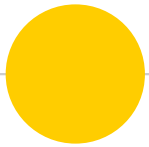
Introdução à Plataforma .NET

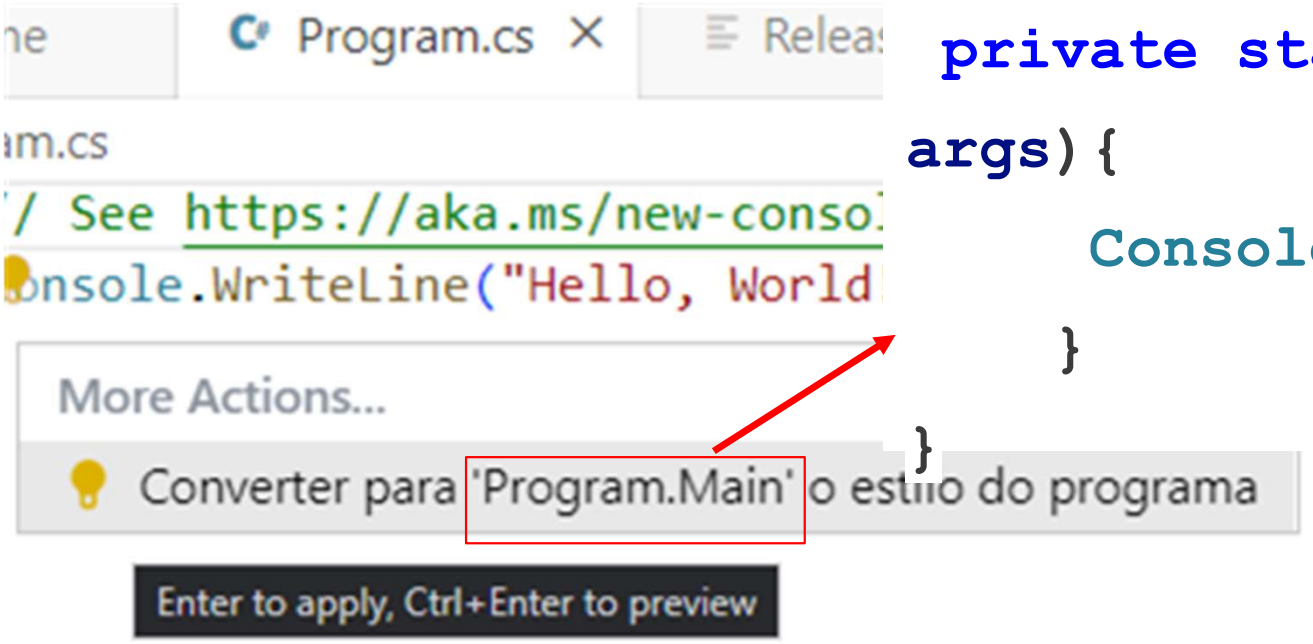
Profa. Luciana Balieiro Cosme
Ciência da Computação/IFNMG Montes Claros

Template SlidesCarnival



Classes e objetos

Projeto com classes e objetos




```
internal class Program{
 private static void Main(string[]
args){
        Console.WriteLine("Hello!");
    }
}
```

Namespace

- Jeito lógico de agrupar um código
- Pode ter hierarquia
- Adicione a sua classe Programa um namespace

```
namespace meuapp;
```

- Diretiva using

```
using meuapp;
```

Classe

```
namespace meuapp;
class MinhaClasse{
  public void Print() {
   System.Console.WriteLine
("Hello World");
 }
}
```

minhaclasse.cs

```
namespace meuapp;
class Program{
  private static void
Main(){
    MinhaClasse m = new();
    m.Print();
  }
}
```

program.cs

Classe e objetos

- Por convenção, letra maiúscula para cada palavra (classe e método)
- Membros de instâncias: necessário criar um objeto (new)
- Pode ter construtor (explícito) e destrutor (~)

Qual é o problema do seguinte código?

```
class Program{
      int a=10;
      static void Main() {
            int x = 10;
         Console.Write("Soma = "+ a+x);
      }
}
```

Tipos anônimos

```
var v = new { first = 1, second =
true };
System.Console.WriteLine(v.second);
```

Métodos

- Sobrecarga e parâmetros

```
//classe MinhaClasse
public void Print(string s1,string s2){
    System.Console.WriteLine(s1+s2);
}
```

Métodos

- Params

```
m.Print("Olá ", "Fulano",
" e ", "Ciclano" );
```

```
//classe MinhaClasse
public void Print(params string[] s) {
            foreach(string x in s)
           Console.WriteLine(x);
      }
```

Métodos

- parâmetros opcionais

```
 public void Print(string s1, string
s2="Olá, ") {
      System.Console.WriteLine(s2+s1);
 }
```

Métodos

- Parâmetros nomeados
- return

```
System.Console.WriteLine
( m.Sum(i:1,2) );
```

```
public int Sum(int i, int j = 0,
int k = 0) {
   return 1*i + 2*j + 3*k;
}
```

Passagem por valor e por referência

```
public void Set(int i) {
i = 10; } //valor

public void Set(int[] i) {
        i[0]=2; };
} //ref
```

Palavra chave ref

- passagem por referência

```
public void Set(ref int i) {
i = 10; }
```

- chamada do método: `m.Set(ref x);`

Palavra chave out

- passagem de variável não assinalada

```
public void SetOut(out int i) {
i = 10; }
```

- chamada do método:

```
m.SetOut(out int x);
System.Console.Write(x);
```

Propriedades: getters e setters

```
class Time{
    private int secs;
    public int Secs{
      get { return secs; }
        set{ //sec=value;
            if (value > 0) secs = value;
             else secs = 0;
            }
      }}
```

```
class Time{
    public int Secs
    { get; set; }
}
Time t = new Time();
t.Secs = 5;
int s = t.Secs;
```

Para praticar

```
namespace test_oo;
public class Example{
    public Example(){}
    public Example(int n, double d){
        N=n; D=d;}
    public int N{get;set;}
    public double D{get;set;}
    public override string ToString(){
        return "Exemplo: n ="+ N+", d = "+ D;
}}
```

Para praticar

```
static void Main(){
   Example e=new(1,2.5);
   Console.WriteLine(e);
   Console.WriteLine(e.N);
…
   Example e2 = new();
   e=e2;
   e2.N = 4;
   e2.D = 11.5;

Console.WriteLine(e);
}
```

Exercício 4

1. Criar uma classe para calcular a **média e o desvio** padrão de um conjunto de dados de **entrada (int)**, que deve ser fornecido pelo usuário sequencialmente, sem tamanho pré-definido. Deve-se ter pelo menos um **construtor**, as **propriedades**, um **método para receber** um número da sequência, um para calcular a média e o desvio, e uma sobrecarga para **ToString**(). O usuário deve escolher se quer adicionar mais um número, ou verificar a médio e o desvio ou finalizar.

Referências

- Parâmetros nomeados e opcionais => https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/named-and-optional-arguments
- Expressões lambda => https://learn.microsoft.com/pt-pt/dotnet/csharp/language-reference/operators/lambda-expressions
- Formas de arredondamento https://learn.microsoft.com/en-us/dotnet/api/system.midpointrounding?view=net-7.0

Referências

- Métodos de extensão => https://learn.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/how-to-implement-and-call-a-custom-extension-method
- Classes
  https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/types/classes
- Herança
  https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/inheritance