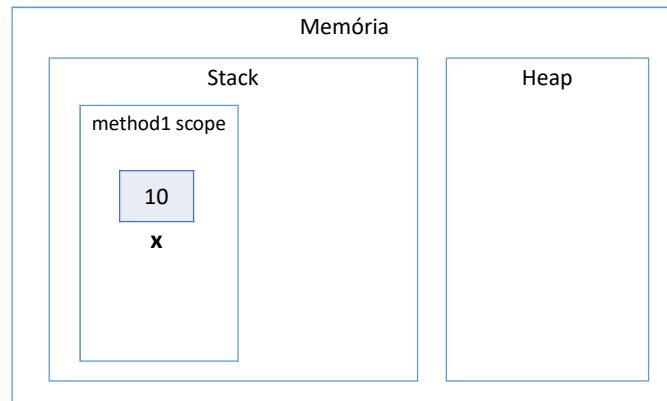
Desalocação por escopo

```
void method1() {  
    int x = 10;  
    if (x > 0) {  
        int y = 20;  
    }  
    Console.WriteLine(x);  
}
```



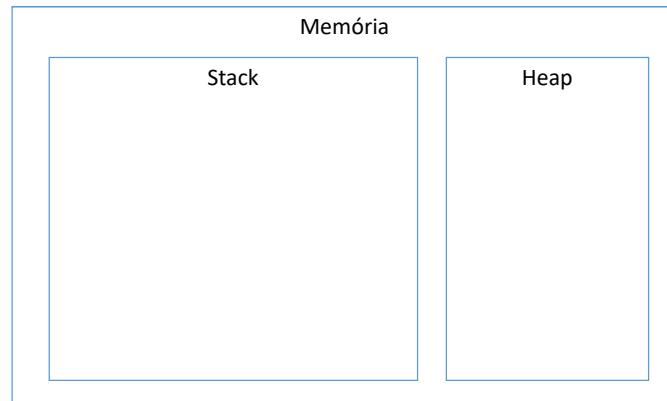
Desalocação por escopo

```
void method1() {  
    int x = 10;  
    if (x > 0) {  
        int y = 20;  
    }  
    Console.WriteLine(x);  
}
```



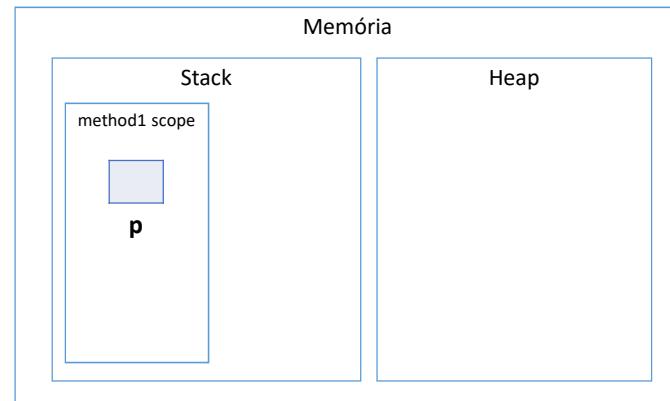
Desalocação por escopo

```
void method1() {  
    int x = 10;  
    if (x > 0) {  
        int y = 20;  
    }  
    Console.WriteLine(x);  
}
```



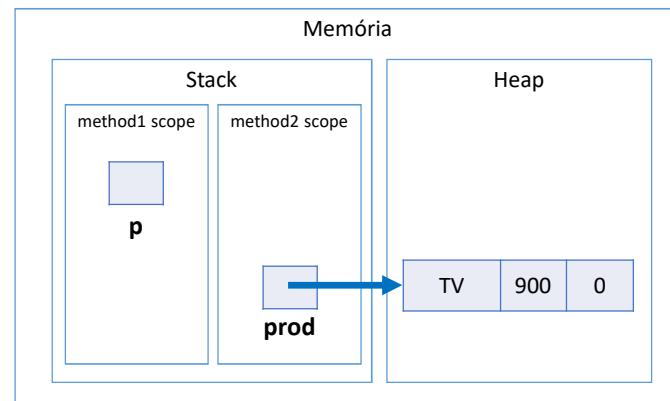
Outro exemplo

```
void method1() {  
    Product p = method2();  
    Console.WriteLine(p.Name);  
}  
  
Product method2() {  
    Product prod = new Product("TV", 900.0, 0);  
    return prod;  
}
```



Outro exemplo

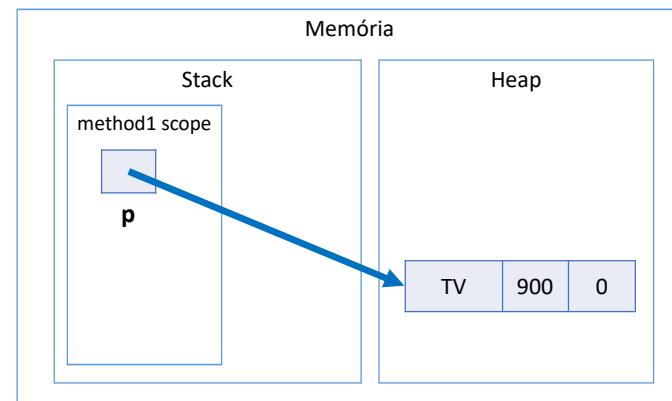
```
void method1() {  
    Product p = method2();  
    Console.WriteLine(p.Name);  
}  
  
Product method2() {  
    Product prod = new Product("TV", 900.0, 0);  
    return prod;  
}
```



Outro exemplo

```
void method1() {
    Product p = method2();
    Console.WriteLine(p.Name);
}

Product method2() {
    Product prod = new Product("TV", 900.0, 0);
    return prod;
}
```



Resumo

- Objetos alocados dinamicamente, quando não possuem mais referência para eles, serão desalocados pelo garbage collector
- Variáveis locais são desalocadas imediatamente assim que seu escopo local sai de execução

Nullable

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Nullable

- É um recurso de C# para que dados de tipo valor (structs) possam receber o valor null
- Uso comum:
 - Campos de banco de dados que podem valer nulo (data de nascimento, algum valor numérico, etc.).
 - Dados e parâmetros opcionais.

Demo

```
double x = null; // erro

Nullable<double> x = null;

double? x = null;

• Métodos:
    • GetValueOrDefault
    • HasValue
    • Value (lança uma exceção se não houver valor)  

• Um nullable não pode ser atribuído para um struct comum  

• Valor default para tipos:  

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/default-value-expressions
```

```
using System;

namespace Course {
    class Program {
        static void Main(string[] args) {

            double? x = null;
            double? y = 10.0;

            Console.WriteLine(x.GetValueOrDefault());
            Console.WriteLine(y.GetValueOrDefault());

            Console.WriteLine(x.HasValue);
            Console.WriteLine(y.HasValue);

            if (x.HasValue)
                Console.WriteLine(x.Value);
            else
                Console.WriteLine("X is null");

            if (y.HasValue)
                Console.WriteLine(y.Value);
            else
                Console.WriteLine("Y is null");
        }
    }
}
```

Operador de coalescência nula

- <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-conditional-operator>

- Demo:

```
double? x = null;
```

```
double y = x ?? 0.0;
```

Vetores - Parte 1

<http://educandoweb.com.br>

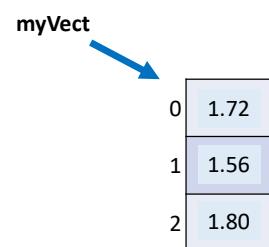
Prof. Dr. Nelio Alves

Checklist

- Revisão do conceito de vetor
- Manipulação de vetor de elementos tipo structs
- Manipulação de vetor de elementos tipo classe
- Acesso aos elementos
- Propriedade Length

Vetores

- Em programação, "vetor" é o nome dado a arranjos unidimensionais
- Arranjo é uma estrutura de dados:
 - Homogênea (dados do mesmo tipo)
 - Ordenada (elementos acessados por meio de posições)
 - Alocada de uma vez só, em um bloco contíguo de memória
- Vantagens:
 - Acesso imediato aos elementos pela sua posição
- Desvantagens:
 - Tamanho fixo
 - Dificuldade para se realizar inserções e deleções

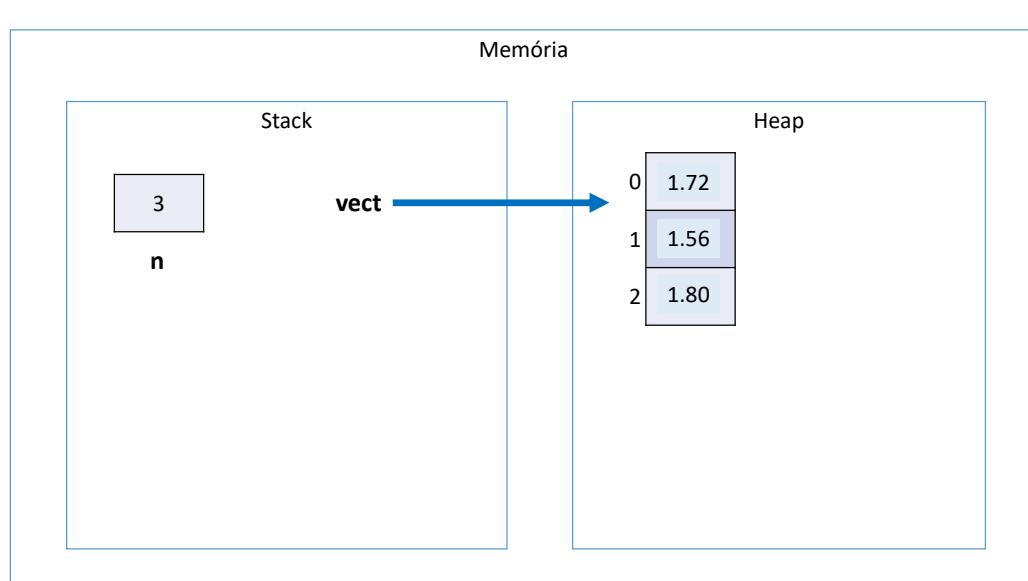


Problema exemplo 1

Fazer um programa para ler um número inteiro N e a altura de N pessoas. Armazene as N alturas em um vetor. Em seguida, mostrar a altura média dessas pessoas.

Exemplo:

Entrada:	Saída:
3 1.72 1.56 1.80	AVERAGE HEIGHT = 1.69



```
using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {

            int n = int.Parse(Console.ReadLine());

            double[] vect = new double[n];

            for (int i = 0; i < n; i++) {
                vect[i] = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
            }

            double sum = 0.0;
            for (int i = 0; i < n; i++) {
                sum += vect[i];
            }

            double avg = sum / n;

            Console.WriteLine("AVERAGE HEIGHT = " + avg.ToString("F2", CultureInfo.InvariantCulture));
        }
    }
}
```

Vetores - Parte 2

<http://educandoweb.com.br>

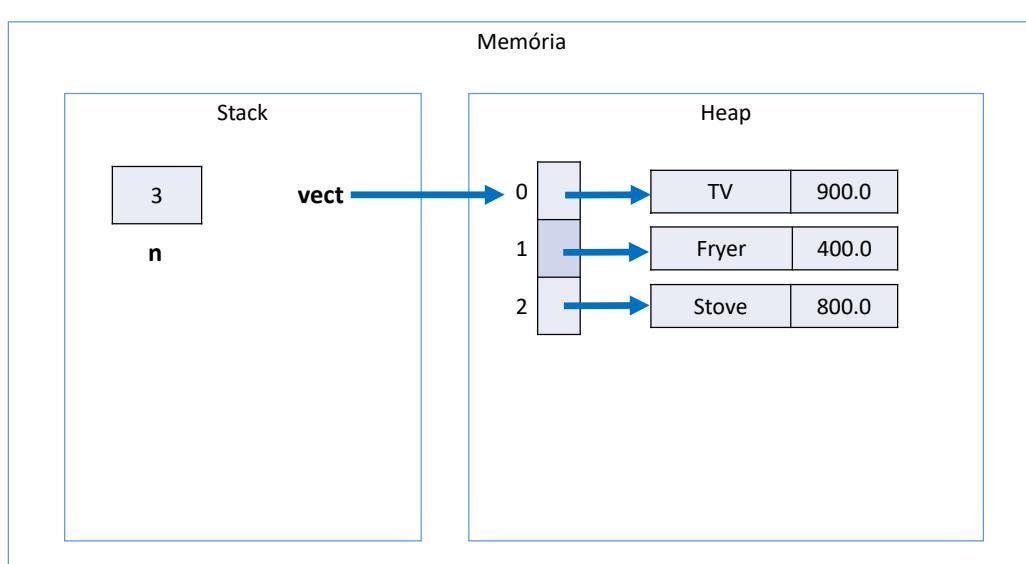
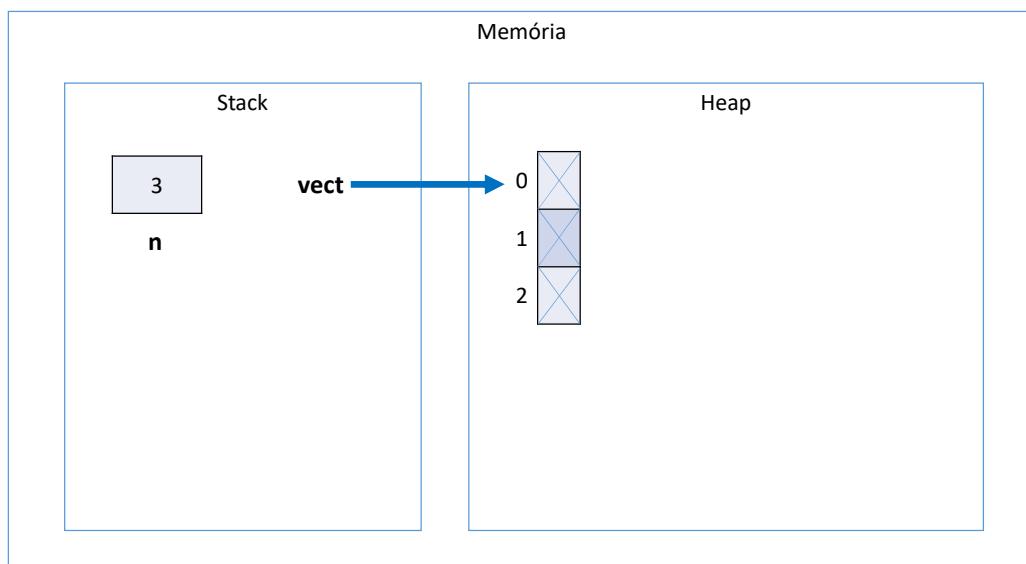
Prof. Dr. Nelio Alves

Problema exemplo 2

Fazer um programa para ler um número inteiro N e os dados (nome e preço) de N Produtos. Armazene os N produtos em um vetor. Em seguida, mostrar o preço médio dos produtos.

Example

Input:	Output:
3 TV 900.00 Fryer 400.00 Stove 800.00	AVERAGE PRICE = 700.00



```
using System;
using System.Globalization;

namespace Course {
    class Program {
        static void Main(string[] args) {

            int n = int.Parse(Console.ReadLine());

            Product[] vect = new Product[n];

            for (int i = 0; i < n; i++) {
                string name = Console.ReadLine();
                double price = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
                vect[i] = new Product { Name = name, Price = price };
            }

            double sum = 0.0;
            for (int i = 0; i < n; i++) {
                sum += vect[i].Price;
            }

            double avg = sum / n;
            Console.WriteLine("AVERAGE PRICE = " + avg.ToString("F2", CultureInfo.InvariantCulture));
        }
    }
}
```

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

A dona de um pensionato possui dez quartos para alugar para estudantes, sendo esses quartos identificados pelos números 0 a 9.

Quando um estudante deseja alugar um quarto, deve-se registrar o nome e email deste estudante.

Fazer um programa que inicie com todos os dez quartos vazios, e depois leia uma quantidade N representando o número de estudantes que vão alugar quartos (N pode ser de 1 a 10). Em seguida, registre o aluguel dos N estudantes. Para cada registro de aluguel, informar o nome e email do estudante, bem como qual dos quartos ele escolheu (de 0 a 9). Suponha que seja escolhido um quarto vago. Ao final, seu programa deve imprimir um relatório de todas ocupações do pensionato, por ordem de quarto, conforme exemplo.

Quantos quartos serão alugados? **3**

Aluguel #1:

Nome: **Maria Green**

Email: **maria@gmail.com**

Quarto: 5

Aluguel #2:

Nome: **Marco Antonio**

Email: **marco@gmail.com**

Quarto: 1

Aluguel #3:

Nome: **Alex Brown**

Email: **alex@gmail.com**

Quarto: 8

Quartos ocupados:

1: Marco Antonio, **marco@gmail.com**

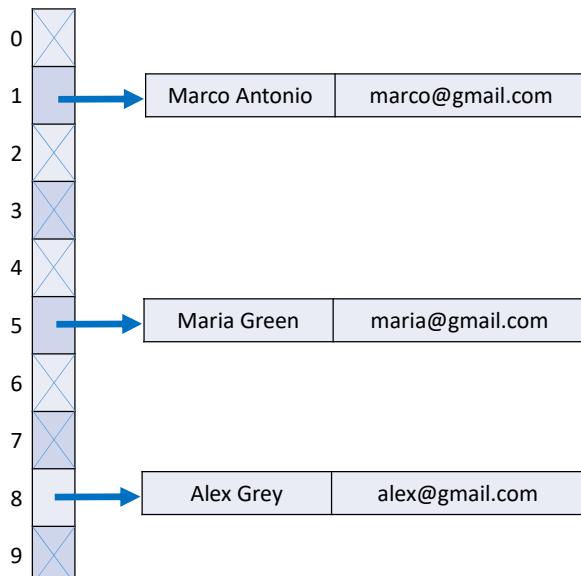
5: Maria Green, **maria@gmail.com**

8: Alex Brown, **alex@gmail.com**

Sugestão

```
if (vect[i] != null)
```

(correção na próxima página)



```
namespace Course {
    class Estudante {
        public string Nome { get; set; }
        public string Email { get; set; }

        public Estudante(string nome, string email) {
            Nome = nome;
            Email = email;
        }

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

```
using System;

namespace Course {
    class Program {
        static void Main(string[] args) {
            Estudante[] vect = new Estudante[10];

            Console.Write("Quantos quartos serão alugados? ");
            int n = int.Parse(Console.ReadLine());

            for (int i = 1; i <= n; i++) {

                Console.WriteLine();
                Console.WriteLine($"Aluguel #{i}:");
                Console.Write("Nome: ");
                string nome = Console.ReadLine();
                Console.Write("Email: ");
                string email = Console.ReadLine();
                Console.WriteLine("Quarto: ");
                int quarto = int.Parse(Console.ReadLine());
                vect[quarto] = new Estudante(nome, email);
            }

            Console.WriteLine();
            Console.WriteLine("Quartos ocupados:");
            for (int i = 0; i < 10; i++) {
                if (vect[i] != null) {
                    Console.WriteLine(i + ": " + vect[i]);
                }
            }
        }
    }
}
```

Modificador de parâmetros: params

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Modificador params

Suponha que se queira uma calculadora para calcular a soma de uma quantidade variável de valores. Solução ruim usando sobrecarga:

```
namespace Course {
    class Calculator {

        public static int Sum(int n1, int n2) {
            return n1 + n2;
        }

        public static int Sum(int n1, int n2, int n3) {
            return n1 + n2 + n3;
        }

        public static int Sum(int n1, int n2, int n3, int n4) {
            return n1 + n2 + n3 + n4;
        }
    }
}
```

Solução com vetor:

```
namespace Course {
    class Calculator {

        public static int Sum(int[] numbers) {
            int sum = 0;
            for (int i=0; i<numbers.Length; i++) {
                sum += numbers[i];
            }
            return sum;
        }
    }
}
```

```
int result = Calculator.Sum(new int[] { 10, 20, 30, 40 });
```

Solução com modificador params:

```
namespace Course {  
    class Calculator {  
  
        public static int Sum(params int[] numbers) {  
            int sum = 0;  
            for (int i=0; i<numbers.Length; i++) {  
                sum += numbers[i];  
            }  
            return sum;  
        }  
    }  
}
```

```
int result = Calculator.Sum(10, 20, 30, 40);
```

Modificador de parâmetros: ref e
out

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Modificador ref

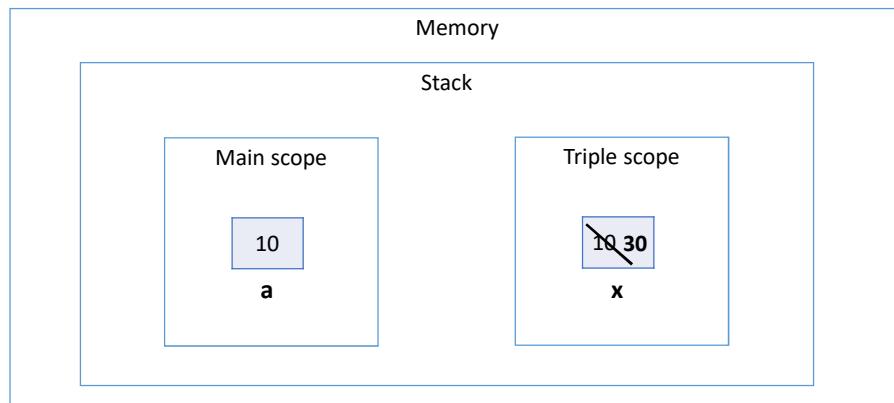
Suponha que se queira uma calculadora com uma operação para triplicar o valor de um número passado como parâmetro. A seguir uma solução que não funciona:

```
class Calculator {  
    public static void Triple(int x) {  
        x = x * 3;  
    }  
}
```

```
class Program {  
    static void Main(string[] args) {  
  
        int a = 10;  
        Calculator.Triple(a);  
        Console.WriteLine(a);  
    }  
}
```

```
class Program {  
    static void Main(string[] args) {  
  
        int a = 10;  
        Calculator.Triple(a);  
        Console.WriteLine(a);  
    }  
}
```

```
class Calculator {  
    public static void Triple(int x) {  
        x = x * 3;  
    }  
}
```

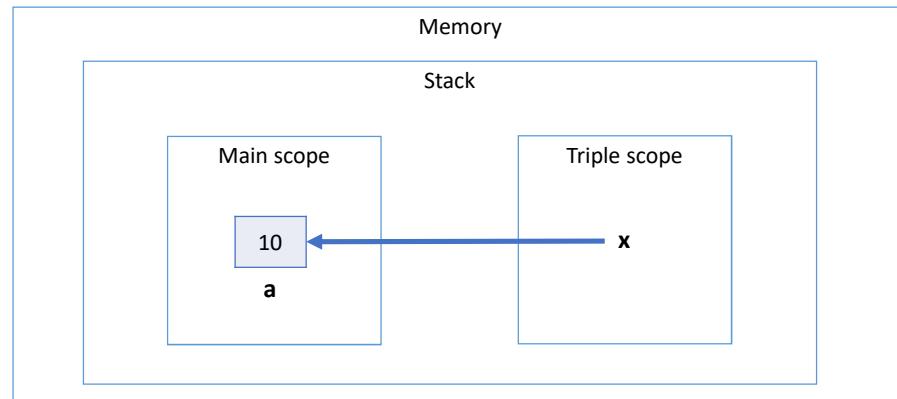


```

class Program {
    static void Main(string[] args) {
        int a = 10;
        Calculator.Triple(ref a);
        Console.WriteLine(a);
    }
}

class Calculator {
    public static void Triple(ref int x) {
        x = x * 3;
    }
}

```



Modificador out

O modificador out é similar ao ref (faz o parâmetro ser uma referência para a variável original), mas não exige que a variável original seja iniciada.

```

class Calculator {

    public static void Triple(int origin, out int result) {
        result = origin * 3;
    }
}

```

```

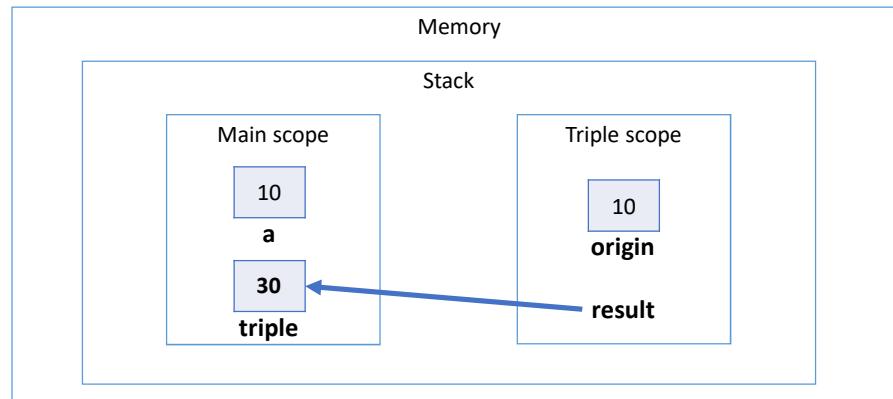
class Program {
    static void Main(string[] args) {

        int a = 10;
        int triple;
        Calculator.Triple(a, out triple);
        Console.WriteLine(triple);
    }
}

```

```
class Program {
    static void Main(string[] args) {
        int a = 10;
        int triple;
        Calculator.Triple(a, out triple);
        Console.WriteLine(triple);
    }
}
```

```
class Calculator {
    public static void Triple(int origin, out int result) {
        result = origin * 3;
    }
}
```



Considerações sobre ref e out

- Diferença:
 - A variável passada como parâmetro **ref** DEVE ter sido iniciada
 - A variável passada como parâmetro **out** não precisa ter sido iniciada
- Conclusão: ambos são muito similares, mas **ref** é uma forma de fazer o compilador obrigar o usuário a iniciar a variável.
- Nota: ambos são considerados "code smells" (design ruim) e devem ser evitados.

Boxing e unboxing

<http://educandoweb.com.br>

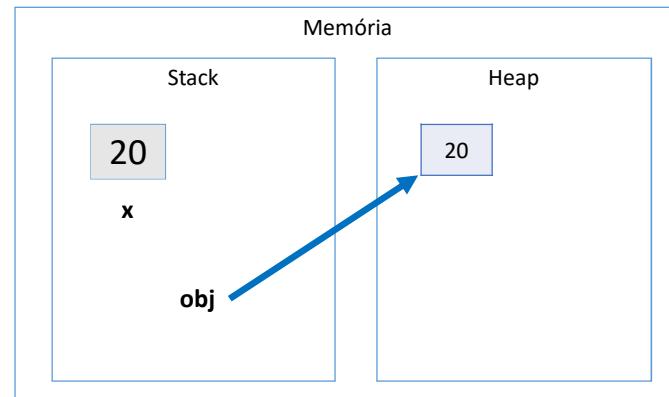
Prof. Dr. Nelio Alves

Boxing

- É o processo de conversão de um objeto tipo valor para um objeto tipo referência compatível

```
int x = 20;
```

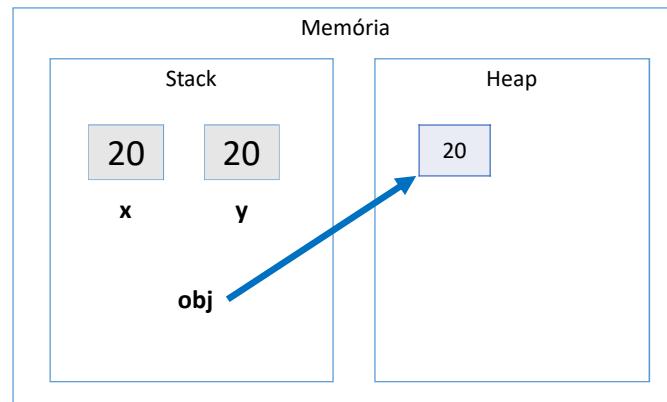
```
Object obj = x;
```



Unboxing

- É o processo de conversão de um objeto tipo referência para um objeto tipo valor compatível

```
int x = 20;  
  
Object obj = x;  
  
int y = (int) obj;
```



Sintaxe opcional: laço foreach

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Laço for each

Sintaxe opcional e simplificada para percorrer coleções

Leitura: "para cada objeto 'obj' contido em vect, faça:"

```
string[] vect = new string[] { "Maria", "Bob", "Alex"};

foreach (string obj in vect) {
    Console.WriteLine(obj);
}
```

Listas - Parte 1

<http://educandoweb.com.br>

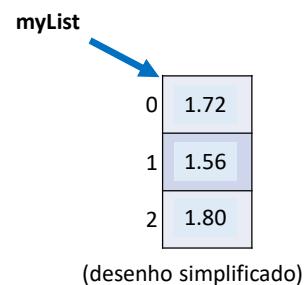
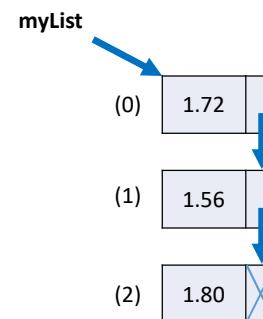
Prof. Dr. Nelio Alves

Checklist

- Conceito de lista
- Tipo List - Declaração, instanciação
- Referência: [https://msdn.microsoft.com/en-us/library/6sh2ey19\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/6sh2ey19(v=vs.110).aspx)
- Assuntos pendentes:
 - generics
 - predicados (lambda)

Listas

- Lista é uma estrutura de dados:
 - Homogênea (dados do mesmo tipo)
 - Ordenada (elementos acessados por meio de posições)
 - Inicia vazia, e seus elementos são alocados sob demanda
 - Cada elemento ocupa um "nó" (ou nodo) da lista
- Classe: List
- Namespace: System.Collections.Generic
- Vantagens:
 - Tamanho variável
 - Facilidade para se realizar inserções e deleções
- Desvantagens:
 - Acesso sequencial aos elementos *



Listas - Parte 2

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Demo

- Inserir elemento na lista: Add, Insert
- Tamanho da lista: Count
- Encontrar primeiro ou último elementos da lista que satisfaça um predicado: list.Find, list.FindLast
- Encontrar primeira ou última posição de elemento da lista que satisfaça um predicado: list.FindIndex, list.FindLastIndex
- Filtrar a lista com base em um predicado: list.FindAll
- Remover elementos da lista: Remove, RemoveAll, RemoveAt, RemoveRange
- Assuntos pendentes:
 - Generics
 - Predicados (lambda)

```

List<string> list = new List<string>();

list.Add("Maria");
list.Add("Alex");
list.Add("Bob");
list.Add("Anna");
list.Insert(2, "Marco");

foreach (string obj in list) {
    Console.WriteLine(obj);
}
Console.WriteLine("List count: " + list.Count);

string s1 = list.Find(x => x[0] == 'A');
Console.WriteLine("First 'A': " + s1);

string s2 = list.FindLast(x => x[0] == 'A');
Console.WriteLine("Last 'A': " + s2);

int pos1 = list.FindIndex(x => x[0] == 'A');
Console.WriteLine("First position 'A': " + pos1);

int pos2 = list.FindLastIndex(x => x[0] == 'A');
Console.WriteLine("Last position 'A': " + pos2);

List<string> list2 = list.FindAll(x => x.Length == 5);
Console.WriteLine("-----");
foreach (string obj in list2) {
    Console.WriteLine(obj);
}

list.Remove("Alex");
Console.WriteLine("-----");
foreach (string obj in list) {
    Console.WriteLine(obj);
}

list.RemoveAll(x => x[0] == 'M');
Console.WriteLine("-----");
foreach (string obj in list) {
    Console.WriteLine(obj);
}

```

Exercício de fixação (listas)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler um número inteiro N e depois os dados (id, nome e salario) de N funcionários. Não deve haver repetição de id.

Em seguida, efetuar o aumento de X por cento no salário de um determinado funcionário. Para isso, o programa deve ler um id e o valor X. Se o id informado não existir, mostrar uma mensagem e abortar a operação. Ao final, mostrar a listagem atualizada dos funcionários, conforme exemplos.

Lembre-se de aplicar a técnica de encapsulamento para não permitir que o salário possa ser mudado livremente. Um salário só pode ser aumentado com base em uma operação de aumento por porcentagem dada.

(exemplo na próxima página)

```
How many employees will be registered? 3
```

```
Employee #1:
```

```
Id: 333
```

```
Name: Maria Brown
```

```
Salary: 4000.00
```

```
Employee #2:
```

```
Id: 536
```

```
Name: Alex Grey
```

```
Salary: 3000.00
```

```
Employee #3:
```

```
Id: 772
```

```
Name: Bob Green
```

```
Salary: 5000.00
```

```
Enter the employee id that will have salary increase : 536
```

```
Enter the percentage: 10.0
```

```
Updated list of employees:
```

```
333, Maria Brown, 4000.00
```

```
536, Alex Grey, 3300.00
```

```
772, Bob Green, 5000.00
```

How many employees will be registered? **2**

Employee #1:

Id: **333**

Name: **Maria Brown**

Salary: **4000.00**

Employee #2:

Id: **536**

Name: **Alex Grey**

Salary: **3000.00**

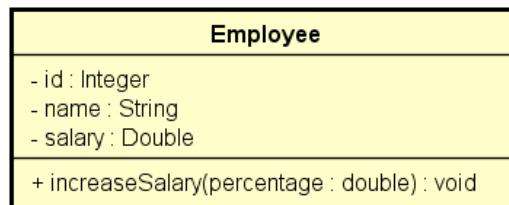
Enter the employee id that will have salary increase: **776**

This id does not exist!

Updated list of employees:

333, Maria Brown, 4000.00

536, Alex Grey, 3000.00



<https://github.com/acenelio/list1-csharp>

Matrizes

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Revisão do conceito de matriz
- Declaração e instanciação
- Acesso aos elementos / como percorrer uma matriz
- Propriedade Length, Rank e GetLength
- Referência: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/multidimensional-arrays>

Matrizes

- Em programação, "matriz" é o nome dado a arranjos bidimensionais
- Arranjo é uma estrutura de dados:
 - Homogênea (dados do mesmo tipo)
 - Ordenada (elementos acessados por meio de posições)
 - Alocada de uma vez só, em um bloco contíguo de memória
- Vantagens:
 - Acesso imediato aos elementos pela sua posição
- Desvantagens:
 - Tamanho fixo
 - Dificuldade para se realizar inserções e deleções

myMat

	0	1	2	3
0	3.5	17.0	12.3	8.2
1	4.1	6.2	7.5	2.9
2	11.0	9.5	14.8	21.7

Demo

```
double[,] mat = new double[2, 3];  
  
Console.WriteLine(mat.Length);  
  
Console.WriteLine(mat.Rank);  
  
Console.WriteLine(mat.GetLength(0));  
  
Console.WriteLine(mat.GetLength(1));
```

Exercício resolvido

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

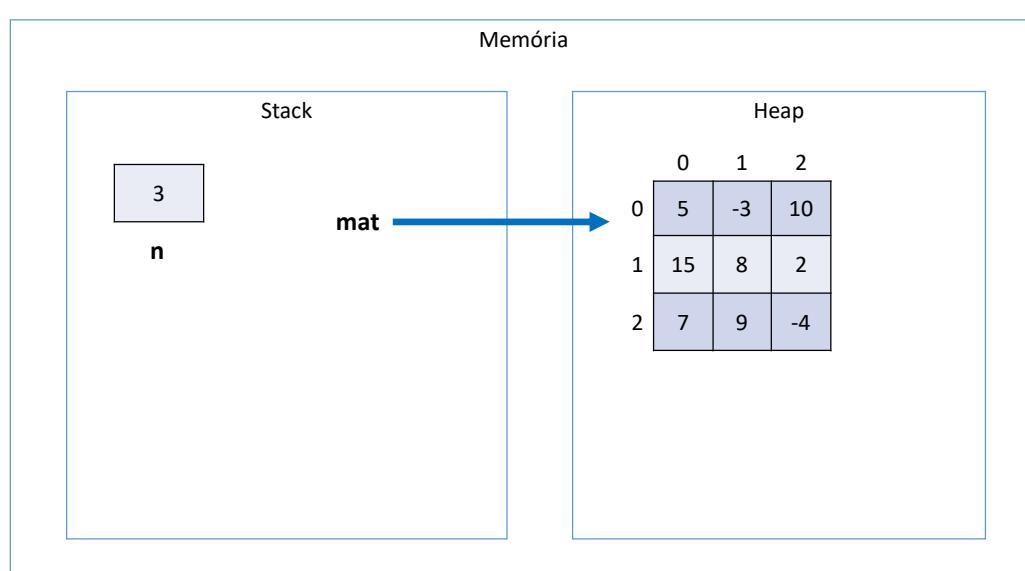
Exercício resolvido

Fazer um programa para ler um número inteiro N e uma matriz de ordem N contendo números inteiros. Em seguida, mostrar a diagonal principal e a quantidade de valores negativos da matriz.

Example

Input:	Output:
3 5 -3 10 15 8 2 7 9 -4	Main diagonal: 5 8 -4 Negative numbers = 2

<https://github.com/acenelio/matrix1-csharp>



Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler dois números inteiros M e N, e depois ler uma matriz de M linhas por N colunas contendo números inteiros, podendo haver repetições. Em seguida, ler um número inteiro X que pertence à matriz. Para cada ocorrência de X, mostrar os valores à esquerda, acima, à direita e abaixo de X, quando houver, conforme exemplo.

Example

```
3 4
10 8 15 12
21 11 23 8
14 5 13 19
8
Position 0,1:
Left: 10
Right: 15
Down: 11
Position 1,3:
Left: 23
Up: 12
Down: 19
```

<https://github.com/acenelio/matrix2-csharp>

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Tópicos especiais em C# - PARTE 1

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Inferência de tipos: palavra var

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Demo

```
var x = 10;  
var y = 20.0;  
var z = "Maria";  
  
Console.WriteLine(x);  
Console.WriteLine(y);  
Console.WriteLine(z);
```

Sintaxe alternativa: switch-case

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

switch-case

Estrutura opcional a vários if-else encadeados, quando a condição envolve o teste do valor de uma variável.

Sintaxe:

```
var minhaVariavel = (...);

switch (minhaVariavel) {
    case 1:
        Console.WriteLine("Caso 1");
        break;
    case 2:
        Console.WriteLine("Caso 2");
        break;
    default:
        Console.WriteLine("Caso padrão");
        break;
}
```

```
int x = int.Parse(Console.ReadLine());
string day;

if (x == 1) {
    day = "Sunday";
}
else if (x == 2) {
    day = "Monday";
}
else if (x == 3) {
    day = "Tuesday";
}
else if (x == 4) {
    day = "Wednesday";
}
else if (x == 5) {
    day = "Thursday";
}
else if (x == 6) {
    day = "Friday";
}
else if (x == 7) {
    day = "Saturday";
}
else {
    day = "Invalid value";
}

Console.WriteLine("Day: " + day);
```

```
int x = int.Parse(Console.ReadLine());
string day;

switch (x) {
    case 1:
        day = "Sunday";
        break;
    case 2:
        day = "Monday";
        break;
    case 3:
        day = "Tuesday";
        break;
    case 4:
        day = "Wednesday";
        break;
    case 5:
        day = "Thursday";
        break;
    case 6:
        day = "Friday";
        break;
    case 7:
        day = "Saturday";
        break;
    default:
        day = "Invalid value";
}

Console.WriteLine("Day: " + day);
```

Expressão condicional ternária

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Expressão condicional ternária

Estrutura opcional ao if-else quando se deseja decidir um **VALOR** com base em uma condição.

Sintaxe:

(*condição*) ? *valor_se_verdadeiro* : *valor_se_falso*

Exemplos:

(2 > 4) ? 50 : 80  80

(10 != 3) ? "Maria" : "Alex"  "Maria"

Demo

```
double preco = 34.5;
double desconto;
if (preco < 20.0) {
    desconto = preco * 0.1;
}
else {
    desconto = preco * 0.05;
}
```

```
double preco = 34.5;
double desconto = (preco < 20.0) ? preco * 0.1 : preco * 0.05;
```

Funções interessantes para string

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Formatar: ToLower(), ToUpper(), Trim()
- Buscar: IndexOf, LastIndexOf
- Recortar: Substring(inicio), Substring(inicio, tamanho)
- Substituir: Replace(char, char), Replace(string, string)
- String.IsNullOrEmpty(str), String.IsNullOrWhiteSpace(str)
- str.Split(' ')
- Conversão para numero: int x = int.Parse(str), int x = Convert.ToInt32(str)
- Conversão de número: str = x.ToString(), str = x.ToString("C"), str = x.ToString("C3"), str = x.ToString("F2")

```
string original = "abcde FGHIJ ABC abc DEFG  ";
string s1 = original.ToUpper();
string s2 = original.ToLower();
string s3 = original.Trim();
int n1 = original.IndexOf("bc");
int n2 = original.LastIndexOf("bc");
string s4 = original.Substring(3);
string s5 = original.Substring(3, 5);
string s6 = original.Replace('a', 'x');
string s7 = original.Replace("abc", "xy");
bool b1 = String.IsNullOrEmpty(original);
bool b2 = String.IsNullOrWhiteSpace(original);

Console.WriteLine("Original: -" + original + "-");
Console.WriteLine("ToUpper: -" + s1 + "-");
Console.WriteLine("ToLower: -" + s2 + "-");
Console.WriteLine("Trim: -" + s3 + "-");
Console.WriteLine("IndexOf('bc'): " + n1);
Console.WriteLine("LastIndexOf('bc'): " + n2);
Console.WriteLine("Substring(3): -" + s4 + "-");
Console.WriteLine("Substring(3, 5): -" + s5 + "-");
Console.WriteLine("Replace('a', 'x'): -" + s6 + "-");
Console.WriteLine("Replace('abc', 'xy'): -" + s7 + "-");
Console.WriteLine("IsNullOrEmpty: " + b1);
Console.WriteLine("IsNullOrWhiteSpace: " + b2);
```

DateTime

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

DateTime

- Representa um INSTANTE
- É um tipo valor (struct)

[https://msdn.microsoft.com/en-us/library/system.datetime\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.datetime(v=vs.110).aspx)

- Agenda:
 - Representação interna
 - Instanciação: construtores, builders / conversão String -> DateTime
 - Formatação: DateTime -> String

Representação interna

- Um objeto DateTime internamente armazena:
 - O número de "ticks" (100 nanosegundos) desde a meia noite do dia 1 de janeiro do ano 1 da era comum

```
DateTime d1 = DateTime.Now;  
Console.WriteLine(d1);  
Console.WriteLine(d1.Ticks);
```

Instanciação

- Construtores
 - DateTime(ano, mes, dia)
 - DateTime(ano, mes, dia, hora, minuto, segundo) *[opcional: kind]*
 - DateTime(ano, mes, dia, hora, minuto, segundo, milissegundo) *[opcional: kind]*
- Builders
 - DateTime.Now
 - DateTime.UtcNow
 - DateTime.Today *[time: 00:00:00]*
 - DateTime.Parse(string)
 - DateTime.ParseExact(string, string)

Demo - construtores

```
DateTime d1 = new DateTime(2000, 8, 15);
DateTime d2 = new DateTime(2000, 8, 15, 13, 5, 58);
DateTime d3 = new DateTime(2000, 8, 15, 13, 5, 58, 275);

Console.WriteLine(d1);
Console.WriteLine(d2);
Console.WriteLine(d3);
```

Demo - Now, UtcNow, Today

```
DateTime d1 = DateTime.Now;
DateTime d2 = DateTime.UtcNow;
DateTime d3 = DateTime.Today;
Console.WriteLine(d1);
Console.WriteLine(d2);
Console.WriteLine(d3);
```

Demo - Parse

```
DateTime d1 = DateTime.Parse("2000-08-15");
DateTime d2 = DateTime.Parse("2000-08-15 13:05:58");
DateTime d3 = DateTime.Parse("15/08/2000");
DateTime d4 = DateTime.Parse("15/08/2000 13:05:58");

Console.WriteLine(d1);
Console.WriteLine(d2);
Console.WriteLine(d3);
Console.WriteLine(d4);
```

Demo - ParseExact

```
DateTime d1 = DateTime.ParseExact("2000-08-15", "yyyy-MM-dd",
CultureInfo.InvariantCulture);

DateTime d2 = DateTime.ParseExact("15/08/2000 13:05:58", "dd/MM/yyyy HH:mm:ss",
CultureInfo.InvariantCulture);

Console.WriteLine(d1);
Console.WriteLine(d2);
```

TimeSpan

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

TimeSpan

- Representa uma DURAÇÃO
- É um tipo valor (struct)

[https://msdn.microsoft.com/en-us/library/system.timespan\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.timespan(v=vs.110).aspx)

Agenda:

- Representação interna
- Instanciação: construtores, fields, métodos From, Parse

Representação interna

- Um objeto TimeSpan internamente armazena uma duração na forma de ticks (100 nanosegundos)

```
TimeSpan t1 = new TimeSpan(0, 1, 30);
Console.WriteLine(t1);
Console.WriteLine(t1.Ticks);
```

Construtores

- TimeSpan()
- TimeSpan(ticks)
- TimeSpan(horas, minutos, segundos)
- TimeSpan(dias, horas, minutos, segundos)
- TimeSpan(dias, horas, minutos, segundos, milissegundos)

Demo - construtores

```
TimeSpan t1 = new TimeSpan();
TimeSpan t2 = new TimeSpan(900000000L);
TimeSpan t3 = new TimeSpan(2, 11, 21);
TimeSpan t4 = new TimeSpan(1, 2, 11, 21);
TimeSpan t5 = new TimeSpan(1, 2, 11, 21, 321);

Console.WriteLine(t1);
Console.WriteLine(t2);
Console.WriteLine(t3);
Console.WriteLine(t4);
Console.WriteLine(t5);
```

Demo - métodos From

```
TimeSpan t1 = TimeSpan.FromDays(1.5);
TimeSpan t2 = TimeSpan.FromHours(1.5);
TimeSpan t3 = TimeSpan.FromMinutes(1.5);
TimeSpan t4 = TimeSpan.FromSeconds(1.5);
TimeSpan t5 = TimeSpan.FromMilliseconds(1.5);
TimeSpan t6 = TimeSpan.FromTicks(900000000L);

Console.WriteLine(t1);
Console.WriteLine(t2);
Console.WriteLine(t3);
Console.WriteLine(t4);
Console.WriteLine(t5);
Console.WriteLine(t6);
```

Propriedades e Operações com DateTime

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Propriedades

- Date (DateTime)
- Day (int)
- DayOfWeek (DayOfWeek)
- DayOfYear (int)
- Hour (int)
- Kind (DateTimeKind)
- Millisecond (int)
- Minute (int)
- Month (int)
- Second (int)
- Ticks (long)
- TimeOfDay (TimeSpan)
- Year (int)

Demo

```
DateTime d = new DateTime(2001, 8, 15, 13, 45, 58, 275);
Console.WriteLine(d);
Console.WriteLine("1) Date: " + d.Date);
Console.WriteLine("2) Day: " + d.Day);
Console.WriteLine("3) DayOfWeek: " + d.DayOfWeek);
Console.WriteLine("4) DayOfYear: " + d.DayOfYear);
Console.WriteLine("5) Hour: " + d.Hour);
Console.WriteLine("6) Kind: " + d.Kind);
Console.WriteLine("7) Millisecond: " + d.Millisecond);
Console.WriteLine("8) Minute: " + d.Minute);
Console.WriteLine("9) Month: " + d.Month);
Console.WriteLine("10) Second: " + d.Second);
Console.WriteLine("11) Ticks: " + d.Ticks);
Console.WriteLine("12) TimeOfDay: " + d.TimeOfDay);
Console.WriteLine("13) Year: " + d.Year);
```

Formatação (DateTime -> string)

```
DateTime d = new DateTime(2001, 8, 15, 13, 45, 58);

string s1 = d.ToString("yyyy-MM-dd HH:mm:ss");
string s2 = d.ToString("yyyy-MM-dd HH:mm:ss.fff");
string s3 = d.ToString("yyyy-MM-dd");
string s4 = d.ToString("HH:mm:ss");
string s5 = d.ToString("dd/MM/yyyy");
string s6 = d.ToString("HH:mm:ss");
string s7 = d.ToString("dd/MM/yyyy HH:mm:ss");

Console.WriteLine(s1);
Console.WriteLine(s2);
Console.WriteLine(s3);
Console.WriteLine(s4);
Console.WriteLine(s5);
Console.WriteLine(s6);
Console.WriteLine(s7);
```

Operações com Datetime

```
DateTime x = ...  
  
DateTime y = x.Add(TimeSpan);  
DateTime y = x.AddDays(double);  
DateTime y = x.AddHours(double);  
DateTime y = x.AddMilliseconds(double);  
DateTime y = x.AddMinutes(double);  
DateTime y = x.AddMonths(int);  
DateTime y = x.AddSeconds(double);  
DateTime y = x.AddTicks(long);  
DateTime y = x.AddYears(int);  
  
DateTime y = x.Subtract(TimeSpan);  
TimeSpan t = x.Subtract(dateTime);
```

Propriedades e Operações com TimeSpan

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Demo: MaxValue, MinValue, Zero

```
TimeSpan t1 = TimeSpan.MaxValue;
TimeSpan t2 = TimeSpan.MinValue;
TimeSpan t3 = TimeSpan.Zero;
Console.WriteLine(t1);
Console.WriteLine(t2);
Console.WriteLine(t3);
```

Demo - propriedades

```
TimeSpan t = new TimeSpan(2, 3, 5, 7, 11);

Console.WriteLine(t);

Console.WriteLine("Days: " + t.Days);
Console.WriteLine("Hours: " + t.Hours);
Console.WriteLine("Minutes: " + t.Minutes);
Console.WriteLine("Milliseconds: " + t.Milliseconds);
Console.WriteLine("Seconds: " + t.Seconds);
Console.WriteLine("Ticks: " + t.Ticks);

Console.WriteLine("TotalDays: " + t.TotalDays);
Console.WriteLine("TotalHours: " + t.TotalHours);
Console.WriteLine("TotalMinutes: " + t.TotalMinutes);
Console.WriteLine("TotalSeconds: " + t.TotalSeconds);
Console.WriteLine("TotalMilliseconds: " + t.TotalMilliseconds);
```

Demo - operações

```
TimeSpan t1 = new TimeSpan(1, 30, 10);
TimeSpan t2 = new TimeSpan(0, 10, 5);

TimeSpan sum = t1.Add(t2);
TimeSpan dif = t1.Subtract(t2);
TimeSpan mult = t2.Multiply(2.0);
TimeSpan div = t2.Divide(2.0);

Console.WriteLine(t1);
Console.WriteLine(t2);
Console.WriteLine(sum);
Console.WriteLine(dif);
Console.WriteLine(mult);
Console.WriteLine(div);
```

DateTimeKind e padrão ISO 8601

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

DateTimeKind

Tipo enumerado especial que define três valores possíveis para a localidade da data:

- Local [*fuso horário do sistema. Exemplo: São Paulo = GMT -3*]
- Utc [*fuso horário GMT (Greenwich Mean Time)*]
- Unspecified

Boa prática

- Armazenar em formato UTC (texto: BD / Json / XML)
- Instanciar e mostrar em formato Local

Para converter um DateTime para Local ou Utc, você deve usar:

- myDateToLocalTime()
- myDateToUniversalTime()

Demo

```
DateTime d1 = new DateTime(2000, 8, 15, 13, 5, 58, DateTimeKind.Local);
DateTime d2 = new DateTime(2000, 8, 15, 13, 5, 58, DateTimeKind.Utc);
DateTime d3 = new DateTime(2000, 8, 15, 13, 5, 58);

Console.WriteLine("d1: " + d1);
Console.WriteLine("d1 Kind: " + d1.Kind);
Console.WriteLine("d1 to Local: " + d1ToLocalTime());
Console.WriteLine("d1 to Utc: " + d1ToUniversalTime());
Console.WriteLine();
Console.WriteLine("d2: " + d2);
Console.WriteLine("d2 Kind: " + d2.Kind);
Console.WriteLine("d2 to Local: " + d2ToLocalTime());
Console.WriteLine("d2 to Utc: " + d2ToUniversalTime());
Console.WriteLine();
Console.WriteLine("d3: " + d3);
Console.WriteLine("d3 Kind: " + d3.Kind);
Console.WriteLine("d3 to Local: " + d3ToLocalTime());
Console.WriteLine("d3 to Utc: " + d3ToUniversalTime());
```

Padrão ISO 8601

- <https://www.iso.org/iso-8601-date-and-time-format.html>
- https://en.wikipedia.org/wiki/ISO_8601
- Formato:

yyyy-MM-ddTHH:mm:ssZ

* Z indica que a data/hora está em Utc

Demo

```
DateTime d1 = DateTime.Parse("2000-08-15 13:05:58");
DateTime d2 = DateTime.Parse("2000-08-15T13:05:58Z"); // cria local DateTime

Console.WriteLine("d1: " + d1);
Console.WriteLine("d1 Kind: " + d1.Kind);
Console.WriteLine("d1 to Local: " + d1ToLocalTime());
Console.WriteLine("d1 to Utc: " + d1ToUniversalTime());
Console.WriteLine();
Console.WriteLine("d2: " + d2);
Console.WriteLine("d2 Kind: " + d2.Kind);
Console.WriteLine("d2 to Local: " + d2ToLocalTime());
Console.WriteLine("d2 to Utc: " + d2ToUniversalTime());
Console.WriteLine();
Console.WriteLine(d2.ToString("yyyy-MM-ddTHH:mm:ssZ")); // cuidado!
Console.WriteLine(d2ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ssZ"));
```

Git e Github

Módulo introdutório

Prof. Dr. Nelio Alves

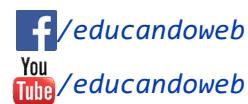
Material de apoio da seção bônus sobre Git e Github do curso:
<https://www.udemy.com/programacao-orientada-a-objetos-csharp/?couponCode=ALUNOSPROMO>

Visão geral sobre Git

- Sistema de versionamento distribuído
- Para utilizar é preciso ter instalado um sistema Git no seu computador

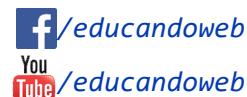


EducandoWeb.com.br



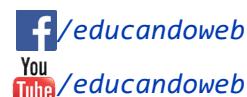
Instalando o Git Bash no Windows

- <https://git-scm.com/downloads>
- Opções recomendadas:
 - Use Git from Git Bash only
 - Checkout Windows-style, commit Unix-style line endings
 - Use MinTTY
 - DESMARQUE: Enable Git Credential Manager



Como criar um novo repositório Git

- O que é e como criar o arquivo .gitignore
- git init

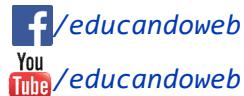


Configurando a identificação do usuário no Git

- `git config --global user.name "Seu Nome"`
- `git config --global user.email "Seu Email"`



EducandoWeb.com.br

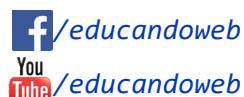


Salvando versões (efetuando commits)

- | | | |
|---|--|--|
| <ul style="list-style-type: none">• untracked
(não rastreados)• <code>git status</code>• <code>git log</code>• <code>git log --oneline</code> | <p><i>git add .</i></p> <p>-></p> <p>staged
(pronto)</p> | <p><i>git commit -m "msg"</i></p> <p>-></p> <p>committed
(salvo)</p> |
|---|--|--|



EducandoWeb.com.br



E se eu esquecer de especificar a mensagem no commit?

Se você esquecer de especificar a mensagem do commit, o Git Bash vai abrir o VIM para edição.

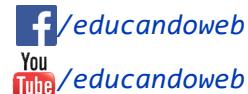
Para sair do VIM tecle <ESC> e depois digite:

:q!

Tecle <ENTER>



EducandoWeb.com.br



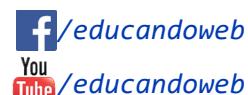
E se eu me perder e quiser voltar ao estado do commit atual?

Descartar todas modificações, voltando ao estado do commit atual:

```
git clean -df  
git checkout -- .
```



EducandoWeb.com.br



E se eu quiser desfazer o último commit?

- 1) Remover o último commit mantendo as alterações nos arquivos:

```
git reset --soft HEAD~1
```

- 2) Remover o último commit INCLUSIVE as alterações nos arquivos (PERIGO!):

```
git reset --hard HEAD~1
```



EducandoWeb.com.br



/educandoweb



You
Tube/educandoweb

E se eu quiser somente dar uma olhada em uma versão anterior?

- 1) Navegar entre commits, alterando os arquivos temporariamente:

```
git checkout <código do commit>
```

- 2) Voltar para o último commit:

```
git checkout <nome do branch>
```



EducandoWeb.com.br



/educandoweb



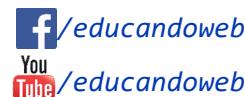
You
Tube/educandoweb

Visão geral do Github

- É um serviço de armazenamento remoto de repositórios Git
- Interface com usuário via web
- Padrão da indústria para armazenamento de projetos de código aberto
- Maior hospedeiro de código fonte do mundo
- Planos pagos para repositórios privados
- É uma "rede social" de repositórios Git. Dica: currículo!



EducandoWeb.com.br

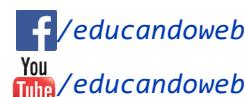


Criando um repositório remoto no Github

- Faça login no Github
- Crie um repositório vazio (sem readme, sem gitignore e sem licença)



EducandoWeb.com.br



Associando um repositório remoto ao seu repositório local

1) Associar nosso repositório local ao repositório remoto, dando o apelido de "origin" a ele:

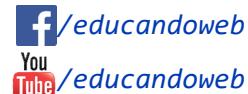
```
git remote add origin <URI do repositório remoto>
```

2) Associar o repositório local a um outro repositório remoto, porém mantendo o mesmo apelido:

```
git remote set-url origin <URI do repositório remoto>
```



EducandoWeb.com.br



Como enviar o repositório local para o Github

```
git push -u origin master
```

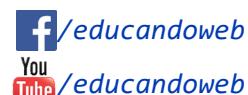
Nota: se você usou a opção -u, nas próximas vezes que for subir o branch master, basta fazer:

```
git push
```

Nota: veja o quanto é fácil visualizar as diferenças entre os commits no Github!



EducandoWeb.com.br



A importância de configurar seu email corretamente no Git Bash

Toda vez que um commit é realizado, é registrado QUEM fez o commit

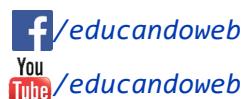
Por isso, sempre que for trabalhar, assegure-se que seu nome e email está devidamente configurado no Git Bash:

```
git config --global user.name "Seu Nome"  
git config --global user.email "Seu Email"
```

Importante: recomenda-se que você use seu email de cadastro no Github, pois assim ficará registrado na rede social que seu usuário do Github é quem fez o commit



EducandoWeb.com.br



Copiando um repositório remoto para seu computador

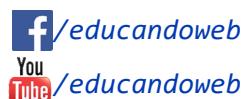
Copiar o repositório, inclusive o histórico de commits:

```
git clone <URI do repositório remoto>
```

ATENÇÃO: simplesmente copiar os arquivos NÃO traz o histórico de commits!



EducandoWeb.com.br



Como atualizar seu repositório local

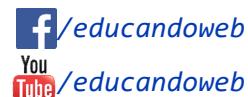
IMPORTANTE: o Git só deixa você continuar um trabalho e depois subi-lo para o repositório remoto, se você mantiver a sequência coerente de commits

Comando para atualizar seu repositório local:

```
git pull origin master
```



EducandoWeb.com.br



Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Enumerações, composição

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Enumerações (enum)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

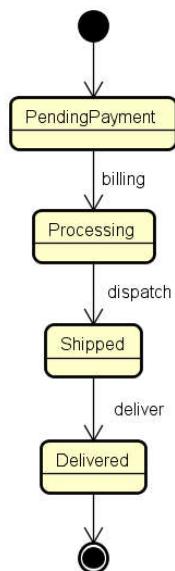
- Definição / discussão
- Exemplo: estados de um pedido
- Conversão de string para enum
- Representação UML

Enumerações

- É um tipo especial que serve para especificar de forma literal um conjunto de constantes relacionadas
- Palavra chave em C#: enum
 - Nota: enum é um tipo valor
- Vantagem: melhor semântica, código mais legível e auxiliado pelo compilador
- Referência: <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/enum>

Exemplo

Ciclo de vida de um pedido.



```
enum OrderStatus : int {  
    PendingPayment = 0,  
    Processing = 1,  
    Shipped = 2,  
    Delivered = 3  
}  
  
class Order {  
    public int Id { get; set; }  
    public DateTime Moment { get; set; }  
    public OrderStatus Status { get; set; }  
}
```

<https://github.com/acenelio/enum1-csharp>

Conversões

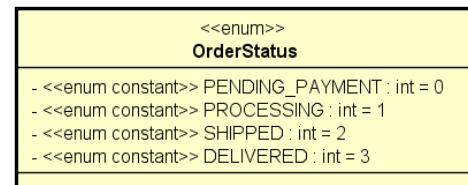
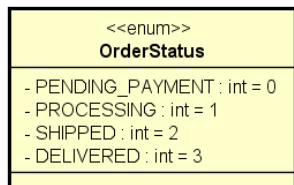
Para o tipo original: use casting

```
OrderStatus os1 = (OrderStatus)2;  
int n1 = (int)OrderStatus.Processing;
```

string - enum:

```
string txt = OrderStatus.PendingPayment.ToString();  
  
OrderStatus os = Enum.Parse<OrderStatus>("Delivered");  
  
Console.WriteLine(os);  
Console.WriteLine(txt);
```

Notação UML



Vamos falar um pouco de design

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Categorias de classes

- Em um sistema orientado a objetos, de modo geral "tudo" é objeto.
- Por questões de design tais como organização, flexibilidade, reuso, delegação, etc., há várias categorias de classes:

Views

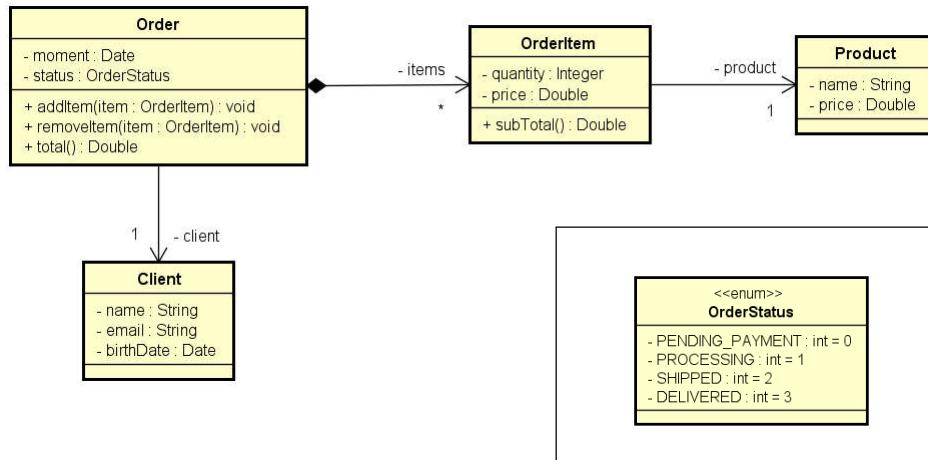
Controllers

Entities

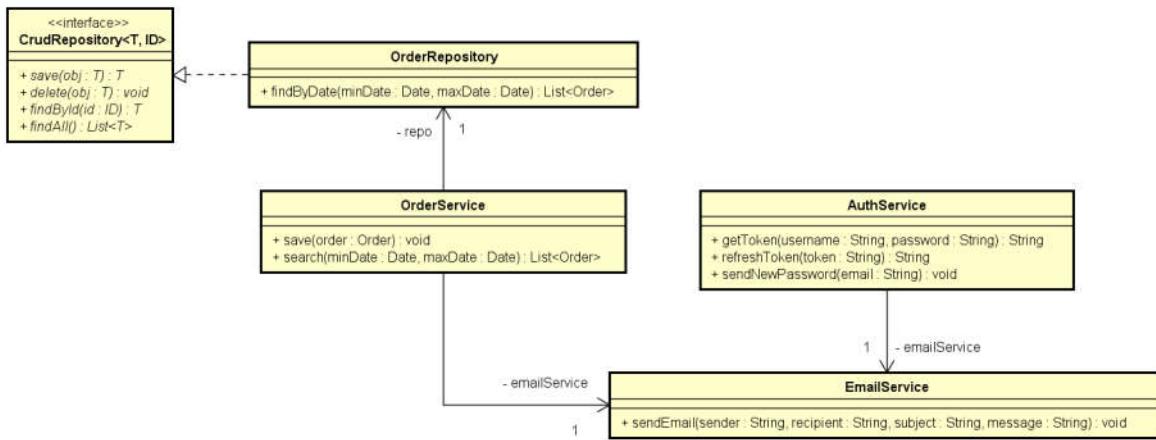
Services

Repositories

Entities



Services



Composição

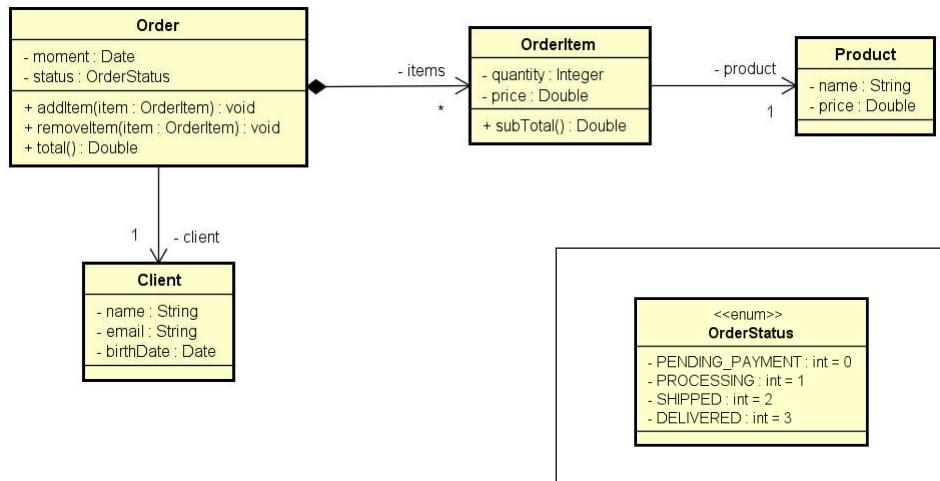
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

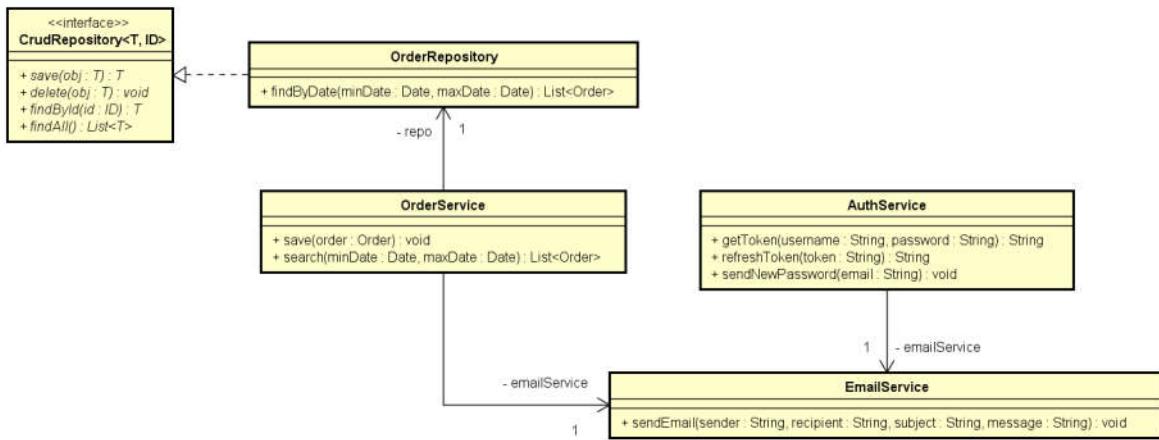
Composição

- É um tipo de associação que permite que um objeto contenha outro
- Relação "tem-um" ou "tem-vários"
- Vantagens
 - Organização: divisão de responsabilidades
 - Coesão
 - Flexibilidade
 - Reuso
- Nota: embora o símbolo UML para composição (todo-parte) seja o diamante preto, neste contexto estamos chamando de composição qualquer associação tipo "tem-um" e "tem-muitos".

Entities



Services

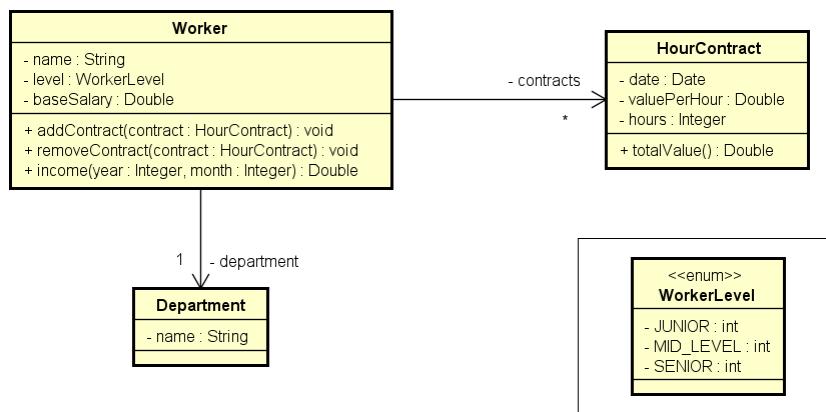


Exercício resolvido 1

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Ler os dados de um trabalhador com N contratos (N fornecido pelo usuário). Depois, solicitar do usuário um mês e mostrar qual foi o salário do funcionário nesse mês, conforme exemplo (próxima página).



```

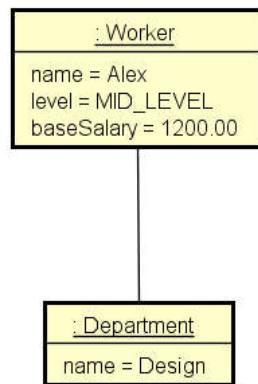
Enter department's name: Design
Enter worker data:
Name: Alex
Level (Junior/MidLevel/Senior): MidLevel
Base salary: 1200.00
How many contracts to this worker? 3
Enter #1 contract data:
Date (DD/MM/YYYY): 20/08/2018
Value per hour: 50.00
Duration (hours): 20
Enter #2 contract data:
Date (DD/MM/YYYY): 13/06/2018
Value per hour: 30.00
Duration (hours): 18
Enter #3 contract data:
Date (DD/MM/YYYY): 25/08/2018
Value per hour: 80.00
Duration (hours): 10

Enter month and year to calculate income (MM/YYYY): 08/2018
Name: Alex
Department: Design
Income for 08/2018: 3000.00

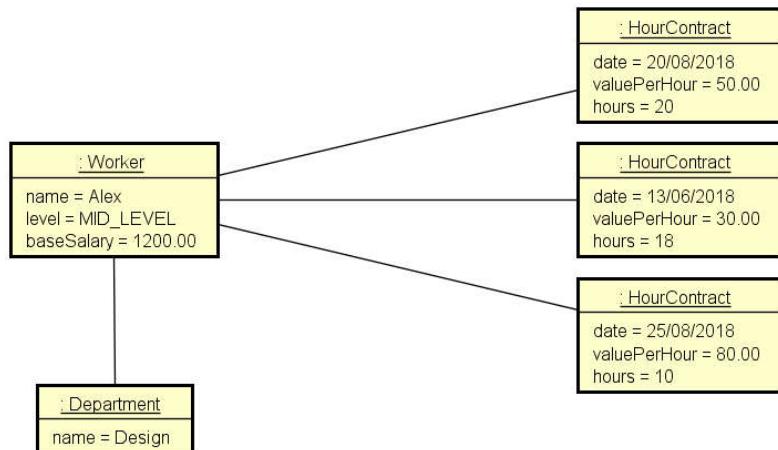
```

<https://github.com/acenelio/composition1-csharp>

Objects in memory:



Objects in memory:



Exercício resolvido 2 (demo
StringBuilder)

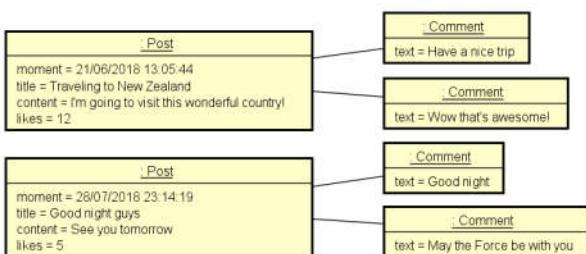
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Instancie manualmente os objetos mostrados abaixo e mostre-os na tela do terminal, conforme exemplo.



Console output:



```

Traveling to New Zealand
12 Likes - 21/06/2018 13:05:44
I'm going to visit this wonderful country!
Comments:
Have a nice trip
Wow that's awesome!

Good night guys
5 Likes - 28/07/2018 23:14:19
See you tomorrow
Comments:
Good night
May the Force be with you

```

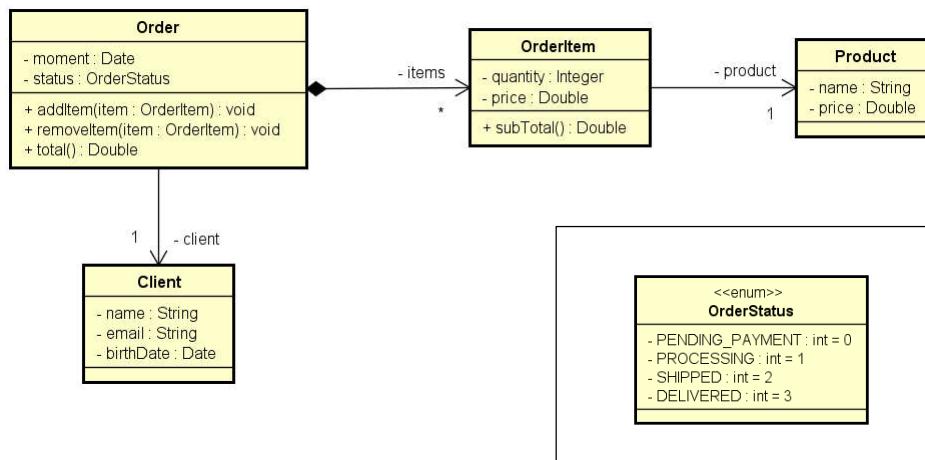
<https://github.com/acenelio/composition2-csharp>

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Ler os dados de um pedido com N itens (N fornecido pelo usuário). Depois, mostrar um sumário do pedido conforme exemplo (próxima página). Nota: o instante do pedido deve ser o instante do sistema: DateTime.Now



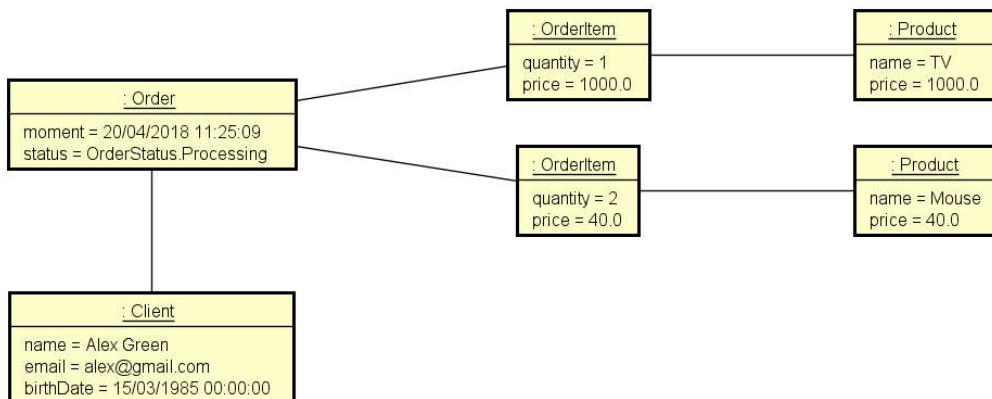
```

Enter cliente data:
Name: Alex Green
Email: alex@gmail.com
Birth date (DD/MM/YYYY): 15/03/1985
Enter order data:
Status: Processing
How many items to this order? 2
Enter #1 item data:
Product name: TV
Product price: 1000.00
Quantity: 1
Enter #2 item data:
Product name: Mouse
Product price: 40.00
Quantity: 2

ORDER SUMMARY:
Order moment: 20/04/2018 11:25:09
Order status: Processing
Client: Alex Green (15/03/1985) - alex@gmail.com
Order items:
TV, $1000.00, Quantity: 1, Subtotal: $1000.00
Mouse, $40.00, Quantity: 2, Subtotal: $80.00
Total price: $1080.00

```

Você deverá instanciar os objetos em memória da seguinte forma:



<https://github.com/acenelio/composition3-csharp>

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Herança e polimorfismo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Herança

<http://educandoweb.com.br>

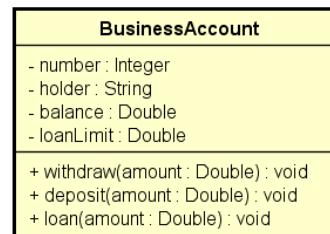
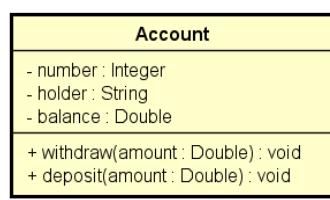
Prof. Dr. Nelio Alves

Herança

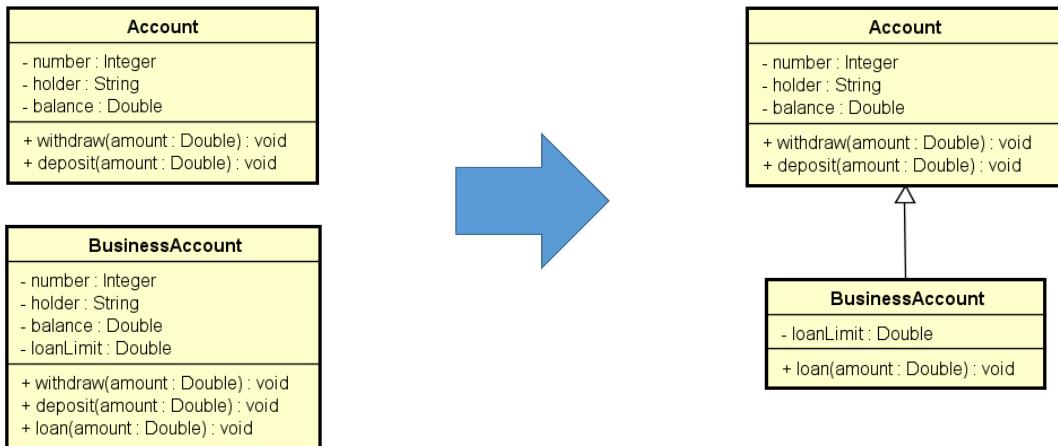
- É um tipo de associação que permite que uma classe herde dados e comportamentos de outra
- Definições importantes
- Vantagens
 - Reuso
 - Polimorfismo
- Sintaxe
 - : (estende)
 - base (referência para a superclasse)

Exemplo

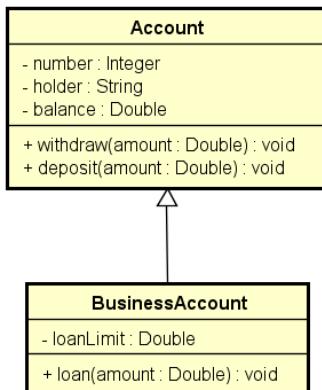
Suponha um negócio de banco que possui uma conta comum e uma conta para empresas, sendo que a conta para empresa possui todos membros da conta comum, mais um limite de empréstimo e uma operação de realizar empréstimo.



Herança permite o reuso de atributos e métodos (dados e comportamento)



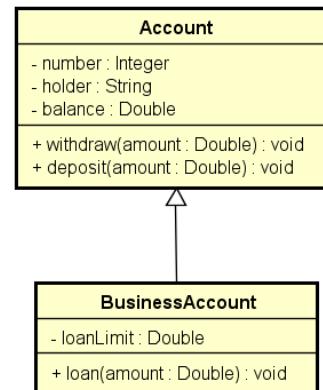
Definições importantes



- Relação "é-um"
- Generalização/especialização
- Superclasse (classe base) / subclasse (classe derivada)
- Herança / extensão
- Herança é uma associação entre classes (e não entre objetos)

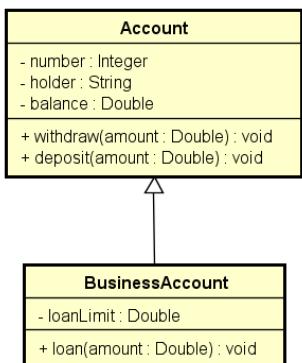
Demo

Vamos implementar as classes Account e BusinessAccount e fazer alguns testes.



Modificadores de acesso

	própria classe	subclasses no assembly	classe do assembly	subclasses fora do assembly	classe fora do assembly
public	x	x	x	x	x
protected internal	x	x	x	x	
internal	x	x	x		
protected	x	x		x	
private protected	x	x			
private	x				



Suponha que, para realizar um empréstimo, é descontada uma taxa no valor de 10.0

Isso resulta em erro:

```

public void Loan(double amount) {
    if (amount <= loanLimit) {
        balance += amount - 10.0;
    }
}

```

<https://github.com/acenelio/inheritance1-csharp>

Modificador de acesso protected

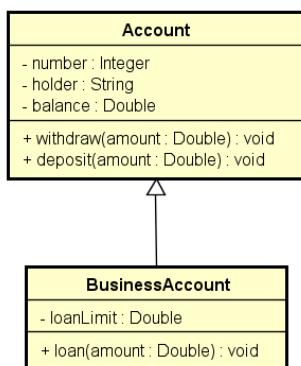
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Membros

	própria classe	subclasses no assembly	classes do assembly	subclasses fora do assembly	classes fora do assembly
public	x	x	x	x	x
protected internal	x	x	x	x	
internal	x	x	x		
protected	x	x		x	
private protected	x	x			
private	x				

Problema exemplo



Se o saldo tiver acesso privativo para alteração, isso resulta em erro:

```
public void Loan(double amount)
{
    if (amount <= LoanLimit)
    {
        Balance += amount;
    }
}
```

<https://github.com/acenelio/inheritance1-csharp>

Upcasting e downcasting

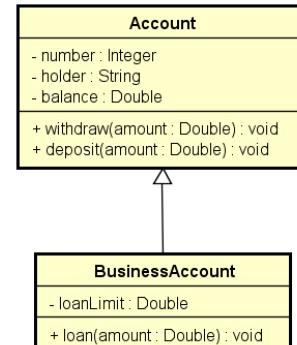
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Upcasting

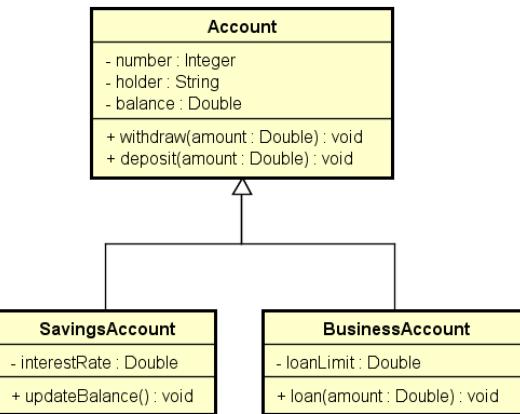
- Casting da subclasse para superclasse
- Uso comum: polimorfismo



- Downcasting

- Casting da superclasse para subclasse
- Palavra `as`
- Palavra `is`
- Uso comum: métodos que recebem parâmetros genéricos (ex: `Equals`)

Exemplo



<https://github.com/acenelio/inheritance2-csharp>

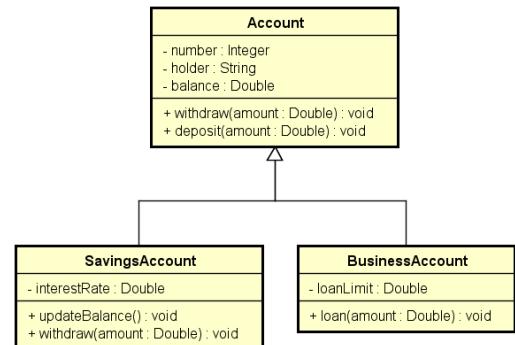
Sobreposição, palavras virtual,
override e base

<http://educandoweb.com.br>

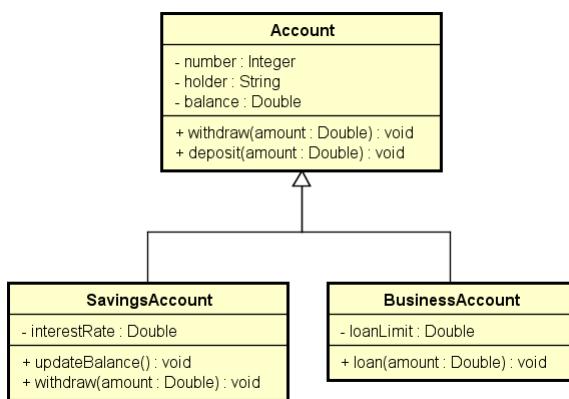
Prof. Dr. Nelio Alves

Sobreposição ou sobrescrita

- É a implementação de um método de uma superclasse na subclasse
- Para que um método comum (não abstrato) possa ser sobreposto, deve ser incluído nele o prefixo "**virtual**"
- Ao sobrescrever um método, devemos incluir nele o prefixo "**override**"



Exemplo



Suponha as seguintes regras para saque:

- Conta comum: é cobrada uma taxa no valor de 5.00.
- Conta poupança: não é cobrada taxa.

Como resolver isso?

Resposta: sobrescrevendo o método withdraw na subclasse **SavingsAccount**

Account:

```
public virtual void Withdraw(double amount) {  
    Balance -= amount + 5.0;  
}
```

SavingsAccount:

```
public override void Withdraw(double amount) {  
    Balance -= amount;  
}
```

Palavra base

É possível chamar a implementação da superclasse usando a palavra base.

Exemplo: suponha que a regra para saque para conta poupança seja realizar o saque normalmente da superclasse (Account), e depois descontar mais 2.0.

```
public override void Withdraw(double amount) {  
    base.Withdraw(amount);  
    Balance -= 2.0;  
}
```

Recordando: usando `base` em construtores

```
class BusinessAccount : Account
{
    public double LoanLimit { get; set; }

    public BusinessAccount()
    {
    }

    public BusinessAccount(int number, string holder, double balance, double loanLimit)
        : base(number, holder, balance)
    {
        LoanLimit = loanLimit;
    }

    (...)
```

Código fonte desta aula

<https://github.com/acenelio/inheritance3-csharp>

Classes e métodos selados

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Classes e métodos selados

- Palavra chave: sealed
- **Classe:** evita que a classe seja herdada
 - Nota: ainda é possível extender a funcionalidade de uma classe selada usando "extension methods"

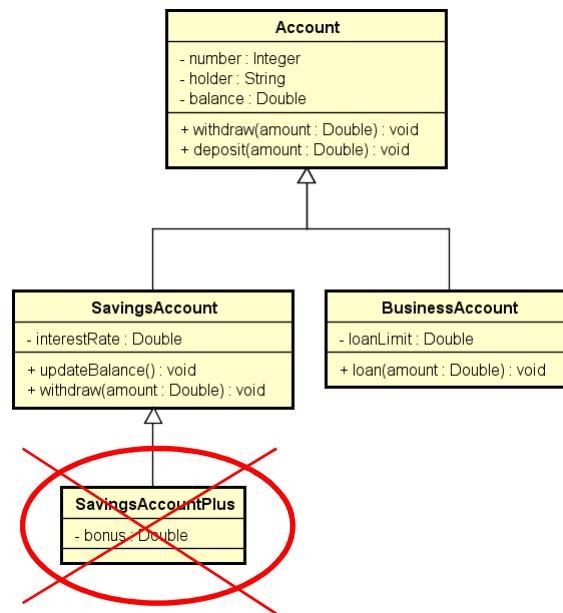
```
namespace Course {  
    sealed class SavingsAccount {
```

- **Método:** evita que um método sobreposto possa ser sobreposto novamente
 - Só pode ser aplicado a métodos sobrepostos

Exemplo - Classe selada

Suponha que você queira evitar que sejam criadas subclasses de SavingsAccount

```
namespace Course {  
    sealed class SavingsAccount {  
        (...)
```



Exemplo - método selado

Suponha que você não queira que o método Withdraw de SavingsAccount seja sobreposto novamente

```
public sealed override void Withdraw(double amount)  
{  
    base.Withdraw(amount);  
    Balance -= 2.0;  
}
```

Pra quê?

- Segurança: dependendo das regras do negócio, às vezes é desejável garantir que uma classe não seja herdada, ou que um método não seja sobreposto.
 - Geralmente convém selar métodos sobrepostos, pois sobreposições múltiplas podem ser uma porta de entrada para inconsistências
- Performance: atributos de tipo de uma classe selada são analisados de forma mais rápida em tempo de execução.
 - Exemplo clássico: string

Introdução ao polimorfismo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Pilares da OOP

- Encapsulamento
- Herança
- Polimorfismo

Polimorfismo

Em Programação Orientada a Objetos, polimorfismo é recurso que permite que variáveis de um mesmo tipo mais genérico possam apontar para objetos de tipos específicos diferentes, tendo assim comportamentos diferentes conforme cada tipo específico.

```
Account acc1 = new Account(1001, "Alex", 500.0);
Account acc2 = new SavingsAccount(1002, "Anna", 500.0, 0.01);

acc1.Withdraw(10.0);
acc2.Withdraw(10.0);

Console.WriteLine(acc1.Balance);
Console.WriteLine(acc2.Balance);
```

```

Account acc1 = new Account(1001, "Alex", 500.0);
Account acc2 = new SavingsAccount(1002, "Anna", 500.0, 0.01);

acc1.Withdraw(10.0);
acc2.Withdraw(10.0);

```

Account:

```

public virtual void Withdraw(double amount) {
    Balance -= amount + 5.0;
}

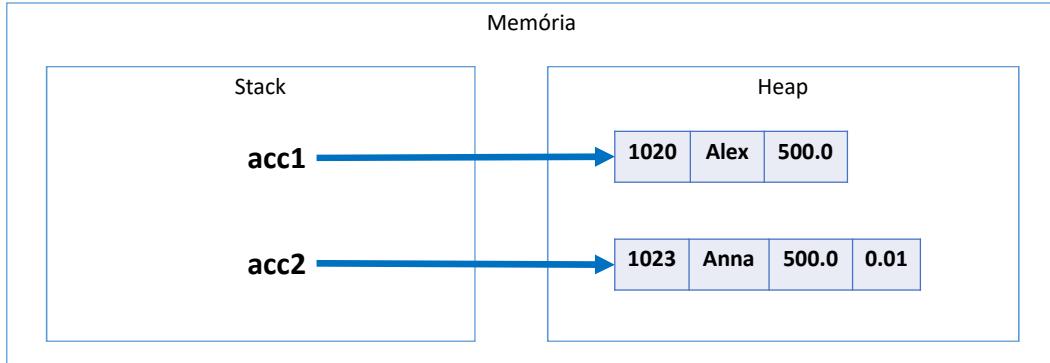
```

SavingsAccount:

```

public override void Withdraw(double amount) {
    base.Withdraw(amount);
    Balance -= 2.0;
}

```



Importante entender

- A associação do tipo específico com o tipo genérico é feita **em tempo de execução** (upcasting).
- O compilador não sabe para qual tipo específico a chamada do método Withdraw está sendo feita (ele só sabe que são duas variáveis tipo Account):

```

Account acc1 = new Account(1001, "Alex", 500.0);
Account acc2 = new SavingsAccount(1002, "Anna", 500.0, 0.01);

acc1.Withdraw(10.0);
acc2.Withdraw(10.0);

```

Exercício resolvido

<http://educandoweb.com.br>

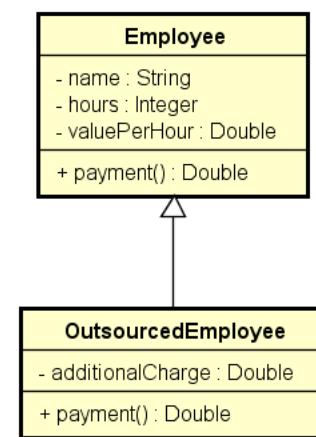
Prof. Dr. Nelio Alves

Uma empresa possui funcionários próprios e terceirizados. Para cada funcionário, deseja-se registrar nome, horas trabalhadas e valor por hora. Funcionários terceirizados possuem ainda uma despesa adicional.

O pagamento dos funcionários corresponde ao valor da hora multiplicado pelas horas trabalhadas, sendo que os funcionários terceirizados ainda recebem um bônus correspondente a 110% de sua despesa adicional.

Fazer um programa para ler os dados de N funcionários (N fornecido pelo usuário) e armazená-los em uma lista. Depois de ler todos os dados, mostrar nome e pagamento de cada funcionário na mesma ordem em que foram digitados.

Construa o programa conforme projeto ao lado. Veja exemplo na próxima página.



```
Enter the number of employees: 3
```

```
Employee #1 data:
```

```
Outsourced (y/n)? n
```

```
Name: Alex
```

```
Hours: 50
```

```
Value per hour: 20.00
```

```
Employee #2 data:
```

```
Outsourced (y/n)? y
```

```
Name: Bob
```

```
Hours: 100
```

```
Value per hour: 15.00
```

```
Additional charge: 200.00
```

```
Employee #3 data:
```

```
Outsourced (y/n)? n
```

```
Name: Maria
```

```
Hours: 60
```

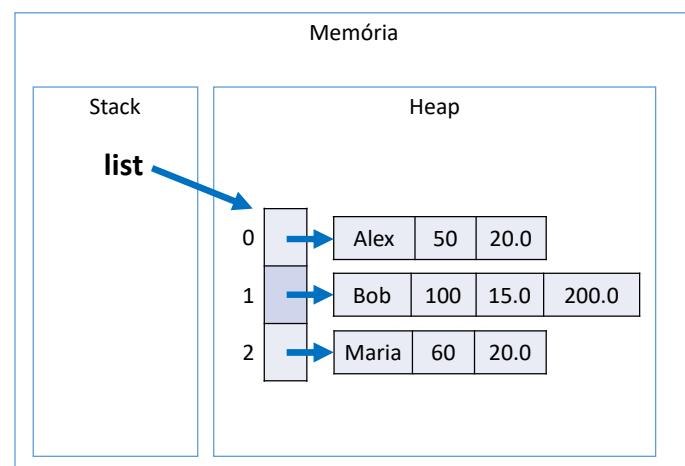
```
Value per hour: 20.00
```

```
PAYMENTS:
```

```
Alex - $ 1000.00
```

```
Bob - $ 1720.00
```

```
Maria - $ 1200.00
```



<https://github.com/acenelio/inheritance4-csharp>

Exercício de fixação

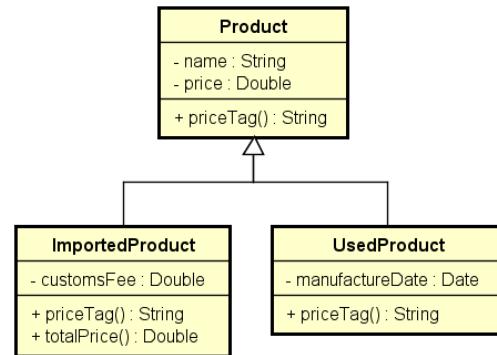
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler os dados de N produtos (N fornecido pelo usuário). Ao final, mostrar a etiqueta de preço de cada produto na mesma ordem em que foram digitados.

Todo produto possui nome e preço. Produtos importados possuem uma taxa de alfândega, e produtos usados possuem data de fabricação. Estes dados específicos devem ser acrescentados na etiqueta de preço conforme exemplo (próxima página). Para produtos importados, a taxa e alfândega deve ser acrescentada ao preço final do produto.

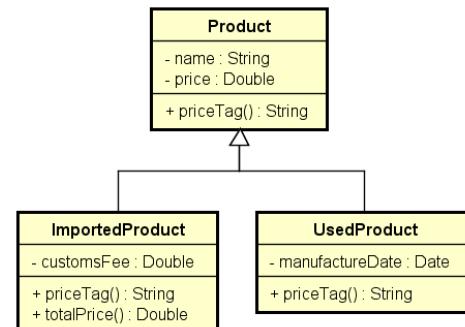
Favor implementar o programa conforme projeto ao lado.



```

Enter the number of products: 3
Product #1 data:
Common, used or imported (c/u/i)? i
Name: Tablet
Price: 260.00
Customs fee: 20.00
Product #2 data:
Common, used or imported (c/u/i)? c
Name: Notebook
Price: 1100.00
Product #3 data:
Common, used or imported (c/u/i)? u
Name: Iphone
Price: 400.00
Manufacture date (DD/MM/YYYY): 15/03/2017

PRICE TAGS:
Tablet $ 280.00 (Customs fee: $ 20.00)
Notebook $ 1100.00
Iphone (used) $ 400.00 (Manufacture date: 15/03/2017)
  
```



<https://github.com/acenelio/inheritance5-csharp>

Classes abstratas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Classes abstratas

- São classes que não podem ser instanciadas
- É uma forma de garantir herança total: somente subclasses não abstratas podem ser instanciadas, mas nunca a superclasse abstrata

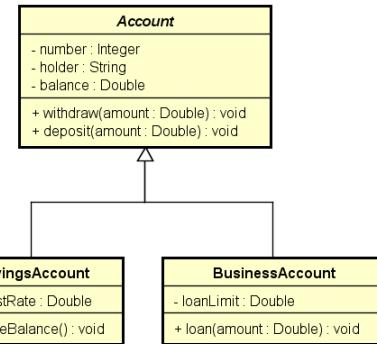
Exemplo

Suponha que em um negócio relacionado a banco, apenas contas poupança e contas para empresas são permitidas. Não existe conta comum.

Para garantir que contas comuns não possam ser instanciadas, basta acrescentarmos a palavra "abstract" na declaração da classe.

```
namespace Course {  
    abstract class Account {  
        (...)
```

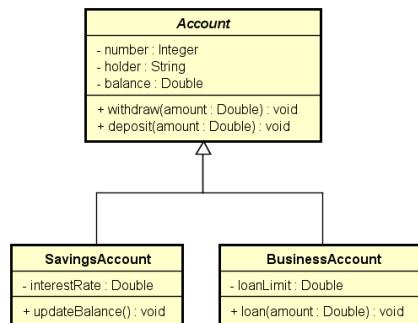
Notação UML: itálico



Vamos partir da implementação em: <https://github.com/acenelio/inheritance3-csharp>

Questionamento

- Se a classe Account não pode ser instanciada, por que simplesmente não criar somente SavingsAccount e BusinessAccount?
- Resposta:
 - **Reuso**
 - **Polimorfismo:** a superclasse classe genérica nos permite tratar de forma fácil e uniforme todos os tipos de conta, inclusive com polimorfismo se for o caso (como fizemos nos últimos exercícios). Por exemplo, você pode colocar todos tipos de contas em uma mesma coleção.
- Demo: suponha que você queira:
 - Totalizar o saldo de todas as contas.
 - Sacar 10.00 de todas as contas.



<https://github.com/acenelio/inheritance6-csharp>

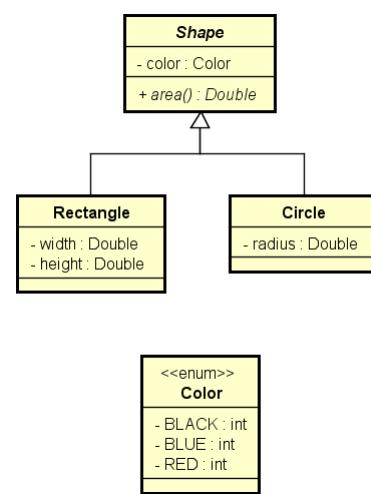
Métodos abstratos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Métodos abstratos

- São métodos que não possuem implementação.
- Métodos precisam ser abstratos quando a classe é genérica demais para conter sua implementação.
- Se uma classe possuir pelo menos um método abstrato, então esta classe também é abstrata.
- Notação UML: itálico
- Exercício resolvido



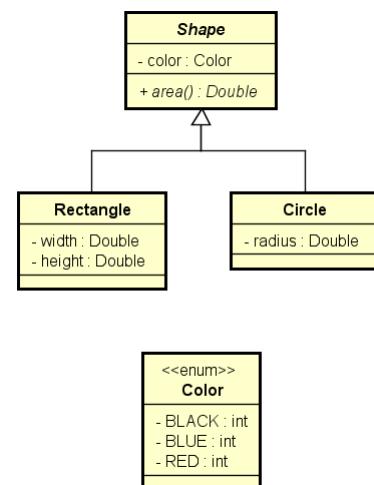
Exercício resolvido (métodos abstratos)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler os dados de N figuras (N fornecido pelo usuário), e depois mostrar as áreas destas figuras na mesma ordem em que foram digitadas.

```
Enter the number of shapes: 2
Shape #1 data:
Rectangle or Circle (r/c)? r
Color (Black/Blue/Red): Black
Width: 4.0
Height: 5.0
Shape #2 data:
Rectangle or Circle (r/c)? c
Color (Black/Blue/Red): Red
Radius: 3.0
SHAPE AREAS:
20.00
28.27
```



<https://github.com/acenelio/inheritance7-csharp>

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler os dados de N contribuintes (N fornecido pelo usuário), os quais podem ser pessoa física ou pessoa jurídica, e depois mostrar o valor do imposto pago por cada um, bem como o total de imposto arrecadado.

Os dados de pessoa física são: nome, renda anual e gastos com saúde. Os dados de pessoa jurídica são nome, renda anual e número de funcionários. As regras para cálculo de imposto são as seguintes:

Pessoa física: pessoas cuja renda foi abaixo de 20000.00 pagam 15% de imposto. Pessoas com renda de 20000.00 em diante pagam 25% de imposto. Se a pessoa teve gastos com saúde, 50% destes gastos são abatidos no imposto.

Exemplo: uma pessoa cuja renda foi 50000.00 e teve 2000.00 em gastos com saúde, o imposto fica: $(50000 * 25\%) - (2000 * 50\%) = \mathbf{11500.00}$

Pessoa jurídica: pessoas jurídicas pagam 16% de imposto. Porém, se a empresa possuir mais de 10 funcionários, ela paga 14% de imposto.

Exemplo: uma empresa cuja renda foi 400000.00 e possui 25 funcionários, o imposto fica: $400000 * 14\% = \mathbf{56000.00}$

```

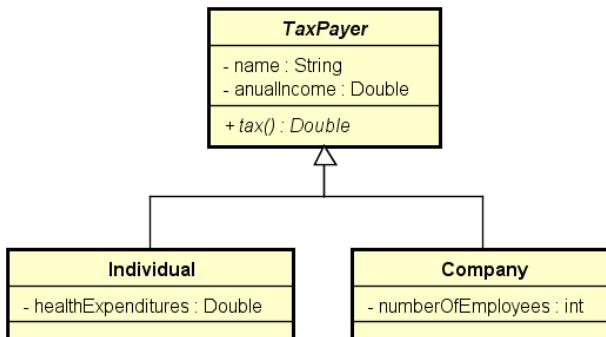
Enter the number of tax payers: 3
Tax payer #1 data:
Individual or company (i/c)? i
Name: Alex
Anual income: 50000.00
Health expenditures: 2000.00
Tax payer #2 data:
Individual or company (i/c)? c
Name: SoftTech
Anual income: 400000.00
Number of employees: 25
Tax payer #3 data:
Individual or company (i/c)? i
Name: Bob
Anual income: 120000.00
Health expenditures: 1000.00

TAXES PAID:
Alex: $ 11500.00
SoftTech: $ 56000.00
Bob: $ 29500.00

TOTAL TAXES: $ 97000.00

```

Correção



<https://github.com/acenelio/inheritance8-csharp>

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Tratamento de exceções

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

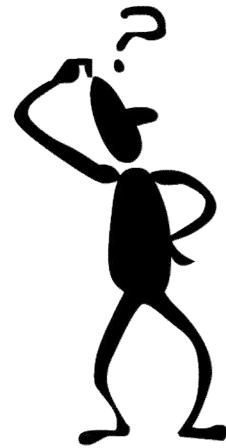
Discussão inicial sobre exceções

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

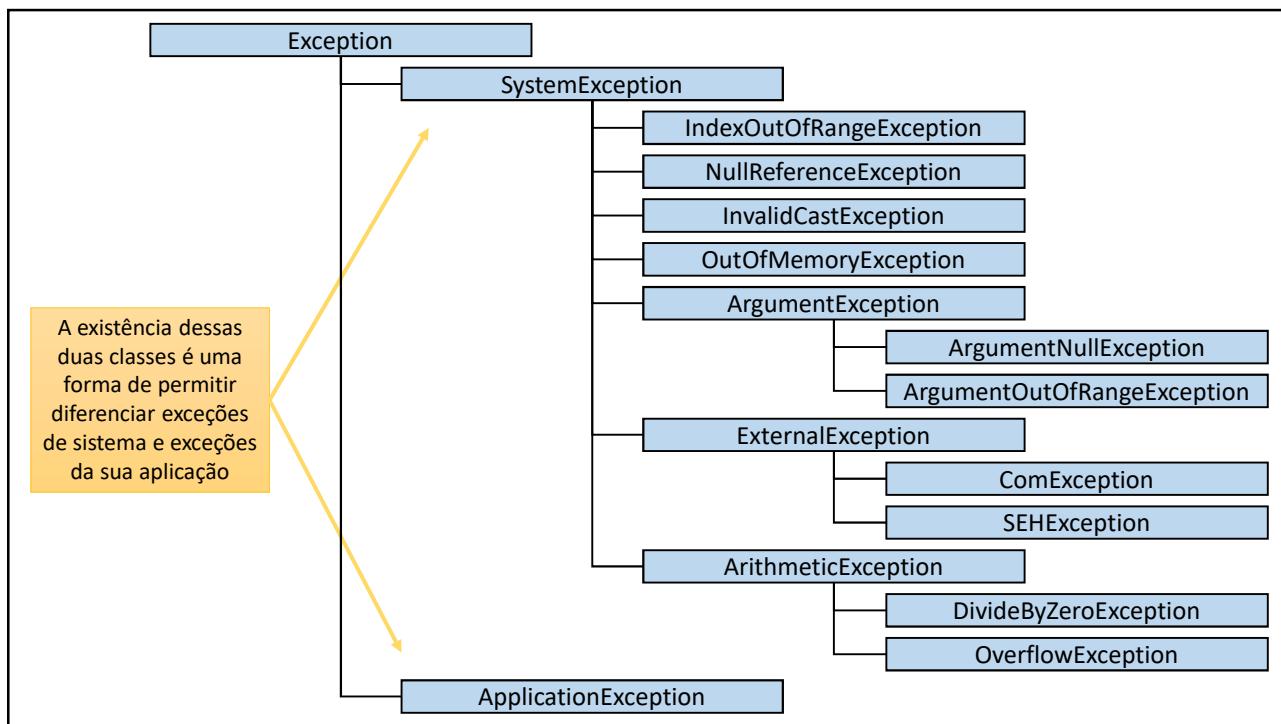
Atenção: pode ser um pouco difícil de entender vendo somente a teoria

Mas tudo ficará claro com os exemplos práticos nas próximas aulas
😊



Exceções

- Uma exceção é qualquer condição de erro ou comportamento inesperado encontrado por um programa **em execução**
- No .NET, uma exceção é um objeto herdado da classe `System.Exception`
- Quando lançada, uma exceção é propagada na pilha de chamadas de métodos em execução, até que seja capturada (tratada) ou o programa seja encerrado



Por que exceções?

- O modelo de tratamento de exceções permite que erros sejam tratados de forma consistente e flexível, usando boas práticas
- Vantagens:
 - Delega a lógica do erro para a classe / método responsável por conhecer as regras que podem ocasionar o erro
 - Trata de forma organizada (inclusive hierárquica) exceções de tipos diferentes
 - A exceção pode carregar dados quaisquer

Estrutura try-catch

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura try-catch

- Bloco try
 - Contém o código que representa a execução normal do trecho de código que **pode** acarretar em uma exceção
- Bloco catch
 - Contém o código a ser executado caso uma exceção ocorra
 - Deve ser especificado o tipo da exceção a ser tratada (upcasting é permitido)
- Demo

Sintaxe

```
try {  
}  
catch (ExceptionType e) {  
}  
catch (ExceptionType e) {  
}  
catch (ExceptionType e) {  
}  
}
```

Demo

```
using System;  
  
namespace Course {  
    class Program {  
        static void Main(string[] args) {  
  
            try {  
                int n1 = int.Parse(Console.ReadLine());  
                int n2 = int.Parse(Console.ReadLine());  
  
                int result = n1 / n2;  
                Console.WriteLine(result);  
            }  
            catch (DivideByZeroException) {  
                Console.WriteLine("Division by zero is not allowed");  
            }  
            catch (FormatException e) {  
                Console.WriteLine("Format error! " + e.Message);  
            }  
        }  
    }  
}
```

Bloco finally

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Bloco finally

- É um bloco que contém código a ser executado independentemente de ter ocorrido ou não uma exceção.
- Exemplo clássico: fechar um arquivo ou conexão de banco de dados ao final do processamento.

Sintaxe:

```
try {  
}  
catch (ExceptionType e) {  
}  
finally {  
}
```

```
using System;
using System.IO;

public class ProcessFile {
    public static void Main() {
        FileStream fs = null;
        try {
            fs = new FileStream(@"C:\temp\data.txt", FileMode.Open);
            StreamReader sr = new StreamReader(fs);
            string line = sr.ReadLine();
            Console.WriteLine(line);
        }
        catch (FileNotFoundException e) {
            Console.WriteLine(e.Message);
        }
        finally {
            if (fs != null)
                fs.Close();
        }
    }
}
```

Criando exceções personalizadas - PARTE 1

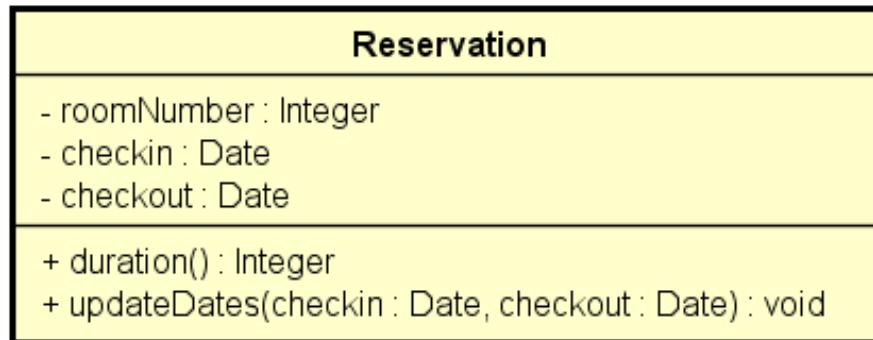
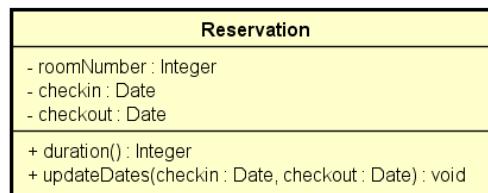
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema exemplo

Fazer um programa para ler os dados de uma reserva de hotel (número do quarto, data de entrada e data de saída) e mostrar os dados da reserva, inclusive sua duração em dias. Em seguida, ler novas datas de entrada e saída, atualizar a reserva, e mostrar novamente a reserva com os dados atualizados. O programa não deve aceitar dados inválidos para a reserva, conforme as seguintes regras:

- Alterações de reserva só podem ocorrer para datas futuras
- A data de saída deve ser maior que a data de entrada



Examples

Room number: **8021**

Check-in date (dd/MM/yyyy): **23/09/2019**

Check-out date (dd/MM/yyyy): **26/09/2019**

Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:

Check-in date (dd/MM/yyyy): **24/09/2019**

Check-out date (dd/MM/yyyy): **29/09/2019**

Reservation: Room 8021, check-in: 24/09/2019, check-out: 29/09/2019, 5 nights

Room number: **8021**

Check-in date (dd/MM/yyyy): **23/09/2019**

Check-out date (dd/MM/yyyy): **21/09/2019**

Error in reservation: Check-out date must be after check-in date

Examples

Room number: **8021**

Check-in date (dd/MM/yyyy): **23/09/2019**

Check-out date (dd/MM/yyyy): **26/09/2019**

Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:

Check-in date (dd/MM/yyyy): **24/09/2015**

Check-out date (dd/MM/yyyy): **29/09/2015**

Error in reservation: Reservation dates for update must be future dates

Room number: **8021**

Check-in date (dd/MM/yyyy): **23/09/2019**

Check-out date (dd/MM/yyyy): **26/09/2019**

Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:

Check-in date (dd/MM/yyyy): **24/09/2020**

Check-out date (dd/MM/yyyy): **22/09/2020**

Error in reservation: Check-out date must be after check-in date

Criando exceções personalizadas

- PARTE 2

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Solução do problema

- Solução 1 (muito ruim): lógica de validação no programa principal
 - Lógica de validação não delegada à reserva
- Solução 2 (ruim): método retornando string
 - A semântica da operação é prejudicada
 - Retornar string não tem nada a ver com atualização de reserva
 - E se a operação tivesse que retornar um string?
 - Ainda não é possível tratar exceções em construtores
 - A lógica fica estruturada em condicionais aninhadas

Criando exceções personalizadas

- PARTE 3

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Resumo da aula

- Cláusula throw: lança a exceção / "corta" o método
- O modelo de tratamento de exceções permite que erros sejam tratados de forma consistente e flexível, usando boas práticas
- Vantagens:
 - Lógica delegada
 - Construtores podem ter exceções
 - Código mais simples. Não há aninhamento de condicionais: a qualquer momento que uma exceção for disparada, a execução é interrompida e cai no bloco catch correspondente.
 - É possível capturar inclusive outras exceções do sistema

<https://github.com/acenelio/exceptions1-csharp>

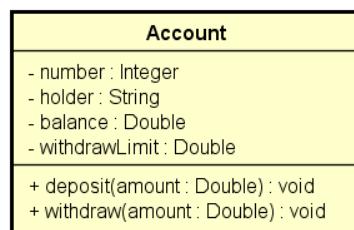
Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Exercício de fixação

Fazer um programa para ler os dados de uma conta bancária e depois realizar um saque nesta conta bancária, mostrando o novo saldo. Um saque não pode ocorrer ou se não houver saldo na conta, ou se o valor do saque for superior ao limite de saque da conta. Implemente a conta bancária conforme projeto abaixo:



Examples

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00
```

```
Enter amount for withdraw: 100.00
New balance: 400.00
```

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00
```

```
Enter amount for withdraw: 400.00
Withdraw error: The amount exceeds withdraw limit
```

Examples

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00
```

```
Enter amount for withdraw: 800.00
Withdraw error: The amount exceeds withdraw limit
```

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 200.00
Withdraw limit: 300.00
```

```
Enter amount for withdraw: 250.00
Withdraw error: Not enough balance
```

<https://github.com/acenelio/exceptions2-csharp>

Curso C# Completo

Capítulo: Trabalhando com arquivos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

File, FileInfo, IOException

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

File, FileInfo

- Namespace System.IO
- Realiza operações com arquivos (create, copy, delete, move, open, etc.) e ajuda na criação de objetos FileStream.
- File
 - static members (simples, mas realiza verificação de segurança para cada operação)
 - [https://msdn.microsoft.com/en-us/library/system.io.file\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.file(v=vs.110).aspx)
- FileInfo
 - instance members
 - [https://msdn.microsoft.com/en-us/library/system.io.fileinfo\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.fileinfo(v=vs.110).aspx)

IOException

Namespace System.IO

- IOException
 - DirectoryNotFoundException
 - DriveNotFoundException
 - EndOfStreamException
 - FileModeException
 - FileNotFoundException
 - PathTooLongException
 - PipeException

Demo File

```
using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {

            string sourcePath = @"c:\temp\file1.txt";
            string targetPath = @"c:\temp\file2.txt";

            try {
                File.Copy(sourcePath, targetPath);
                string[] lines = File.ReadAllLines(sourcePath);
                foreach (string line in lines) {
                    Console.WriteLine(line);
                }
            }
            catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Demo FileInfo

```
using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {

            string sourcePath = @"c:\temp\file1.txt";
            string targetPath = @"c:\temp\file2.txt";

            try {
                FileInfo fileInfo = new FileInfo(sourcePath);
                fileInfo.CopyTo(targetPath);
                string[] lines = File.ReadAllLines(sourcePath);
                foreach (string line in lines) {
                    Console.WriteLine(line);
                }
            }
            catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

FileStream, StreamReader

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

FileStream

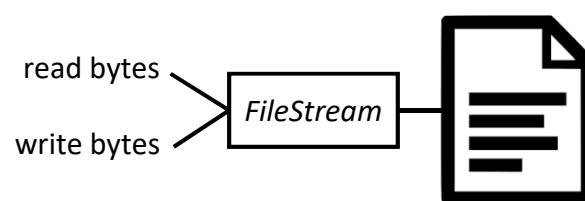
[https://msdn.microsoft.com/en-us/library/system.io.filestream\(v=vs.90\).aspx](https://msdn.microsoft.com/en-us/library/system.io.filestream(v=vs.90).aspx)

Disponibiliza uma stream associada a um arquivo, permitindo operações de leitura e escrita.

Suporte a dados binários.

Instanciação:

- Vários construtores
- File / FileInfo



StreamReader

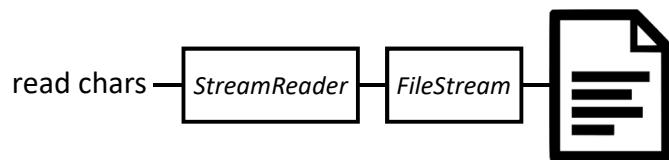
[https://msdn.microsoft.com/en-us/library/system.io.streamreader\(v=vs.90\).aspx](https://msdn.microsoft.com/en-us/library/system.io.streamreader(v=vs.90).aspx)

É uma stream capaz de ler caracteres a partir de uma stream binária (ex: FileStream).

Suporte a dados no formato de texto.

Instanciação:

- Vários construtores
- File / FileInfo



Demo 1

```

using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {

            string path = @"c:\temp\file1.txt";
            FileStream fs = null;
            StreamReader sr = null;
            try {
                fs = new FileStream(path, FileMode.Open); // File.OpenRead(path);
                sr = new StreamReader(fs);
                string line = sr.ReadLine();
                Console.WriteLine(line);
            }
            catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
            finally {
                if (sr != null) sr.Close();
                if (fs != null) fs.Close();
            }
        }
    }
}
  
```

Demo 2

```
using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {
            string path = @"c:\temp\file1.txt";

            StreamReader sr = null;
            try {
                sr = File.OpenText(path);
                while (!sr.EndOfStream) {
                    string line = sr.ReadLine();
                    Console.WriteLine(line);
                }
            }
            catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
            finally {
                if (sr != null) sr.Close();
            }
        }
    }
}
```

using block

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

using block

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-statement>

Sintaxe simplificada que garante que os objetos IDisposable serão fechados.

Objetos IDisposable NÃO são gerenciados pelo CLR. Eles precisam ser manualmente fechados.

Exemplos: Font, FileStream, StreamReader, StreamWriter

Demo 1

```
using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {
            string path = @"c:\temp\file1.txt";

            try {
                using (FileStream fs = new FileStream(path, FileMode.Open)) {
                    using (StreamReader sr = new StreamReader(fs)) {
                        string line = sr.ReadLine();
                        Console.WriteLine(line);
                    }
                }
            }
            catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Demo 2

```
using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {

            string path = @"c:\temp\file1.txt";

            try {
                using (StreamReader sr = File.OpenText(path)) {
                    while (!sr.EndOfStream) {
                        string line = sr.ReadLine();
                        Console.WriteLine(line);
                    }
                }
            } catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

StreamWriter

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

StreamWriter

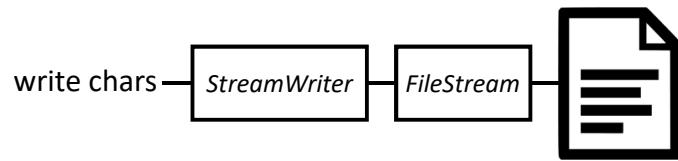
[https://msdn.microsoft.com/en-us/library/system.io.streamwriter\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.streamwriter(v=vs.110).aspx)

É uma stream capaz de escrever caracteres a partir de uma stream binária (ex: FileStream).

Suporte a dados no formato de texto.

Instantiation:

- Several constructors
- File / FileInfo
 - CreateText(path)
 - AppendText(String)



Demo

```

using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {

            string sourcePath = @"c:\temp\file1.txt";
            string targetPath = @"c:\temp\file2.txt";

            try {
                string[] lines = File.ReadAllLines(sourcePath);

                using (StreamWriter sw = File.AppendText(targetPath)) {
                    foreach (string line in lines) {
                        sw.WriteLine(line.ToUpper());
                    }
                }
            }
            catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
        }
    }
}
  
```

Directory, DirectoryInfo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Directory, DirectoryInfo

- Namespace System.IO
- Operações com pastas (create, enumerate, get files, etc.).
- Directory
 - static members (simple, but performs security check for each operation)
 - [https://msdn.microsoft.com/en-us/library/system.io.directory\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.directory(v=vs.110).aspx)
- DirectoryInfo
 - instance members
 - [https://msdn.microsoft.com/en-us/library/system.io.directoryinfo\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.directoryinfo(v=vs.110).aspx)

Demo

- Vamos listar as pastas a partir de uma pasta informada
- Vamos listar os arquivos a partir de uma pasta informada
- Vamos criar uma pasta

```
using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {
            string path = @"c:\temp\myfolder";

            try {
                var folders = Directory.EnumerateDirectories(path, "*.*", SearchOption.AllDirectories);
                Console.WriteLine("FOLDERS:");
                foreach (string s in folders) {
                    Console.WriteLine(s);
                }

                var files = Directory.EnumerateFiles(path, "*.*", SearchOption.AllDirectories);
                Console.WriteLine("FILES:");
                foreach (string s in files) {
                    Console.WriteLine(s);
                }
            }

            Directory.CreateDirectory(@"c:\temp\myfolder\newfolder");
        }
        catch (IOException e) {
            Console.WriteLine("An error occurred");
            Console.WriteLine(e.Message);
        }
    }
}
```

Path

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Path

- Namespace System.IO
- Realiza operações com strings que contém informações de arquivos ou pastas.
- [https://msdn.microsoft.com/en-us/library/system.io.path\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.path(v=vs.110).aspx)

Demo

```
using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {

            string path = @"c:\temp\myfolder\file1.txt";

            Console.WriteLine("DirectorySeparatorChar: " + Path.DirectorySeparatorChar);
            Console.WriteLine("PathSeparator: " + Path.PathSeparator);
            Console.WriteLine("GetDirectoryName: " + Path.GetDirectoryName(path));
            Console.WriteLine("GetFileName: " + Path.GetFileName(path));
            Console.WriteLine("GetExtension: " + Path.GetExtension(path));
            Console.WriteLine("GetFileNameWithoutExtension: " + Path.GetFileNameWithoutExtension(path));
            Console.WriteLine("GetFullPath: " + Path.GetFullPath(path));
            Console.WriteLine("GetTempPath: " + Path.GetTempPath());
        }
    }
}
```

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler o caminho de um arquivo .csv contendo os dados de itens vendidos. Cada item possui um nome, preço unitário e quantidade, separados por vírgula. Você deve gerar um novo arquivo chamado "summary.csv", localizado em uma subpasta chamada "out" a partir da pasta original do arquivo de origem, contendo apenas o nome e o valor total para aquele item (preço unitário multiplicado pela quantidade), conforme exemplo.

Example:

Source file:

```
TV LED,1290.99,1  
Video Game Chair,350.50,3  
Iphone X,900.00,2  
Samsung Galaxy 9,850.00,2
```

Output file (out/summary.csv):

```
TV LED,1290.99  
Video Game Chair,1051.50  
Iphone X,1800.00  
Samsung Galaxy 9,1700.00
```

<https://github.com/acenelio/files1-csharp>

Curso C# Completo

Capítulo: Generics, Set, Dictionary

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Introdução aos Generics

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Generics

- Generics permitem que **classes, interfaces e métodos** possam ser parametrizados por tipo. Seus benefícios são:
 - Reuso
 - Type safety
 - Performance
- Uso comum: coleções

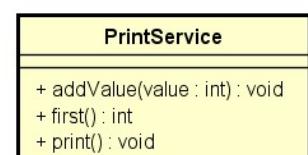
```
List<string> list = new List<string>();  
list.Add("Maria");  
string name = list[0];
```

Problema motivador 1 (reuso)

Deseja-se fazer um programa que leia um conjunto de N números inteiros (N de 1 a 10), e depois imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

```
How many values? 3  
10  
8  
23  
[10, 8, 23]  
First: 10
```

Criar um serviço de impressão:



Problema motivador 2 (type safety & performance)

Deseja-se fazer um programa que leia um conjunto de N números inteiros (N de 1 a 10), e depois imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

How many values? **3**

10

8

23

[10, 8, 23]

First: 10

Criar um serviço de impressão:

PrintService

+ addValue(value : object) : void
+ first() : object
+ print() : void

Solução com generics

Deseja-se fazer um programa que leia um conjunto de N números inteiros (N de 1 a 10), e depois imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

How many values? **3**

10

8

23

[10, 8, 23]

First: 10

Criar um serviço de impressão:

PrintService<T>

+ addValue(value : T) : void
+ first() : T
+ print() : void

<https://github.com/acenelio/generics1-csharp>

Restrições para generics

<http://educandoweb.com.br>

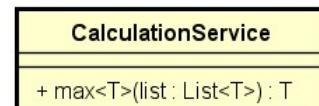
Prof. Dr. Nelio Alves

Problema

Uma empresa de consultoria deseja avaliar a performance de produtos, funcionários, dentre outras coisas. Um dos cálculos que ela precisa é encontrar o maior dentre um conjunto de elementos. Fazer um programa que leia um conjunto de N produtos, conforme exemplo, e depois mostre o mais caro deles.

```
Enter N: 3
Computer,890.50
IPhone X,910.00
Tablet,550.00
Max:
IPhone, 910.00
```

Criar um serviço de cálculo:



<https://github.com/acenelio/generics2-csharp>

Restrições possíveis

- <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/constraints-on-type-parameters>
- where T: struct
- where T : class
- where T : unmanaged
- where T : new()
- where T : <base type name>
- where T : U

GetHashCode e Equals

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

GetHashCode e Equals

- São operações da classe Object utilizadas para comparar se um objeto é igual a outro
- Equals: lento, resposta 100%
- GetHashCode: rápido, porém resposta positiva não é 100%
- Os tipos pré-definidos já possuem implementação para essas operações. Classes e structs personalizados precisam sobrepor-las.

Equals

Método que compara se o objeto é igual a outro, retornando true ou false.

```
string a = "Maria";
string b = "Alex";

Console.WriteLine(a.Equals(b));
```

GetHashCode

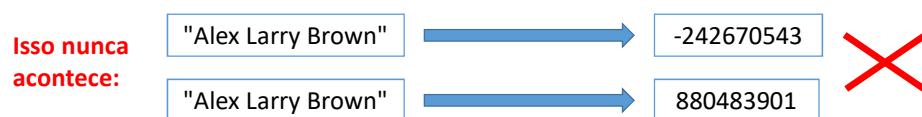
Método que retorna um número inteiro representando um código gerado a partir das informações do objeto

```
string a = "Maria";
string b = "Alex";

Console.WriteLine(a.GetHashCode());
Console.WriteLine(b.GetHashCode());
```

Regra de ouro do GetHashCode

- Se o código de dois objetos for diferente, então os dois objetos são diferentes



- Se o código de dois objetos for igual, **muito provavelmente** os objetos são iguais (pode haver colisão)

GetHashCode e Equals personalizados

```
class Client {  
    public string Name { get; set; }  
    public string Email { get; set; }  
}
```

HashSet<T> e SortedSet<T>

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

HashSet<T> e SortedSet<T>

- Representa um conjunto de elementos (similar ao da Álgebra)
 - Não admite repetições
 - Elementos não possuem posição
 - Acesso, inserção e remoção de elementos são rápidos
 - Oferece operações eficientes de conjunto: interseção, união, diferença.
- HashSet
 - [https://msdn.microsoft.com/en-us/library/bb359438\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/bb359438(v=vs.110).aspx)
- SortedSet
 - [https://msdn.microsoft.com/en-us/library/dd412070\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/dd412070(v=vs.110).aspx)

Diferenças

- HashSet
 - Armazenamento em tabela hash
 - Extremamente rápido: inserção, remoção e busca O(1)
 - A ordem dos elementos não é garantida
- SortedSet
 - Armazenamento em árvore
 - Rápido: inserção, remoção e busca O(log(n))
 - Os elementos são armazenados ordenadamente conforme implementação IComparer<T>

Alguns métodos importantes

- Add
- Clear
- Contains
- UnionWith(other) - operação união: adiciona no conjunto os elementos do outro conjunto, sem repetição
- IntersectWith(other) - operação interseção: remove do conjunto os elementos não contidos em other
- ExceptWith(other) - operação diferença: remove do conjunto os elementos contidos em other
- Remove(T)
- RemoveWhere(predicate)

Demo 1

```
using System;
using System.Collections.Generic;

namespace Course {
    class Program {

        static void Main(string[] args) {
            HashSet<string> set = new HashSet<string>();

            set.Add("TV");
            set.Add("Notebook");
            set.Add("Tablet");

            Console.WriteLine(set.Contains("Notebook"));

            foreach (String p in set) {
                Console.WriteLine(p);
            }
        }
    }
}
```

Demo 2

```
using System;
using System.Collections.Generic;

namespace Course {
    class Program {

        static void Main(string[] args) {
            SortedSet<int> a = new SortedSet<int>() { 0, 2, 4, 5, 6, 8, 10 };
            SortedSet<int> b = new SortedSet<int>() { 5, 6, 7, 8, 9, 10 };

            //union
            SortedSet<int> c = new SortedSet<int>(a);
            c.UnionWith(b);
            printCollection(c);

            //intersection
            SortedSet<int> d = new SortedSet<int>(a);
            d.IntersectWith(b);
            printCollection(d);

            //difference
            SortedSet<int> e = new SortedSet<int>(a);
            e.ExceptWith(b);
            printCollection(e);
        }

        static void printCollection<T>(IEnumerable<T> collection) {
            foreach(T obj in collection) {
                Console.Write(obj + " ");
            }
            Console.WriteLine();
        }
    }
}
```

Como as coleções Hash testam igualdade?

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Como as coleções Hash testam igualdade?

- Se GetHashCode e Equals estiverem implementados:
 - Primeiro GetHashCode. Se der igual, usa Equals para confirmar.
- Se GetHashCode e Equals **NÃO** estiverem implementados:
 - Tipos referência: compara as referências dos objetos
 - Tipos valor: comparar os valores dos atributos

```
namespace Course.Entities {
    struct Point {
        public int X { get; set; }
        public int Y { get; set; }

        public Point(int x, int y) : this() {
            X = x;
            Y = y;
        }
    }
}
```

```
namespace Course.Entities {
    class Product {

        public string Name { get; set; }
        public double Price { get; set; }

        public Product(string name, double price) {
            Name = name;
            Price = price;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using Course.Entities;

namespace Course {
    class Program {
        static void Main(string[] args) {

            HashSet<Product> a = new HashSet<Product>();
            a.Add(new Product("TV", 900.0));
            a.Add(new Product("Notebook", 1200.0));

            HashSet<Point> b = new HashSet<Point>();
            b.Add(new Point(3, 4));
            b.Add(new Point(5, 10));

            Product prod = new Product("Notebook", 1200.0);
            Console.WriteLine(a.Contains(prod));

            Point point = new Point(5, 10);
            Console.WriteLine(b.Contains(point));
        }
    }
}
```

Exercício resolvido (Set)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema exemplo

Um site de internet registra um log de acessos dos usuários. Um registro de log consiste no nome de usuário e o instante em que o usuário acessou o site no padrão ISO 8601, separados por espaço, conforme exemplo. Fazer um programa que leia o log de acessos a partir de um arquivo, e daí informe quantos usuários distintos acessaram o site.

Example

input file:

```
amanda 2020-08-26T20:45:08
alex86 2020-08-26T21:49:37
bobbrown 2020-08-27T03:19:13
amanda 2020-08-27T08:11:00
jeniffer3 2020-08-27T09:19:24
alex86 2020-08-27T22:39:52
amanda 2020-08-28T07:42:19
```

Execution:

```
Enter file full path: c:\temp\in.txt
Total users: 4
```

<https://github.com/acenelio/set1-csharp>

```
using System;
using System.IO;

namespace Course {
    class Program {
        static void Main(string[] args) {

            Console.Write("Enter file full path: ");
            string path = Console.ReadLine();

            try {
                using (StreamReader sr = File.OpenText(path)) {
                    while (!sr.EndOfStream) {
                        string line = sr.ReadLine();
                        Console.WriteLine(line);
                    }
                }
            } catch (IOException e) {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Exercício proposto (Set)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Em um portal de cursos online, cada usuário possui um código único, representado por um número inteiro.

Cada instrutor do portal pode ter vários cursos, sendo que um mesmo aluno pode se matricular em quantos cursos quiser. Assim, o número total de alunos de um instrutor não é simplesmente a soma dos alunos de todos os cursos que ele possui, pois pode haver alunos repetidos em mais de um curso.

O instrutor Alex possui três cursos A, B e C, e deseja saber seu número total de alunos.

Seu programa deve ler os alunos dos cursos A, B e C do instrutor Alex, depois mostrar a quantidade total e alunos dele, conforme exemplo.

<https://github.com/acenelio/set2-csharp>

Example:

How many students for course A? **3**

21

35

22

How many students for course B? **2**

21

50

How many students for course C? **3**

42

35

13

Total students: 6

Dictionary e SortedDictionary

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Dictionary< TKey, TValue >

- É uma coleção de pares chave / valor
 - Não admite repetições do objeto chave
 - Os elementos são indexados pelo objeto chave (não possuem posição)
 - Acesso, inserção e remoção de elementos são rápidos
- Uso comum: cookies, local storage, qualquer modelo chave-valor
- Dictionary
 - [https://msdn.microsoft.com/en-us/library/xfhwa508\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/xfhwa508(v=vs.110).aspx)
- SortedDictionary
 - [https://msdn.microsoft.com/en-us/library/f7fta44c\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/f7fta44c(v=vs.110).aspx)

Diferenças

- Dictionary
 - Armazenamento em tabela hash
 - Extremamente rápido: inserção, remoção e busca O(1)
 - A ordem dos elementos não é garantida
- SortedDictionary
 - Armazenamento em árvore
 - Rápido: inserção, remoção e busca O(log(n))
 - Os elementos são armazenados ordenadamente conforme implementação IComparer<T>

Alguns métodos importantes

- dictionary[key] - acessa o elemento pela chave informada
- Add(key, value) - adiciona elemento (exceção em caso de repetição)
- Clear() - esvazia a coleção
- Count - quantidade de elementos
- ContainsKey(key) - verifica se a dada chave existe
- ContainsValue(value) - verifica se o dado valor existe
- Remove(key) - remove um elemento pela chave

Demo

```
using System;
using System.Collections.Generic;

namespace Course {
    class Program {
        static void Main(string[] args) {
            Dictionary<string, string> cookies = new Dictionary<string, string>();
            cookies["user"] = "maria";
            cookies["email"] = "maria@gmail.com";
            cookies["phone"] = "99771122";
            cookies["phone"] = "99771133";

            Console.WriteLine(cookies["email"]);

            cookies.Remove("email");

            Console.WriteLine("Phone number: " + cookies["phone"]);

            if (cookies.ContainsKey("email")) {
                Console.WriteLine("Email: " + cookies["email"]);
            }
            else {
                Console.WriteLine("There is not 'email' key");
            }

            Console.WriteLine("Size: " + cookies.Count);

            Console.WriteLine("ALL COOKIES:");
            foreach (KeyValuePair<string, string> item in cookies) {
                Console.WriteLine(item.Key + ": " + item.Value);
            }
        }
    }
}
```

Exercício proposto (Dictionary)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Na contagem de votos de uma eleição, são gerados vários registros de votação contendo o nome do candidato e a quantidade de votos (formato .csv) que ele obteve em uma urna de votação. Você deve fazer um programa para ler os registros de votação a partir de um arquivo, e daí gerar um relatório consolidado com os totais de cada candidato.

Input file example:

```
Alex Blue,15
Maria Green,22
Bob Brown,21
Alex Blue,30
Bob Brown,15
Maria Green,27
Maria Green,22
Bob Brown,25
Alex Blue,31
```

Execution:

```
Enter file full path: c:\temp\in.txt
Alex Blue: 76
Maria Green: 71
Bob Brown: 61
```

Solução do exercício

<https://github.com/acenelio/dictionary1-csharp>

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Tópicos Especiais em C# - PARTE 2

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Extension methods

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Extension methods

- São métodos que estendem a funcionalidade de um tipo, sem precisar alterar o código fonte deste tipo
- Como fazer um extension method?
 - Criar uma classe estática
 - Na classe, criar um método estático
 - O primeiro parâmetro do método deverá ter o prefixo `this`, seguido da declaração de um parâmetro do tipo que se deseja estender. Esta será uma referência para o próprio objeto.

Demo 1

Vamos criar um *extension method* chamado "`ElapsedTime()`" no struct `DateTime` para apresentar um objeto `DateTime` na forma de tempo decorrido, podendo ser em horas (se menor que 24h) ou em dias caso contrário. Por exemplo:

```
DateTime dt = new DateTime(2018, 11, 16, 8, 10, 45);  
Console.WriteLine(dt.ElapsedTime());
```

"4.5 hours"

"3.2 days"

```
using System.Globalization;

namespace System
{
    static class DateTimeExtensions
    {
        public static string ElapsedTime(this DateTime thisObj)
        {
            TimeSpan duration = DateTime.Now.Subtract(thisObj);

            if (duration.TotalHours < 24.0)
            {
                return duration.TotalHours.ToString("F1", CultureInfo.InvariantCulture) + " hours";
            }
            else
            {
                return duration.TotalDays.ToString("F1", CultureInfo.InvariantCulture) + " days";
            }
        }
    }
}
```

Demo 2

Vamos criar um *extension method* chamado "Cut(int)" na classe String para receber um valor inteiro como parâmetro e gerar um recorte do string original daquele tamanho. Por exemplo:

```
String s1 = "Good morning dear students!";
Console.WriteLine(s1.Cut(10));
```

"Good morni..."

```
namespace System
{
    static class StringExtensions
    {
        public static string Cut(this string thisObj, int count)
        {
            if (thisObj.Length <= count)
            {
                return thisObj;
            }
            else
            {
                return thisObj.Substring(0, count) + "...";
            }
        }
    }
}
```

Curso C# Completo Programação Orientada a Objetos + Projetos

Capítulo: Interfaces

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Interfaces

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Interface

Interface é um tipo que define um conjunto de operações que uma classe (ou struct) deve implementar.

A interface estabelece um **contrato** que a classe (ou struct) deve cumprir.

```
interface IShape {  
    double Area();  
    double Perimeter();  
}
```

Pra quê interfaces?

- Para criar sistemas com **baixo acoplamento** e **flexíveis**.

Problema exemplo

Uma locadora brasileira de carros cobra um valor por hora para locações de até 12 horas. Porém, se a duração da locação ultrapassar 12 horas, a locação será cobrada com base em um valor diário. Além do valor da locação, é acrescido no preço o valor do imposto conforme regras do país que, no caso do Brasil, é 20% para valores até 100.00, ou 15% para valores acima de 100.00. Fazer um programa que lê os dados da locação (modelo do carro, instante inicial e final da locação), bem como o valor por hora e o valor diário de locação. O programa deve então gerar a nota de pagamento (contendo valor da locação, valor do imposto e valor total do pagamento) e informar os dados na tela. Veja os exemplos.

Example 1:

```

Enter rental data
Car model: Civic
Pickup (dd/MM/yyyy hh:mm): 25/06/2018 10:30
Return (dd/MM/yyyy hh:mm): 25/06/2018 14:40
Enter price per hour: 10.00
Enter price per day: 130.00
INVOICE:
Basic payment: 50.00
Tax: 10.00
Total payment: 60.00

```

Calculations:

$$\text{Duration} = (25/06/2018 14:40) - (25/06/2018 10:30) = 4:10 = 5 \text{ hours}$$

$$\text{Basic payment} = 5 * 10 = 50$$

$$\text{Tax} = 50 * 20\% = 50 * 0.2 = 10$$

Example 2:

```

Enter rental data
Car model: Civic
Pickup (dd/MM/yyyy hh:mm): 25/06/2018 10:30
Return (dd/MM/yyyy hh:mm): 27/06/2018 11:40
Enter price per hour: 10.00
Enter price per day: 130.00
INVOICE:
Basic payment: 390.00
Tax: 58.50
Total payment: 448.50

```

Calculations:

$$\text{Duration} = (27/06/2018 11:40) - (25/06/2018 10:30) = 2 \text{ days} + 1:10 = 3 \text{ days}$$

$$\text{Basic payment} = 3 * 130 = 390$$

$$\text{Tax} = 390 * 15\% = 390 * 0.15 = 58.50$$

Solução do problema

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

(recordando - cap. 6)

Views

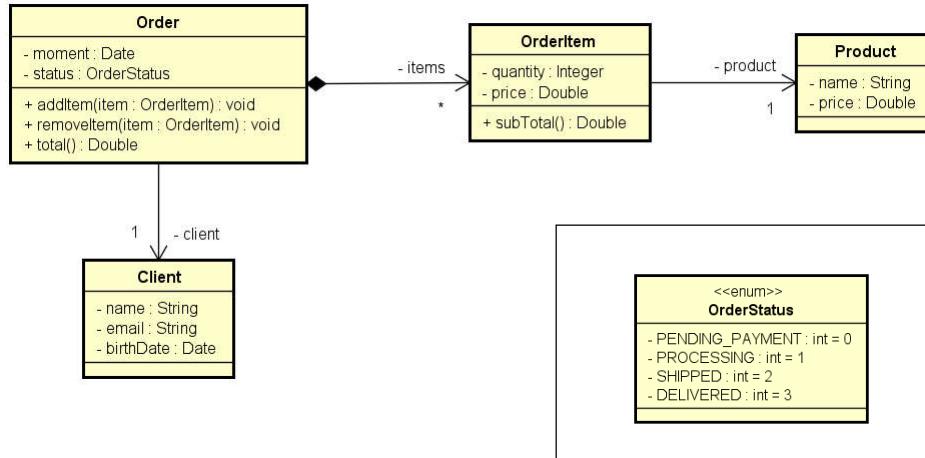
Controllers

Entities

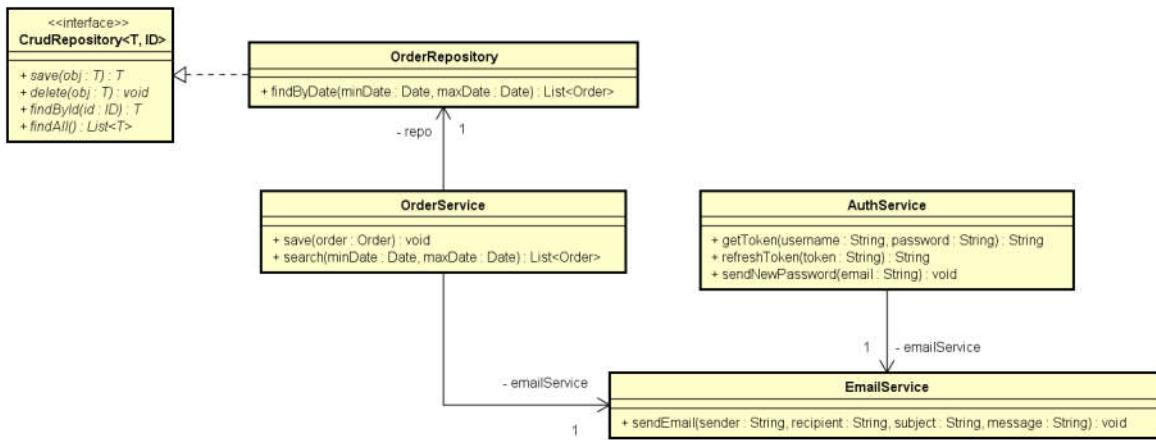
Services

Repositories

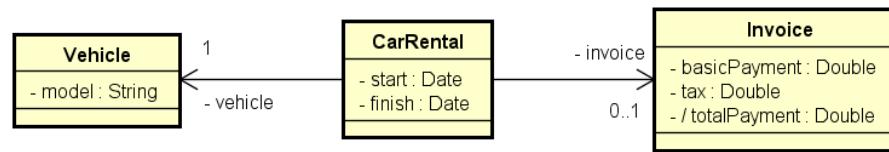
Entities



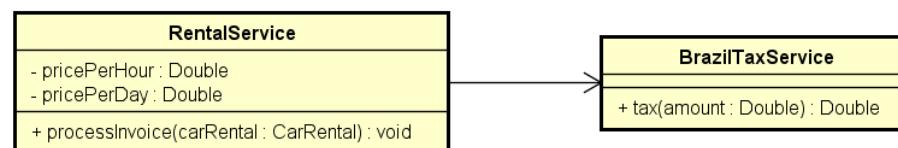
Services



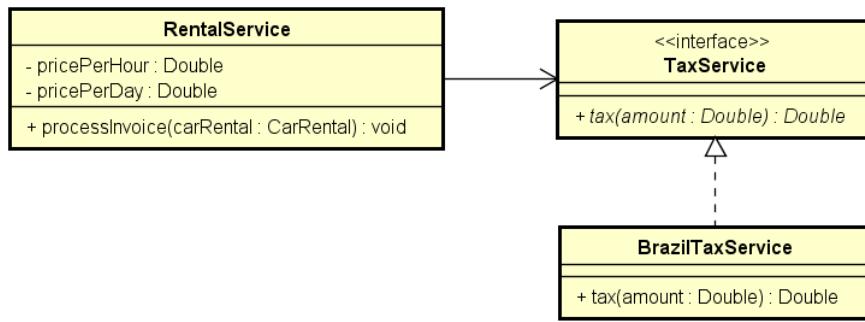
Domain layer design



Service layer design (no interface)



Service layer design



Projeto no Github

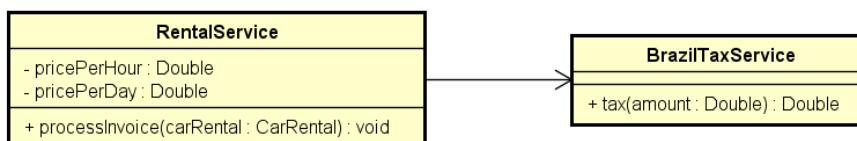
<https://github.com/acenelio/interfaces1-csharp>

Inversão de controle, Injeção de dependência

<http://educandoweb.com.br>

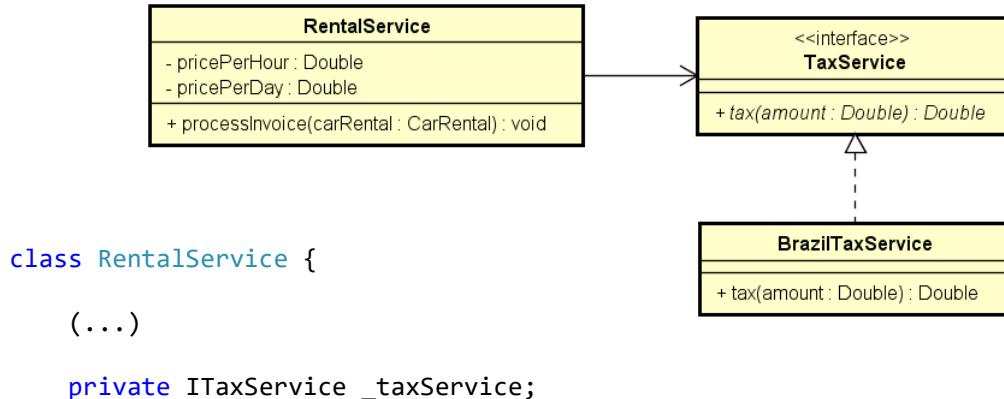
Prof. Dr. Nelio Alves

- Acoplamento forte
- A classe RentalService conhece a dependência concreta
- Se a classe concreta mudar, é preciso mudar a classe RentalService



```
class RentalService {
    ...
    private BrazilTaxService _brazilTaxService = new BrazilTaxService();
```

- Acoplamento fraco
- A classe RentalService não conhece a dependência concreta
- Se a classe concreta mudar, a classe RentalService não muda nada



Injeção de dependência por meio de construtor

```

class Program {
    static void Main(string[] args) {
        (...)

        RentalService rentalService = new RentalService(hour, day, new BrazilTaxService());
    }
}

class RentalService {
    private ITaxService _taxService;

    public RentalService(double pricePerHour, double pricePerDay, ITaxService taxService) {
        PricePerHour = pricePerHour;
        PricePerDay = pricePerDay;
        _taxService = taxService;
    }
}

```

A red arrow points from the line `new BrazilTaxService()` in the `Program` class code to the constructor of the `RentalService` class, indicating the injection of the dependency through the constructor.

Inversão de controle

- **Inversão de controle**

Padrão de desenvolvimento que consiste em retirar da classe a responsabilidade de instanciar suas dependências.

- **Injeção de dependência**

É uma forma de realizar a inversão de controle: um componente externo instancia a dependência, que é então injetada no objeto "pai". Pode ser implementada de várias formas:

- Construtor
- Objeto de instanciação (builder / factory)
- Container / framework

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Uma empresa deseja automatizar o processamento de seus contratos. O processamento de um contrato consiste em gerar as parcelas a serem pagas para aquele contrato, com base no número de meses desejado.

A empresa utiliza um serviço de pagamento online para realizar o pagamento das parcelas. Os serviços de pagamento online tipicamente cobram um juro mensal, bem como uma taxa por pagamento. Por enquanto, o serviço contratado pela empresa é o do Paypal, que aplica **juros simples** de 1% a cada parcela, mais uma **taxa** de pagamento de 2%.

Fazer um programa para ler os dados de um contrato (número do contrato, data do contrato, e valor total do contrato). Em seguida, o programa deve ler o número de meses para parcelamento do contrato, e daí gerar os registros de parcelas a serem pagas (data e valor), sendo a primeira parcela a ser paga um mês após a data do contrato, a segunda parcela dois meses após o contrato e assim por diante. Mostrar os dados das parcelas na tela.

Veja exemplo na próxima página.

Example:

```
Enter contract data
Number: 8028
Date (dd/MM/yyyy): 25/06/2018
Contract value: 600.00
Enter number of installments: 3
Installments:
25/07/2018 - 206.04
25/08/2018 - 208.08
25/09/2018 - 210.12
```

Calculations (1% monthly simple interest + 2% payment fee):

Quota #1:

$$\begin{aligned} 200 + 1\% * 1 &= 202 \\ 202 + 2\% &= 206.04 \end{aligned}$$

Quota #2:

$$\begin{aligned} 200 + 1\% * 2 &= 204 \\ 204 + 2\% &= 208.08 \end{aligned}$$

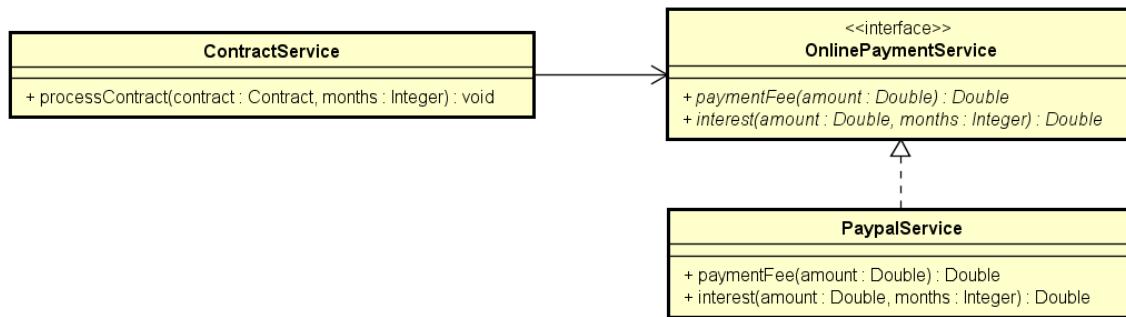
Quota #3:

$$\begin{aligned} 200 + 1\% * 3 &= 206 \\ 206 + 2\% &= 210.12 \end{aligned}$$

Domain layer design (entities)



Service layer design



Repositório Github

<https://github.com/acenelio/interfaces4-csharp>

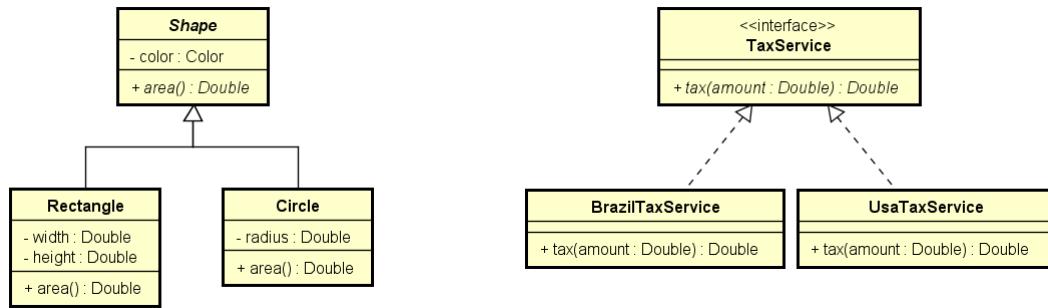
Herdar vs. cumprir contrato

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

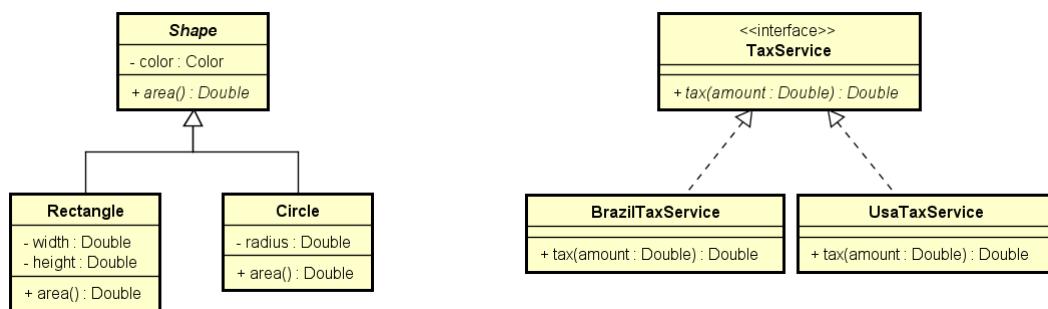
Aspectos em comum entre herança e interfaces

- Relação é-um
- Generalização/especialização
- Polimorfismo

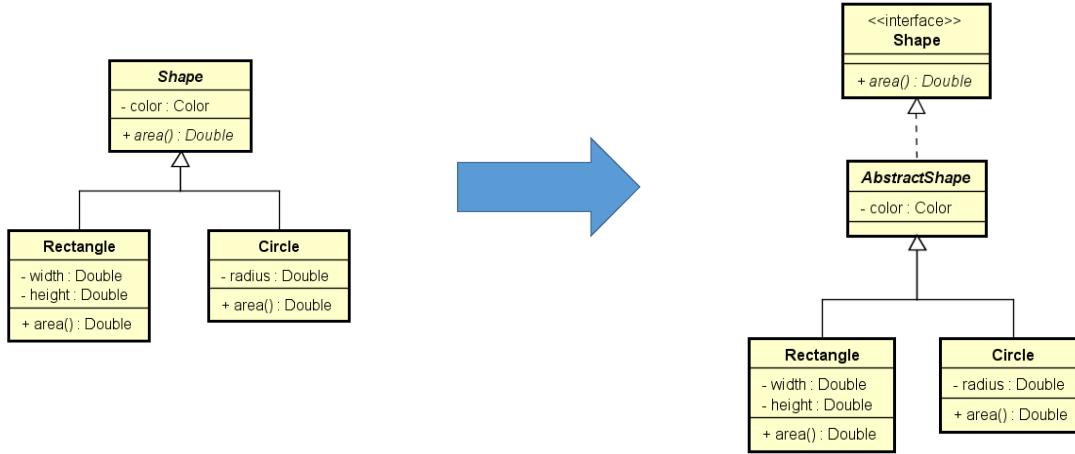


Diferença fundamental

- Herança => reuso
- Interface => contrato a ser cumprido



E se eu precisar implementar Shape como interface, porém também quiser definir uma estrutura comum reutilizável para todas figuras?



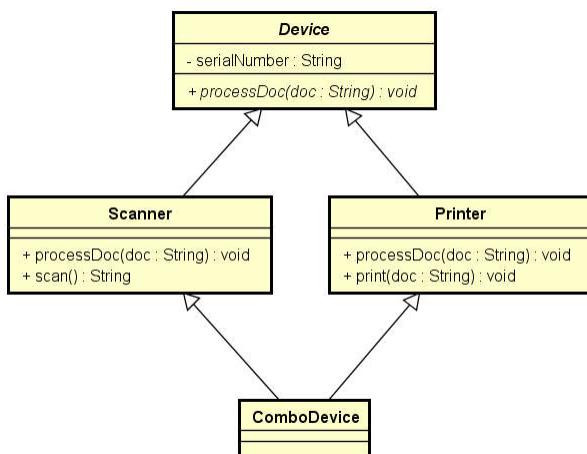
<https://github.com/acenelio/interfaces2-csharp>

Herança múltipla e o problema do diamante

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

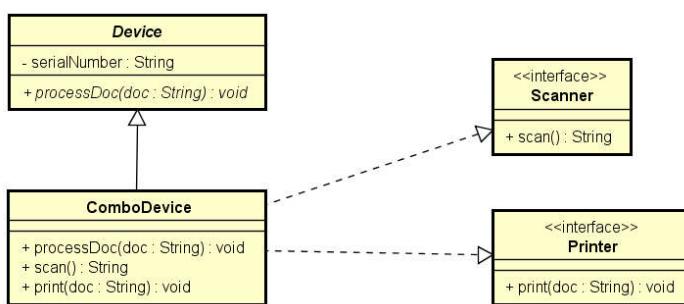
Problema do diamante



A herança múltipla pode gerar o problema do diamante: uma ambiguidade causada pela existência do mesmo método em mais de uma superclasse.

Herança múltipla não é permitida na maioria das linguagens!

Porém, uma classe (ou struct) pode implementar mais de uma interface



ATENÇÃO:

Isso NÃO é herança múltipla, pois NÃO HÁ REUSO na relação entre ComboDevice e as interfaces Scanner e Printer.

ComboDevice não herda, mas sim implementa as interfaces (cumpre o contrato).

<https://github.com/acenelio/interfaces3-csharp>

Interface IComparable

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Interface Comparable

[https://msdn.microsoft.com/en-us/library/system.icomparable\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.icomparable(v=vs.110).aspx)

```
public interface IComparable {
    int CompareTo(object other);
}
```

Problema motivador

Faça um programa para ler um arquivo contendo nomes de pessoas (um nome por linha), armazenando-os em uma lista. Depois, ordenar os dados dessa lista e mostra-los ordenadamente na tela. Nota: o caminho do arquivo pode ser informado "hardcode".

```
Maria Brown
Alex Green
Bob Grey
Anna White
Alex Black
Eduardo Rose
Willian Red
Marta Blue
Alex Brown
```

```
using System;
using System.IO;
using System.Collections.Generic;

namespace Course {
    class Program {
        static void Main(string[] args) {

            string path = @"c:\temp\in.txt";

            try {
                using (StreamReader sr = File.OpenText(path)) {
                    List<string> list = new List<string>();
                    while (!sr.EndOfStream) {
                        list.Add(sr.ReadLine());
                    }
                    list.Sort();
                    foreach (string str in list) {
                        Console.WriteLine(str);
                    }
                }
            }
            catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Outro problema

Faça um programa para ler um arquivo contendo funcionários (nome e salário) no formato .csv, armazenando-os em uma lista. Depois, ordenar a lista por nome e mostrar o resultado na tela. Nota: o caminho do arquivo pode ser informado "hardcode".

```
Maria Brown,4300.00
Alex Green,3100.00
Bob Grey,3100.00
Anna White,3500.00
Alex Black,2450.00
Eduardo Rose,4390.00
Willian Red,2900.00
Marta Blue,6100.00
Alex Brown,5000.00
```

```
using System.Globalization;

namespace Course {
    class Employee {

        public string Name { get; set; }
        public double Salary { get; set; }

        public Employee(string csvEmployee) {
            string[] vect = csvEmployee.Split(',');
            Name = vect[0];
            Salary = double.Parse(vect[1], CultureInfo.InvariantCulture);
        }

        public override string ToString() {
            return Name + ", " + Salary.ToString("F2", CultureInfo.InvariantCulture);
        }
    }
}
```

Interface IComparable

```
namespace System {
    public interface IComparable {
        int CompareTo(object obj);
    }
}
```

```
Console.WriteLine("maria".CompareTo("alex"));
Console.WriteLine("alex".CompareTo("maria"));
Console.WriteLine("maria".CompareTo("maria"));
```

Output:

```
1
-1
0
```

Value	Meaning
Less than zero	The current instance precedes the object specified by the <code>CompareTo</code> method in the sort order.
Zero	This current instance occurs in the same position in the sort order as the object specified by the <code>CompareTo</code> method.
Greater than zero	This current instance follows the object specified by the <code>CompareTo</code> method in the sort order.

```
using System;
using System.Globalization;

namespace Course {
    class Employee : IComparable {

        public string Name { get; set; }
        public double Salary { get; set; }

        public Employee(string csvEmployee) {
            string[] vect = csvEmployee.Split(',');
            Name = vect[0];
            Salary = double.Parse(vect[1], CultureInfo.InvariantCulture);
        }

        public override string ToString() {
            return Name + ", " + Salary.ToString("F2", CultureInfo.InvariantCulture);
        }

        public int CompareTo(object obj) {
            if (!(obj is Employee)) {
                throw new ArgumentException("Comparing error: argument is not an Employee");
            }
            Employee other = obj as Employee;
            return Name.CompareTo(other.Name);
        }
    }
}
```

Curso C# Completo

Programação Orientada a Objetos + Projetos

Capítulo: Expressões lambda, delegates e LINQ

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

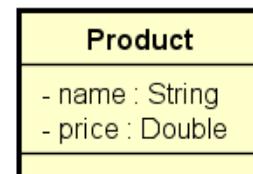
Uma experiência com Comparison<T>

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema

- Suponha uma classe Product com os atributos name e price.
Suponha que precisamos ordenar uma lista de objetos Product.
- Podemos implementar a comparação de produtos por meio da implementação da interface IComparable<Product>
- Entretanto, desta forma nossa classe Product não fica fechada para alteração: se o critério de comparação mudar, precisaremos alterar a classe Product.
- Podemos então usar outra sobrecarga do método "Sort" da classe List:
`public void Sort(Comparison<T> comparison)`



Comparison<T> (System)

[https://msdn.microsoft.com/en-us/library/tfakywvh\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/tfakywvh(v=vs.110).aspx)

`public delegate int Comparison<in T>(T x, T y);`

Método Sort com Comparison<T> da classe List:

<https://msdn.microsoft.com/en-us/library/w56d4y5z%28v=vs.110%29.aspx>

Resumo da aula

```
public void Sort(Comparison<T> comparison)
```

- Referência simples de método como parâmetro
- Referência de método atribuído a uma variável tipo delegate
- Expressão lambda atribuída a uma variável tipo delegate
- Expressão lambda inline

<https://github.com/acenelio/lambda1-csharp>

Programação funcional e cálculo lambda

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Paradigmas de programação

- **Imperativo** (C, Pascal, Fortran, Cobol)
- **Orientado a objetos** (C++, Object Pascal, Java (< 8), C# (< 3))
- **Funcional** (Haskell, Closure, Clean, Erlang)
- **Lógico** (Prolog)
- **Multiparadigma** (JavaScript, Java (8+), C# (3+), Ruby, Python, Go)

Paradigma funcional de programação

Baseado no formalismo matemático Cálculo Lambda (Church 1930)

	Programação Imperativa	Programação Funcional
Como se descreve algo a ser computado (*)	comandos ("como" - imperativa)	expressões ("o quê" - declarativa)
Funções possuem transparência referencial (ausência de efeitos colaterais)	fraco	forte
Objetos imutáveis (*)	raro	comum
Funções são objetos de primeira ordem	não	sim
Expressividade / código conciso	baixa	alta
Inferência de tipos	raro	comum
Execução tardia (lazy)	raro	comum

Transparência referencial

Uma função possui transparência referencial se seu resultado for sempre o mesmo para os mesmos dados de entrada. Benefícios: simplicidade e previsibilidade.

```
using System;

namespace Course {
    class Program {

        public static int globalValue = 3;

        static void Main(string[] args) {
            int[] vect = new int[] { 3, 4, 5 };
            ChangeOddValues(vect);
            Console.WriteLine(string.Join(" ", vect));
        }

        public static void ChangeOddValues(int[] numbers) {
            for (int i = 0; i < numbers.Length; i++) {
                if (numbers[i] % 2 != 0) {
                    numbers[i] += globalValue;
                }
            }
        }
    }
}
```



Exemplo de função que não é referencialmente transparente

Funções são objetos de primeira ordem (ou primeira classe)

Isso significa que funções podem, por exemplo, serem passadas como parâmetros de métodos, bem como retornadas como resultado de métodos.

```
class Program {

    static int CompareProducts(Product p1, Product p2) {
        return p1.Name.ToUpper().CompareTo(p2.Name.ToUpper());
    }

    static void Main(string[] args) {
        List<Product> list = new List<Product>();

        list.Add(new Product("TV", 900.00));
        list.Add(new Product("Notebook", 1200.00));
        list.Add(new Product("Tablet", 450.00));

        list.Sort(CompareProducts);

        (...)
```

Inferência de tipos

```
List<Product> list = new List<Product>();

list.Add(new Product("TV", 900.00));
list.Add(new Product("Notebook", 1200.00));
list.Add(new Product("Tablet", 450.00));

list.Sort((p1, p2) => p1.Name.ToUpper().CompareTo(p2.Name.ToUpper()));

foreach (Product p in list) {
    Console.WriteLine(p);
}
```

Expressividade / "como" vs. "o quê"

```
int sum = 0;
foreach (int x in list) {
    sum += x;
}
```

VS.

```
int sum = list.Aggregate(0, (x, y) => x + y);
```

O que são "expressões lambda"?

Em programação funcional, expressão lambda corresponde a uma função anônima de primeira classe.

```
class Program {  
    static int CompareProducts(Product p1, Product p2) {  
        return p1.Name.ToUpper().CompareTo(p2.Name.ToUpper());  
    }  
  
    static void Main(string[] args) {  
        (...)  
        list.Sort(CompareProducts);  
        list.Sort((p1, p2) => p1.Name.ToUpper().CompareTo(p2.Name.ToUpper()));  
        (...)  
    }  
}
```

Resumo da aula

	Programação Imperativa	Programação Funcional
Como se descreve algo a ser computado (*)	comandos ("como" - imperativa)	expressões ("o quê" - declarativa)
Funções possuem transparência referencial (ausência de efeitos colaterais)	fraco	forte
Objetos imutáveis (*)	raro	comum
Funções são objetos de primeira ordem	não	sim
Expressividade / código conciso	baixa	alta
Tipagem dinâmica / inferência de tipos	raro	comum
Execução tardia (lazy)	raro	comum

Cálculo Lambda = formalismo matemático base da programação funcional

Expressão lambda = função anônima de primeira classe

Introdução a delegates

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Delegates

- <https://docs.microsoft.com/en-us/dotnet/standard/delegates-lambdas>
- É uma referência (com type safety) para um ou mais métodos
 - É um tipo referência
- Usos comuns:
 - Comunicação entre objetos de forma flexível e extensível (eventos / callbacks)
 - Parametrização de operações por métodos (programação funcional)

Delegates pré-definidos

- Action
- Func
- Predicate

Demo

```
namespace Course.Services {
    class CalculationService {

        public static double Max(double x, double y) {
            return (x > y) ? x : y;
        }

        public static double Sum(double x, double y) {
            return x + y;
        }

        public static double Square(double x) {
            return x * x;
        }
    }
}
```

Demo

```
using System;
using Course.Services;

namespace Course {

    delegate double BinaryNumericOperation(double n1, double n2);

    class Program {
        static void Main(string[] args) {
            double a = 10;
            double b = 12;

            // BinaryNumericOperation op = CalculationService.Sum;
            BinaryNumericOperation op = new BinaryNumericOperation(CalculationService.Sum);

            // double result = op(a, b);
            double result = op.Invoke(a, b);
            Console.WriteLine(result);
        }
    }
}
```

Multicast delegates

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Multicast delegates

- Delegates que guardam a referência para mais de um método
- Para adicionar uma referência, pode-se usar o operador `+=`
- A chamada `Invoke` (ou sintaxe reduzida) executa todos os métodos na ordem em que foram adicionados
- Seu uso faz sentido para métodos `void`

Demo

```
using System;

namespace Course.Services {
    class CalculationService {

        public static void ShowMax(double x, double y) {
            double max = (x > y) ? x : y;
            Console.WriteLine(max);
        }

        public static void ShowSum(double x, double y) {
            double sum = x + y;
            Console.WriteLine(sum);
        }
    }
}
```

```
using System;
using Course.Services;

namespace Course {

    delegate void BinaryNumericOperation(double n1, double n2);

    class Program {
        static void Main(string[] args) {
            double a = 10;
            double b = 12;

            BinaryNumericOperation op = CalculationService.ShowSum;
            op += CalculationService.ShowMax;
            op(a, b);
        }
    }
}
```

Predicate (exemplo com RemoveAll)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Predicate (System)

- Representa um método que recebe um objeto do tipo T e retorna um valor booleano

- <https://msdn.microsoft.com/en-us/library/bfcke1bz%28v=vs.110%29.aspx>

```
public delegate bool Predicate<in T>(T obj);
```

Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, remova da lista somente aqueles cujo preço mínimo seja 100.

```
List<Product> list = new List<Product>();  
  
list.Add(new Product("Tv", 900.00));  
list.Add(new Product("Mouse", 50.00));  
list.Add(new Product("Tablet", 350.50));  
list.Add(new Product("HD Case", 80.90));
```

<https://github.com/acenelio/lambda2-csharp>

Action (exemplo com ForEach)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Action (System)

- Representa um método void que recebe zero ou mais argumentos
 - <https://msdn.microsoft.com/en-us/library/system.action%28v=vs.110%29.aspx>

```
public delegate void Action();
public delegate void Action<in T>(T obj);
public delegate void Action<in T1, in T2>(T1 arg1, T2 arg2);
public delegate void Action<in T1, in T2, in T3>(T1 arg1, T2 arg2, T3 arg3);
(16 sobrecargas)
```

Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, aumente o preço dos produtos em 10%.

```
List<Product> list = new List<Product>();  
  
list.Add(new Product("Tv", 900.00));  
list.Add(new Product("Mouse", 50.00));  
list.Add(new Product("Tablet", 350.50));  
list.Add(new Product("HD Case", 80.90));
```

<https://github.com/acenelio/lambda3-csharp>

Func (exemplo com Select)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Func (System)

- Representa um método que recebe zero ou mais argumentos, e retorna um valor

- <https://msdn.microsoft.com/en-us/library/bb534960%28v=vs.110%29.aspx>

```
public delegate TResult Func<out TResult>();  
public delegate TResult Func<in T, out TResult>(T obj);  
public delegate TResult Func<in T1, in T2, out TResult>(T1 arg1, T2 arg2);  
public delegate TResult Func<in T1, in T2, in T3, out TResult>(T1 arg1, T2  
arg2, T3 arg3);
```

(16 sobrecargas)

Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, gere uma nova lista contendo os nomes dos produtos em caixa alta.

```
List<Product> list = new List<Product>();  
  
list.Add(new Product("Tv", 900.00));  
list.Add(new Product("Mouse", 50.00));  
list.Add(new Product("Tablet", 350.50));  
list.Add(new Product("HD Case", 80.90));
```

<https://github.com/acenelio/lambda4-csharp>

Nota sobre a função Select

- A função "Select" (pertencente ao LINQ) é uma função que aplica uma função a todos elementos de uma coleção, gerando assim uma nova coleção (do tipo IEnumerable).

```
List<int> numbers = new List<int> { 2, 3, 4 };
IEnumerable<int> newList = numbers.Select(x => 2 * x);
Console.WriteLine(string.Join(" ", newList));
```

4 6 8

Criando funções que recebem funções como argumento

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Recordando

- removeAll(Predicate)
- ForEach(Action)
- Select(Func)

Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, calcule a soma dos preços somente dos produtos cujo nome começa com "T".

```
List<Product> list = new List<Product>();  
  
list.Add(new Product("Tv", 900.00));  
list.Add(new Product("Mouse", 50.00));  
list.Add(new Product("Tablet", 350.50));  
list.Add(new Product("HD Case", 80.90));
```



1250.50

<https://github.com/acenelio/lambda5-csharp>

Introdução ao LINQ

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

LINQ - Language Integrated Query

- É um conjunto de tecnologias baseadas na integração de funcionalidades de consulta diretamente na linguagem C#
 - Operações chamadas diretamente a partir das coleções
 - Consultas são objetos de primeira classe
 - Suporte do compilador e IntelliSense da IDE
- Namespace: System.Linq
- Possui diversas operações de consulta, cujos parâmetros tipicamente são expressões lambda ou expressões de sintaxe similar à SQL
- Referência:
 - <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/index>

Três passos

- Criar um data source (coleção, array, recurso de E/S, etc.)
- Definir a query
- Executar a query (foreach ou alguma operação terminal)



Demo

```
// Specify the data source.  
int[] numbers = new int[] { 2, 3, 4, 5 };  
  
// Define the query expression.  
IQueryable<int> result = numbers  
    .Where(x => x % 2 == 0)  
    .Select(x => 10 * x);  
  
// Execute the query.  
foreach (int x in result) {  
    Console.WriteLine(x);  
}
```

Operações do LINQ / Referências

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Operações do LINQ

- **Filtering:** Where, OfType
- **Sorting:** OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse
- **Set:** Distinct, Except, Intersect, Union
- **Quantification:** All, Any, Contains
- **Projection:** Select, SelectMany
- **Partition:** Skip, Take
- **Join:** Join, GroupJoin
- **Grouping:** GroupBy
- **Generational:** Empty
- **Equality:** SequenceEquals
- **Element:** ElementAt, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault
- **Conversions:** AsEnumerable, AsQueryable
- **Concatenation:** Concat
- **Aggregation:** Aggregate, Average, Count, LongCount, Max, Min, Sum

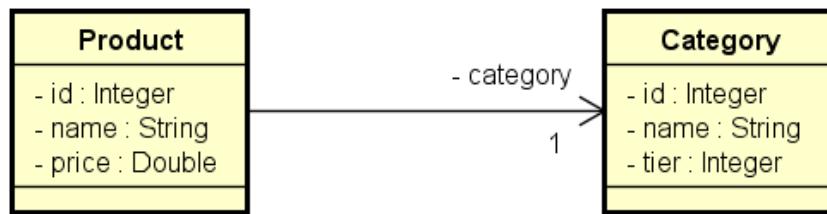
Referências

- <https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b/view/SamplePack/1?sortBy=Popularity>
- <https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b/view/SamplePack/2?sortBy=Popularity>
- <https://odetocode.com/articles/739.aspx>

Demo: LINQ com Lambda

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves



Resumo da aula

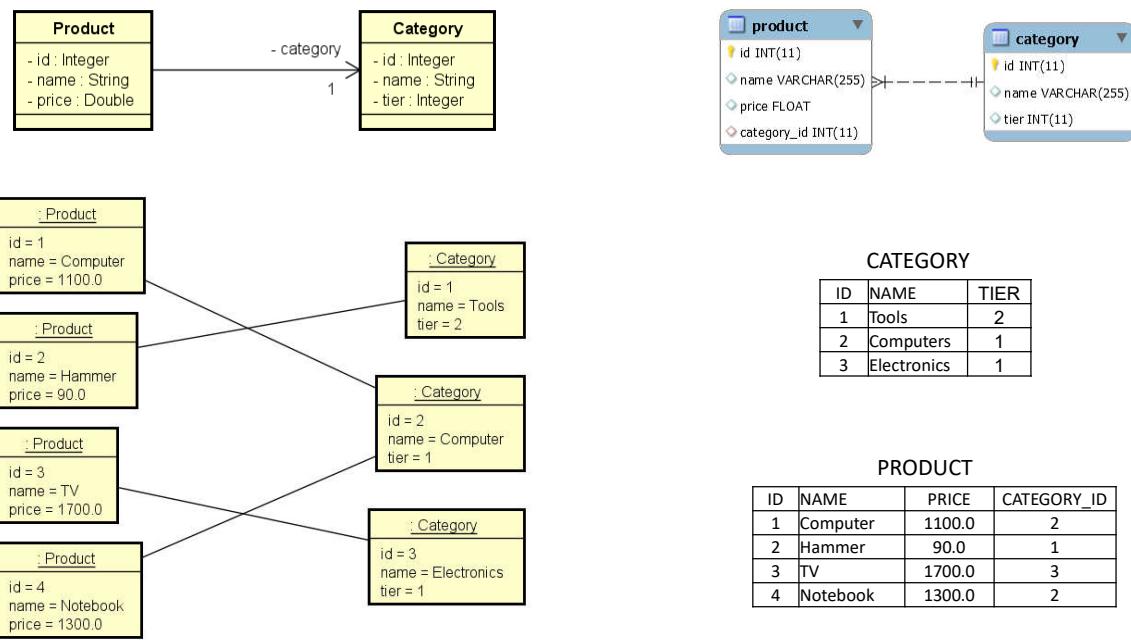
- Where (operação "filter" / "restrição")
- Select (operação "map" / "projeção")
- OrderBy, OrderByDescending, ThenBy, ThenByDescending
- Skip, Take
- First, FirstOrDefault Last, LastOrDefault, Single, SingleOrDefault
- Max, Min, Count, Sum, Average, Aggregate (operação "reduce")
- GroupBy

<https://github.com/acenelio/linq-demo1>

Nivelamento: Álgebra Relacional e SQL

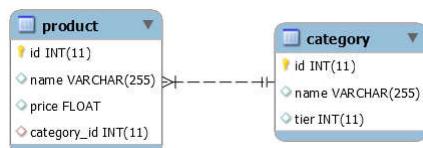
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves



Operações básicas da álgebra relacional

- Restrição
- Projeção
- Produto cartesiano
- Junção (produto cartesiano + restrição de chaves correspondentes)



Operação "produto cartesiano":

```
SELECT *  
FROM PRODUCT, CATEGORY
```

CATEGORY

ID	NAME	TIER
1	Tools	2
2	Computers	1
3	Electronics	1

PRODUCT

ID	NAME	PRICE	CATEGORY_ID
1	Computer	1100.0	2
2	Hammer	90.0	1
3	TV	1700.0	3
4	Notebook	1300.0	2

ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
1	Computer	1100.0	2	1	Tools	2
2	Hammer	90.0	1	1	Tools	2
3	TV	1700.0	3	1	Tools	2
4	Notebook	1300.0	2	1	Tools	2
1	Computer	1100.0	2	2	Computers	1
2	Hammer	90.0	1	2	Computers	1
3	TV	1700.0	3	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1
1	Computer	1100.0	2	3	Electronics	1
2	Hammer	90.0	1	3	Electronics	1
3	TV	1700.0	3	3	Electronics	1
4	Notebook	1300.0	2	3	Electronics	1

Operação "junção":

```
SELECT *
FROM PRODUCT, CATEGORY
WHERE
    PRODUCT.CATEGORY_ID = CATEGORY.ID
```

```
SELECT *
FROM PRODUCT
INNER JOIN CATEGORY cat
ON PRODUCT.CATEGORY_ID = cat.ID
```

ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
1	Computer	1100.0	2	1	Tools	2
2	Hammer	90.0	1	1	Tools	2
3	TV	1700.0	3	1	Tools	2
4	Notebook	1300.0	2	1	Tools	2
1	Computer	1100.0	2	2	Computers	1
2	Hammer	90.0	1	2	Computers	1
3	TV	1700.0	3	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1
1	Computer	1100.0	2	3	Electronics	1
2	Hammer	90.0	1	3	Electronics	1
3	TV	1700.0	3	3	Electronics	1
4	Notebook	1300.0	2	3	Electronics	1

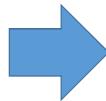


ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
2	Hammer	90.0	1	1	Tools	2
1	Computer	1100.0	2	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1
3	TV	1700.0	3	3	Electronics	1

Operação "restrição":

```
SELECT *
FROM PRODUCT
INNER JOIN CATEGORY cat ON PRODUCT.CATEGORY_ID = cat.ID
WHERE CATEGORY.NAME = 'Computers'
```

ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
1	Computer	1100.0	2	1	Tools	2
2	Hammer	90.0	1	1	Tools	2
3	TV	1700.0	3	1	Tools	2
4	Notebook	1300.0	2	1	Tools	2
1	Computer	1100.0	2	2	Computers	1
2	Hammer	90.0	1	2	Computers	1
3	TV	1700.0	3	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1
1	Computer	1100.0	2	3	Electronics	1
2	Hammer	90.0	1	3	Electronics	1
3	TV	1700.0	3	3	Electronics	1
4	Notebook	1300.0	2	3	Electronics	1



ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
2	Hammer	90.0	1	1	Tools	2
1	Computer	1100.0	2	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1
3	TV	1700.0	3	3	Electronics	1



ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
1	Computer	1100.0	2	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1

Operação "projeção":

```
SELECT PRODUCT.*, CATEGORY.NAME  
FROM PRODUCT  
INNER JOIN CATEGORY cat ON PRODUCT.CATEGORY_ID = cat.ID  
WHERE CATEGORY.NAME = 'Computers'
```

ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
1	Computer	1100.0	2	1	Tools	2
2	Hammer	90.0	1	1	Tools	2
3	TV	1700.0	3	1	Tools	2
4	Notebook	1300.0	2	1	Tools	2
1	Computer	1100.0	2	2	Computers	1
2	Hammer	90.0	1	2	Computers	1
3	TV	1700.0	3	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1
1	Computer	1100.0	2	3	Electronics	1
2	Hammer	90.0	1	3	Electronics	1
3	TV	1700.0	3	3	Electronics	1
4	Notebook	1300.0	2	3	Electronics	1



ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
2	Hammer	90.0	1	1	Tools	2
1	Computer	1100.0	2	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1
3	TV	1700.0	3	3	Electronics	1



ID	NAME	PRICE	CATEGORY_ID	ID	NAME	TIER
1	Computer	1100.0	2	2	Computers	1
4	Notebook	1300.0	2	2	Computers	1



ID	NAME	PRICE	CATEGORY_ID	NAME
1	Computer	1100.0	2	Computers
4	Notebook	1300.0	2	Computers

Demo: LINQ com notação similar a SQL

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Código fonte

- <https://github.com/acenelio/linq-demo2>

Exercício resolvido

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler um conjunto de produtos a partir de um arquivo em formato .csv (suponha que exista pelo menos um produto).

Em seguida mostrar o preço médio dos produtos. Depois, mostrar os nomes, em ordem decrescente, dos produtos que possuem preço inferior ao preço médio.

Veja exemplo na próxima página.

<https://github.com/acenelio/lambda6-csharp>

Input file:

```
Tv,900.00
Mouse,50.00
Tablet,350.50
HD Case,80.90
Computer,850.00
Monitor,290.00
```

Execution:

```
Enter full file path: c:\temp\in.txt
Average price: 420.23
Tablet
Mouse
Monitor
HD Case
```

<https://github.com/acenelio/lambda6-csharp>

Exercício de fixação

<http://educandoweb.com.br>

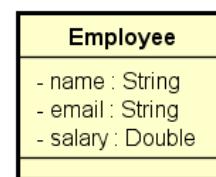
Prof. Dr. Nelio Alves

Fazer um programa para ler os dados (nome, email e salário) de funcionários a partir de um arquivo em formato .csv.

Em seguida mostrar, em ordem alfabética, o email dos funcionários cujo salário seja superior a um dado valor fornecido pelo usuário.

Mostrar também a soma dos salários dos funcionários cujo nome começa com a letra 'M'.

Veja exemplo na próxima página.



<https://github.com/acenelio/lambda7-csharp>

Input file:

```
Maria,maria@gmail.com,3200.00
Alex,alex@gmail.com,1900.00
Marco,marco@gmail.com,1700.00
Bob,bob@gmail.com,3500.00
Anna,anna@gmail.com,2800.00
```

Execution:

```
Enter full file path: c:\temp\in.txt
Enter salary: 2000.00
Email of people whose salary is more than 2000.00:
anna@gmail.com
bob@gmail.com
maria@gmail.com
Sum of salary of people whose name starts with 'M': 4900.00
```

<https://github.com/acenelio/lambda7-csharp>

Curso: C# COMPLETO - Programação Orientada a Objetos + Projetos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Projeto: GUI Web com ASP.NET Core

Objetivo geral:

- Introduzir o aluno ao desenvolvimento de aplicações web com ASP.NET Core MVC
- Permitir que o aluno conheça os fundamentos e a utilização do framework, de modo que ele possa depois prosseguir estudando as especificidades que desejar

PROJETO NO GITHUB:

<https://github.com/acenelio/workshop-asp-net-core-mvc>

Visão geral do ASP.NET Core MVC

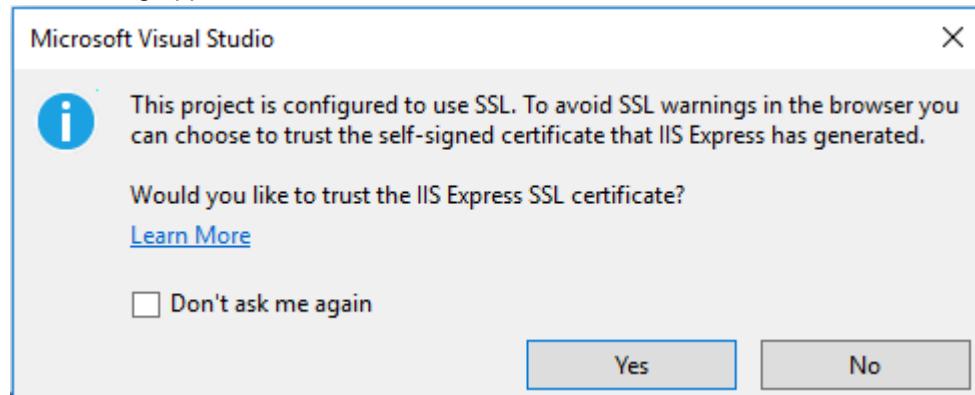
- É um framework para criação de aplicações web
- Criado pela Microsoft e comunidade
- Open source
- Roda tanto no .NET Framework quanto no .NET Core
- O framework trabalha com uma estrutura bem definida, incluindo:
 - Controllers
 - Views
 - Models
 - View Models
- <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview>

Project creation

Checklist:

- File -> New -> Project -> Visual C# -> Web -> ASP.NET Core Web Application
 - Create directory for solution
 - Create new Git repository
 - Web Application (Model-View-Controller)
 - (NO) authentication
 - (NO) Enable Docker Support
 - Configure for HTTPS
- Observe project folder and commits
- Run project
 - With debug: F5
 - Without debug: CTRL+F5
 - Live reloading
 - It's possible to stop IIS manually
- Create remote Git repository and push project

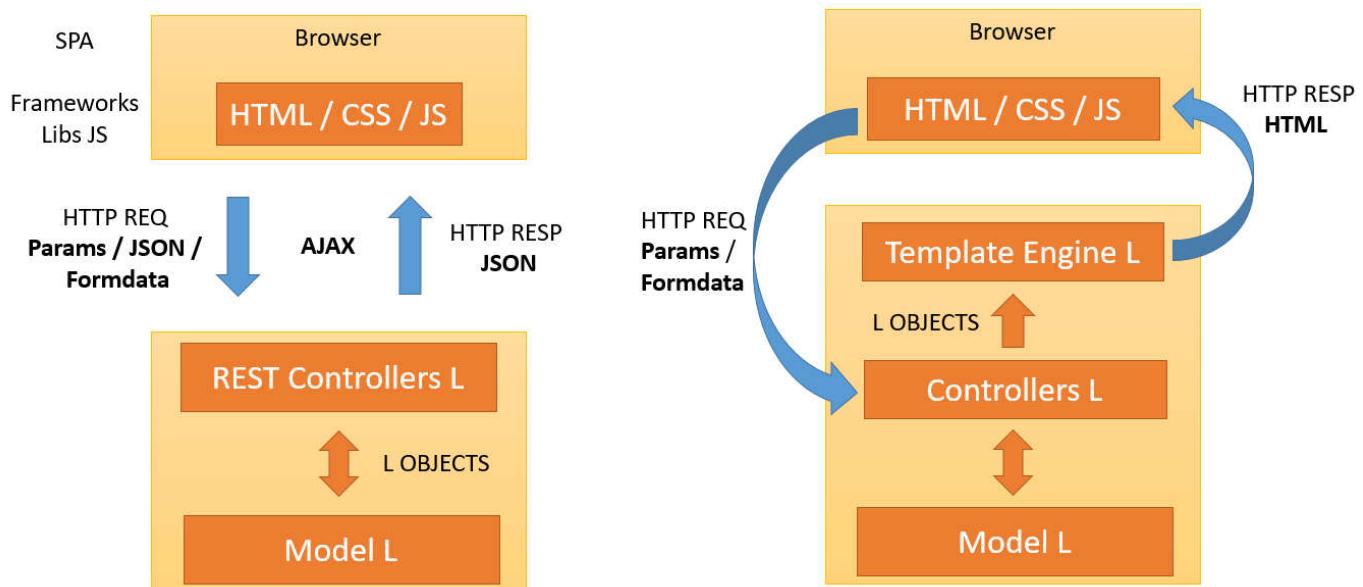
If this dialog appears:



Yes -> Install Certificate -> Yes

Refresher: Web MVC applications with template engine

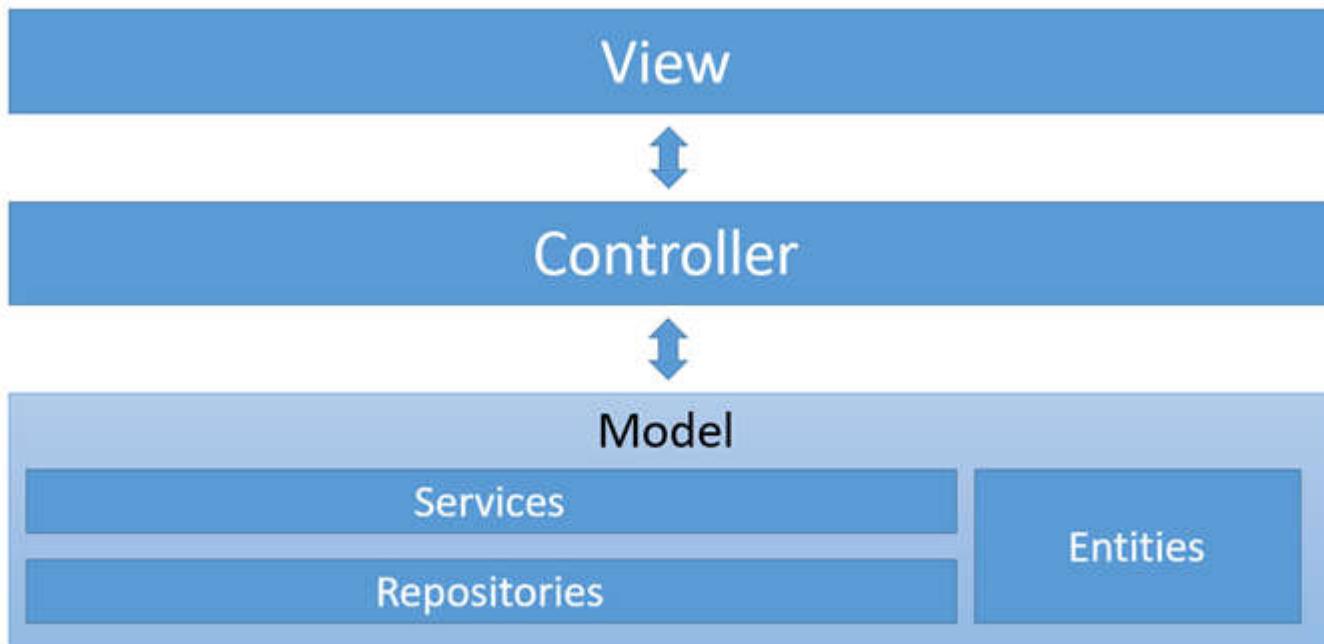
Web Services vs. Template Engine



Responsibility of each MVC part:

- **Model:** domain entities structure and their transformations (domain model)
 - Entities
 - Services (related to entities)
 - Repositories (persistent data access)
- **Controllers:** receives user interactions and treat them
- **Views:** defines structure and behaviour of user interface

General architecture:



Project structure

Checklist:

- wwwroot: application resources (css, imagens, etc.)
- Controllers: application's MVC controllers
- Models: entities and "view models"
- Views: pages (notice naming conventions against controllers)
 - Shared: views used for more than one controller
- appsettings.json: external resources configuration (logging, connection strings, etc.)
- Program.cs: entry point
- Startup.cs: app configuration

First controller and Razor pages tests

Checklist:

- Route pattern: Controller / Ação / Id
 - Each controller method is mapped to an action
- Natural Templates
- C# block in Razor Page: @{ }
- ViewData dictionary
- Tag Helpers in Razor Pages. Examples: asp-controller and asp-action
- IActionResult

Type	Method builder
ViewResult	View
PartialViewResult	PartialView
ContentResult	Content
RedirectResult	Redirect
RedirectToRouteResult	RedirectToAction Ex: RedirectToAction("Index", "Home", new { page = 1, sortBy = price })
JsonResult	Json
FileResult	File
HttpNotFoundResult	HttpNotFound
EmptyResult	-

First Model-Controller-View - Department

Checklist:

- Create new folder ViewModels e move ErrorViewModel (including namespace)
 - CTRL+SHIFT+B to fix references
- Create class Models/Department
- Create controller: right button Controllers -> Add -> Controller -> MVC Controller Empty
 - Name: DepartmentsController (**PLURAL**)
 - Instantiate a List<Department> and return it as View method parameter
- Create new folder Views/Departments (**PLURAL**)
- Create view: right button Views/Departments -> Add -> View
 - View name: Index
 - Template: List
 - Model class: Department
 - Change Title to Departments
 - Notice:
 - @model definition
 - intellisense for model
 - Helper methods
 - @foreach block

Deleting Department view and controller

Checklist:

- Delete controller
- Delete folder Views/Departments

CRUD Scaffolding

Checklist:

- Right button Controllers -> Add -> New Scaffolded Item
 - MVC controllers with views, using Entity Framework
 - Model class: Department
 - Data context class: + and accept the name
 - Views (options): all three
 - Controller name: DepartmentsController

MySQL adaptation and first migration

Note: we're using CODE-FIRST workflow

Checklist:

- Em appsettings.json, set connection string:
 - "server=localhost;userid=developer;password=1234567;database=saleswebmvccappdb"
- Em Startup.cs, fix DbContext definition for dependency injection system:

```
services.AddDbContext<SalesWebMvcAppContext>(options =>
    options.UseMySql(Configuration.GetConnectionString("SalesWebMvcContext"), builder =>
builder.MigrationsAssembly("SalesWebMvc")));
```

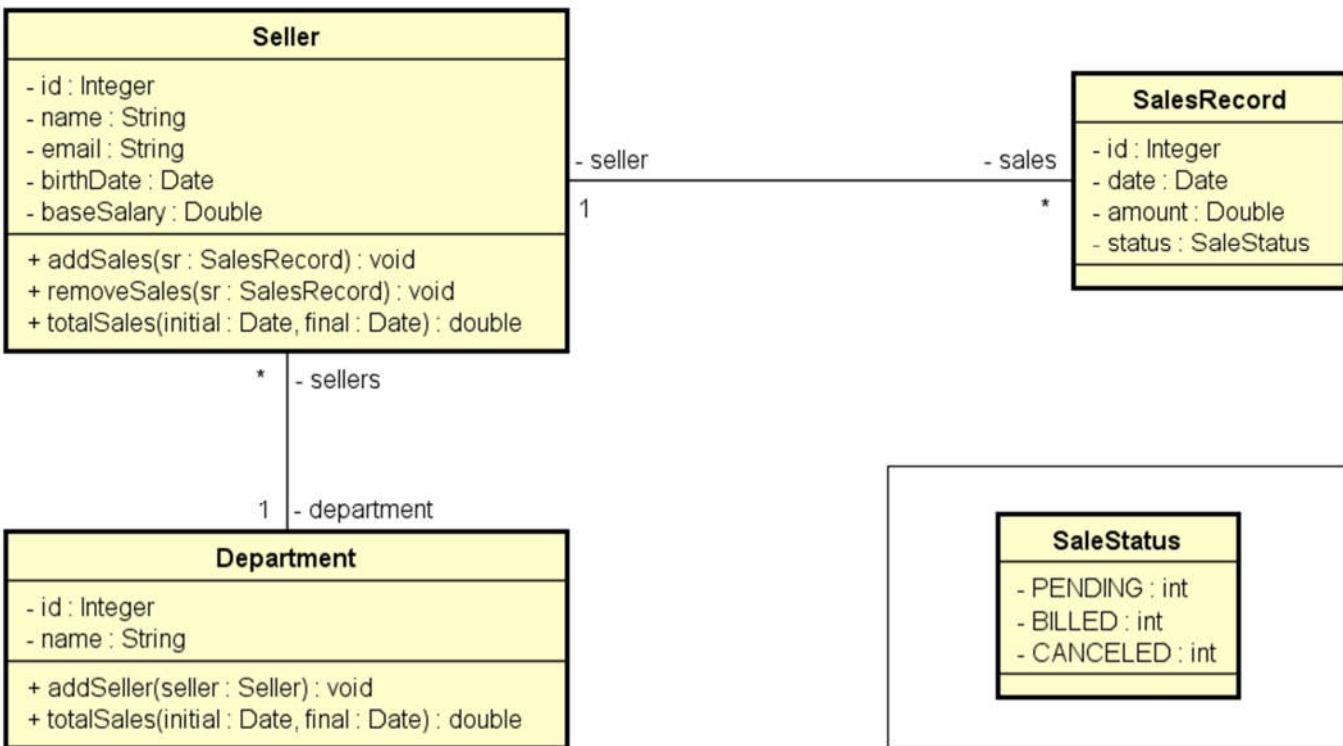
- Install MySQL provider:
 - Open NuGet Package Manager Console
 - Install-Package Pomelo.EntityFrameworkCore.MySql
- Stop IIS
- CTRL+SHIFT+B
- Start MySQL server:
 - Control Panel -> Administrative Tools -> Services
- Start MySQL Workbench
- Package Manager Console -> create first Migration:
 - Add-Migration Initial
 - Update-Database
- Check database in MySQL Workbench
- Test app: CTRL+F5

Changing theme

Checklist:

- Go to: <http://bootswatch.com/3> (check Bootstrap version)
- Choose a theme
- Download bootstrap.css
 - Suggestion: rename to bootstrap-name.css
 - Save file to wwwroot/lib/bootstrap/dist/css (paste it inside Visual Studio)
- Open _Layout.cshtml
 - Update bootstrap reference

Other entities and second migration



Checklist:

- Implement domain model
 - Basic attributes
 - Association (let's use **ICollection**, which matches List, HashSet, etc. - **INSTANTIATE!**)
 - Constructors (**default** and **with arguments**)
 - Custom methods
- Add DbSet's in DbContext
- Add-Migration OtherEntities
 - Update-Database

Seeding Service

Checklist:

- Stop IIS
- In Data, create SeedingService
- In Startup.cs, register SeedingService for dependency injection system
- In Startup.cs, add SeedingService as parameter of Configure method. Call Seed for development profile

SellersController

Checklist:

- Create Departments and Sellers links on navbar
- Controller -> Add -> Controller -> MVC Controller - Empty -> **SellersController**
- Create folder Views/Sellers
- Views/Sellers -> Add -> View
 - View name: Index
 - Change title

SellerService and basic FindAll

Checklist:

- Create folder Services
- Create SellerService
- In Startup.cs, register SellerService to dependency injection system
- In SellerService, implement FindAll, returning List<Seller>
- In SellersController, implement Index method, which should call SellerService.FindAll
- In Views/Sellers/Index, write template code to show Sellers
- Suggestion: user classes "table-striped table-hover" for table
- Note: we're going to apply formatting in later classes

Simple Create form

Checklist:

- In Views/Sellers/Index, create link to "Create"
- In controller, implement "Create" GET action
- In Views/Sellers, create "Create" view
- In Services/SellerService create Insert method
- In controller, implement "Create" POST action

Reference:

<https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery>

Foreign key not null (referential integrity)

Checklist:

- In Seller, add DepartmentId
- Drop database
- Create new migration, update database
- Update SellerService.Insert for now: obj.Department = _context.Department.First();

SellerFormViewModel and Department select component

Checklist:

- Create DepartmentService with FindAll method
- In Startup.cs, register DepartmentService to dependency injection system
- Create SellerFormViewModel
- In controller:
 - New dependency: DepartmentService
 - Update "Create" GET action
- In Views/Sellers/Create:
 - Update model type to SellerFormViewModel
 - Update form fields
 - Add select component for DepartmentId

```
<div class="form-group">
    <label asp-for="Seller.DepartmentId" class="control-label"></label>
    <select asp-for="Seller.DepartmentId" asp-items="@((new SelectList(Model.Departments, "Id", "Name")))" class="form-control"></select>
</div>
```

- In controller, update "Create" POST action -> **NOT NECESSARY! :)**
- In SellerService.Insert, delete "First" call

Reference: <https://stackoverflow.com/questions/34624034/select-tag-helper-in-asp-net-core-mvc>

Delete seller

Checklist:

- In SellerService, create FindById and Remove operations
- In controller, create "Delete" GET action
- In View/Sellers/Index, check link to "Delete" action
- Create delete confirmation view: View/Sellers/Delete
- Test App
- In controller, create "Delete" POST action
- Test App

Seller details and eager loading

Checklist:

- <https://docs.microsoft.com/en-us/ef/core/querying/related-data>

Checklist:

- In View/Sellers/Index, check link to "Details" action
- In controller, create "Details" GET action
- Create view: View/Sellers/Details
- Include in FindAll: Include(obj => obj.Department) (namespace: Microsoft.EntityFrameworkCore)

Update seller and custom service exception

Checklist:

- Create Services/Exceptions folder
- Create NotFoundException and DbConcurrencyException
- In SellerService, create Update method
- In View/Sellers/Index, check link to "Edit" action
- In controller, create "Edit" GET action
- Create view: View/Sellers/Edit (similar do Create, plus hidden id)
- Test app
- In controller, create "Edit" POST action
- Test app

• **Notice:** ASP.NET Core selects option based on DepartmentId

Returning custom error page

Checklist:

- Update ErrorViewModel
- Update Error.cshtml
- In SellerController:
 - Create Error action with message parameter
 - Update method calls

App locale, number and date formatting

Checklist:

- In Startup.cs, define localization options
- In Seller:
 - Define custom labels [Display]
 - Define semantics for date [DataType]
 - Define display formats [DisplayFormat]

Validation

Checklist:

- In Seller, add validation annotations

```
[Required(ErrorMessage = "{0} required")]
[EmailAddress(ErrorMessage = "Enter a valid email")]
[Range(100.0, 50000.0, ErrorMessage = "{0} must be from {1} to {2}")]
```

- Update HTML for Create and Edit view

Summary:

```
<div asp-validation-summary="All" class="text-danger"></div>
```

Field:

```
<span asp-validation-for="Name" class="text-danger"></span>
```

Client-side validation:

```
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

- Update SellersController

Asynchronous operations using Tasks (async, await)

Checklist:

- Update DepartmentService
- Update SellerService
- Update SellersController

Exception handling for delete (referential integrity)

Checklist:

- Create custom exception IntegrityException
- In SellerService.RemoveAsync, catch DbUpdateException and throw IntegrityException
- In SellersController, update Delete POST action

Preparing sales search navigation views

Checklist:

- Create SalesRecordsController with Index, SimpleSearch and GroupingSearch action
- Create folder Views/SalesRecords
- Create Index view with search forms
- Create "Sales" link on main navbar
- Create SimpleSearch and GroupingSearch views

Implementing simple search

Checklist:

- Create SalesRecordService with FindByDate operation
- In Startup.cs, register SalesRecordService to dependency injection system
- In SalesRecordsController, update SimpleSearch action
- Update SimpleSearch view
- Optional: format SalesRecord date and number

Implementing grouping search

Checklist:

- In SalesRecordService create FindByDateGrouping operation
- In SalesRecordsController, update GroupingSearch action
- Update GroupingSearch view

Nivelamento: Entity Framework

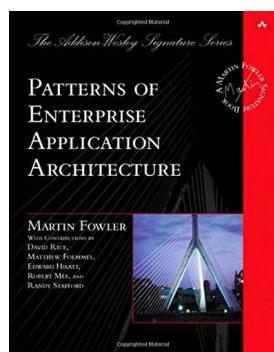
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

1

Problema

Por muitos anos, uma grande dificuldade de se criar sistemas orientados a objetos foi a comunicação com o banco de dados relacional.



Martin Fowler: ~30% do esforço de se fazer um sistema

2

Exemplo simples

```
Client client = null;
using (connection)
{
    using (var command = new SqlCommand("SELECT * FROM Clients WHERE Id = @id;", connection))
    {
        command.Parameters.Add(new SqlParameter("@id", id));
        connection.Open();
        using (var reader = command.ExecuteReader())
        {
            if (reader.Read())
            {
                client = new Client();
                client.Id = reader.GetString(0);
                client.Name = reader.GetString(1);
                client.Email = reader.GetString(2);
                client.Phone = reader.GetString(3);
            }
        }
    }
    return client;
}
```

3

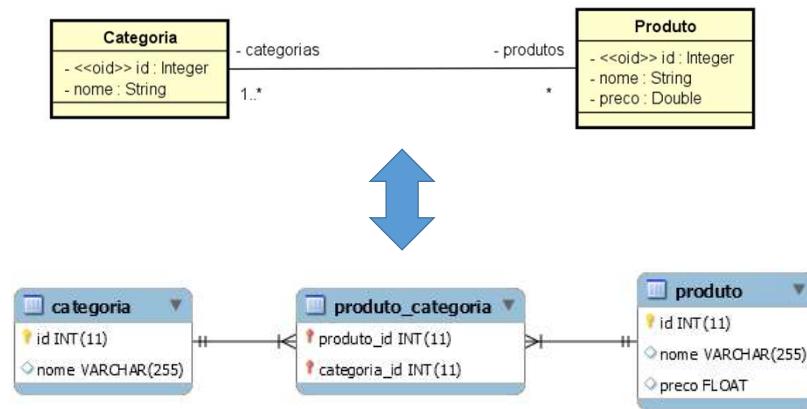
Outras questões que devem ser tratadas:

- Contexto de persistência (monitorar alterações nos objetos que estão atrelados a uma conexão em um dado momento)
 - Alterações
 - Transação
 - Concorrência
- Mapa de identidade (cache de objetos já carregados)
- Carregamento tardio (lazy loading)
- Etc.

4

Solução: Mapeamento Objeto-Relacional

ORM (Object-Relational Mapping): Permite programar em nível de objetos e comunicar de forma transparente com um banco de dados relacional



5

Entity Framework

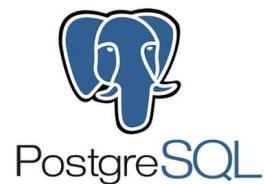
<https://docs.microsoft.com/en-us/ef/>



6

Providers

<https://docs.microsoft.com/en-us/ef/core/providers/index>



7

Principais classes

- **DbContext:** um objeto DbContext encapsula uma sessão com o banco de dados para um determinado modelo de dados (representado por DbSet's).
 - É usado para consultar e salvar entidades no banco de dados
 - Define quais entidades farão parte do modelo de dados do sistema
 - Pode definir várias configurações
 - É uma combinação dos padrões Unity of Work e Repository
 - **Unity of work:** "mantém uma lista de objetos afetados por uma transação e coordena a escrita de mudanças e trata possíveis problemas de concorrência" - Martin Fowler.
 - **Repository:** define um objeto capaz de realizar operações de acesso a dados (consultar, salvar, atualizar, deletar) para uma entidade.
- **DbSet< TEntity >:** representa a coleção de entidades de um dado tipo em um contexto. Tipicamente corresponde a uma tabela do banco de dados.

8

Processo geral para se executar operações

