

Disciplina: POO - Programação Orientada a Objetos

Métodos, Formatações

Prof^a. Dr^a. Giovana Angélica Ros Miola
giovana.miola@fatec.sp.gov.br



Convenção de nomes em C#

- **Cases:** padrão de escrita para nomes
- **Sintaxe:** conjunto de regras que deve-se obedecer, para que o programa seja executado corretamente
- **Convenção:** maneira de padronizar a codificação e auxiliar outras pessoas da equipe a entenderem a implementação
- Utilizar a convenção **PascalCase** para nomes simples ou compostos de classes e métodos, por exemplo:

Conta
ContaPoupanca

- Para nome de variáveis e argumentos/parâmetros de método utilizar o **camelCase**, por exemplo:

anoDeNascimento
valorDaCotacao

Método

- Na passagem de parâmetros, além de passar variáveis (passagem de parâmetro **POR VALOR**), podem ser passados objetos (passagem de parâmetro **POR REFERÊNCIA**), por exemplo, deseja-se transferir um valor de um correntista para outro:

```
public void TransferenciaBancaria(double valor, Conta contaDestino)
{
    if (saldo >= valor)
    {
        saldo -= valor;
        contaDestino.saldo += valor;
    }
}
```

return é executado e **sai do método**

```
public bool Sacar(double valor)
{
    if (saldo >= valor)
    {
        saldo -= valor;
        return true;
    }
    else
        return false;
}
```



```
public bool Sacar(double valor)
{
    if (saldo >= valor)
    {
        saldo -= valor;
        return true; //sair
        v = v + 1;
    }
    return false;
}
```

Formatação usando a classe String

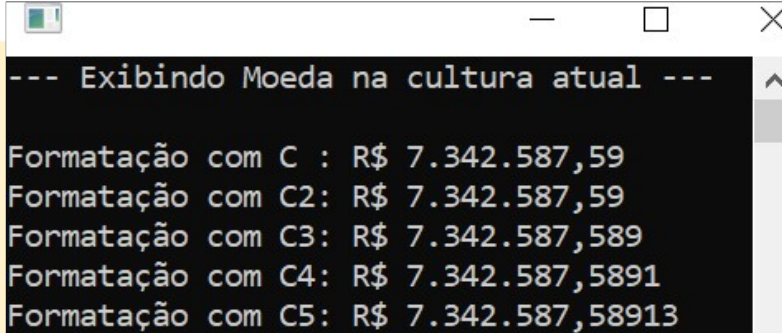
```
double valor = 1000.567, w = 7342587.58;
Console.WriteLine(String.Format("{0:N} {1:N}", valor,w));//N
numérico
// 1.000,57 ou... Token {0}      7.342.587,58 Token{1}
Console.WriteLine(String.Format("{0:0,0.00}", valor));
// 1.000,57 ou formato de MOEDA :C
Console.WriteLine("Resultado = " + String.Format("{0:C}", w));
//Resultado = R$ 7.342.587,58
```

Data e hora

```
double w = 43239.12345;
//data e hora atual
DateTime dataAtual = System.DateTime.Now;
Console.WriteLine(String.Format("Hoje dia {0}, tenho apenas {1:C} no bolso.",
dataAtual, w));
//Hoje dia 14/09/2020 11:08:06, tenho apenas R$ 43.239,12 no bolso.
//ou
string nome = "Paul"; var data = DateTime.Now;
// Formatação composta
Console.WriteLine("Olá, {0}! Hoje é {1} e são {2:HH:mm}.", nome, data.DayOfWeek, data);
//Olá, Paul! Hoje é Monday e são 11: 08.
// String interpolada
Console.WriteLine($"Olá, {nome}! Hoje é {data.DayOfWeek} e são {data:HH:mm} agora.");
//Olá, Paul! Hoje é Monday e são 11:08 agora.
```

C# - Convertendo valores para o formato monetário em diferentes culturas

```
//using é uma instrução
using System.Globalization;
static void Main(string[] args)
{
    double valor = 7342587.589134;
    //Cultura atual
    Console.WriteLine("--- Exibindo Moeda na cultura atual ---\n");
    //Por padrão, o especificador de formato "C" exibe a moeda até duas casas decimais
    Console.WriteLine("Formatação com C : " + valor.ToString("C", CultureInfo.CurrentCulture));
    // C2 exibe a moeda até dois dígitos
    Console.WriteLine("Formatação com C2: " + valor.ToString("C2", CultureInfo.CurrentCulture));
    // C3 exibe a moeda até 3 dígitos
    Console.WriteLine("Formatação com C3: " + valor.ToString("C3", CultureInfo.CurrentCulture));
    // C4 exibe a moeda até 4 dígitos
    Console.WriteLine("Formatação com C4: " + valor.ToString("C4", CultureInfo.CurrentCulture));
    // C5 exibe a moeda até 5 dígitos
    Console.WriteLine("Formatação com C5: " + valor.ToString("C5", CultureInfo.CurrentCulture));
}
```



```
--- Exibindo Moeda na cultura atual ---
Formatação com C : R$ 7.342.587,59
Formatação com C2: R$ 7.342.587,59
Formatação com C3: R$ 7.342.587,589
Formatação com C4: R$ 7.342.587,5891
Formatação com C5: R$ 7.342.587,58913
```

System.Globalizati0n - CultureInfo

- **System.Globalizati0n**: Contém classes que definem informações relacionadas à cultura, incluindo idioma, país/região, calendários em uso, padrões de formato para datas, moeda, números e ordem de classificação para cadeias de caracteres. Essas classes são úteis para escrever aplicativos globalizados (internacionalizados)
- **CultureInfo**: Fornece informações sobre uma cultura específica (chamada de *localidade* para o desenvolvimento de código). As informações incluem os nomes da cultura, o sistema de escrita, o calendário usado, a ordem de classificação das cadeias de caracteres e a formatação de datas e números.

Namespaces organizam tipos

- Os **namespaces** são usados intensamente em programações de C# de duas maneiras.
- Em **primeiro lugar**, o .NET usa namespaces para organizar suas muitas classes, da seguinte forma:

```
System.Console.WriteLine("Hello World!");
```

System é um **namespace** e **Console** é uma classe nesse namespace. A palavra-chave pode ser usada para que o nome completo **using** seja necessário, como no exemplo a seguir:

```
using System;  
Console.WriteLine("Hello World!");
```

Namespaces organizam tipos

- Em **segundo lugar**, declarar seus próprios namespaces pode ajudar a controlar o escopo dos nomes de classe e de método em projetos de programação maiores. Use a palavra-chave [namespace](#) para declarar um namespace, como no exemplo a seguir:

```
namespace NamespaceExemplo
{
    class ClasseExemplo
    {
        public void MetodoExemplo()
        {
            System.Console.WriteLine( "Hello World");
        }
    }
}
```

Tipo implícito var

- **var** instrui o compilador para **deduzir** o tipo da variável da expressão
- O tipo inferido pode ser um tipo interno, um tipo anônimo, um tipo definido pelo usuário ou um tipo definido na biblioteca de classes do .NET
- Restrições que se aplicam às declarações de variável de **tipo implícito**:
 - var pode ser usado apenas quando uma variável local é declarada e **inicializada** na mesma instrução,
 - a variável **não pode** ser inicializada **como nula**.
 - var não pode ser usado em atributos no escopo da classe.

- Exemplos:

```
int a= 10; //explícito
//b é compilado como int
var b= 10; //implícito
//c é compilado como string
var c= "Saulo";
//v é compilado como um int[]
var v= new[] {1 2 3 4 5};
```