Programação Visual

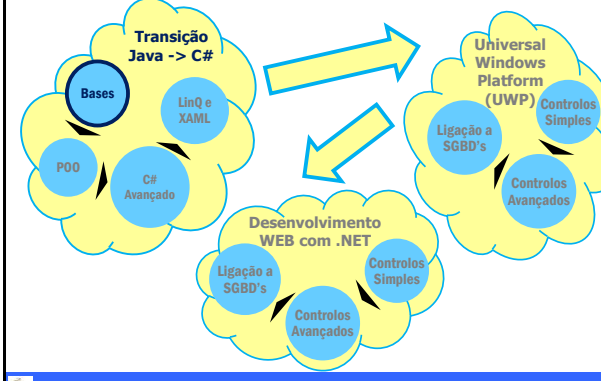
C# - Transição Java -> C#

1

PROGRAMA

1. TRANSIÇÃO P/ C#

1.2 POO



2

C# - Elementos básicos

- Instruções de seleção
  - (?: , switch/case, if/else)
- Instruções de repetição
  - (foreach, for, do/while, while)
- Tabelas (arrays ou vetores)
  - Tinham em java (unidimensionais[ ],
  - NÃO TINHAM EM JAVA multidimensionais[,,],
  - Tinham em java de tabelas [ ][ ]
- Tipo Enumerativo
  - (enum)
- Classes
  - (public class nome)

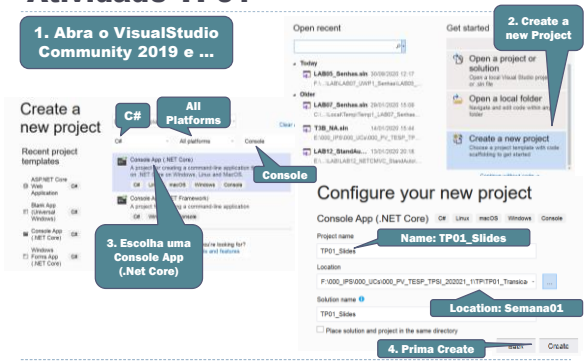
3

Atividade TP01

- **No pasta onde guarda os materiais do curso:**
  1. **Crie uma pasta ProgVis2021**
  2. **Dentro da pasta ProgVis2021 crie um outra pasta Semana01**
  3. **Abra o browser e vá ao moodle**
  4. **Descarregue o .zip S012, guarde-o na pasta Semana01 e descompacte-o.**

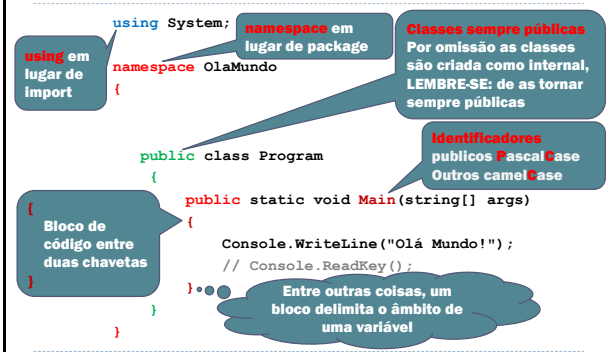
4

Atividade TP01



5

C# - Estrutura dos programas



6

C# - Tipos de dados

- **Um tipo de dados define:**
  - **Um conjunto de valores e um conjunto de operações com esses valores**
- **Tipos de dados por valor e por refereência**
  - **Tipo por valor:**
    - **Na variável é escrito um valor**
  - **Tipo por referencia:**
    - **Na variável é escrita uma referencia (O endereço de memória onde está o objeto)**
- **Em c# os tipos simples (incorporados ou built-in) :**
  - **São nomes alternativos (alias) para as classes existentes.**
  - **Os tipos simples são tipos por valor.**
- **Identificadores:**
  - **São os nomes das entidades de um programa**
  - **Em c# os identificadores públicos são PascalCase, os restantes são camelCase**

7

C# - Tipos de dados incorporados (built-in)

Por valor	.NET Class	Type
byte	Byte	Unsigned integer
sbyte	SByte	Signed integer
int	Int32	Signed integer
uint	UInt32	Unsigned integer
short	Int16	Signed integer
ushort	UInt16	Unsigned integer
long	Int64	Signed integer
ulong	UInt64	Unsigned integer
float	Single	Single-precision floating point type
double	Double	Double-precision floating point type
char	Char	A single Unicode character
bool	Boolean	Logical Boolean type
decimal	Decimal	Precise fractional or int can represent decimals w/ 29 significant digits
object	Object	Base type of all other types
string	String	A sequence of characters
dynamic	Object	Variáveis e referencias não são verificadas em compile time

8

C# - Operadores

Operator category	Operators
Arithmetic	+ - * / %
Logical (Boolean and bitwise)	& | ^ ! ~ && || true false
String concatenation	+
Increment, decrement	++ --
Shift	<< >>
Relational	== != < > <= >=
Assignment	= += -= *= /= %= &= |= ^= <<= >>= ??
Member access	.
Indexing	[ ]
Cast	( )
Conditional	? :
Delegate concatenation and removal	+ -
Object creation	new
Type information	as is sizeof typeof
Overflow exception control	checked unchecked
Indirection and Address	* -> [ ] &

9

C# - Estruturas de Selecção - ?:

```
condition ? consequent : alternative
```

Leia como

is this condition true ? yes : no

Quando Usar?
Sempre que o resultado da avaliação seja atribuível a uma variável e facilite a leitura

```
Random rand = new Random();
int x = (rand.NextDouble() > 0.5) ? 1 : 0;
```


10

C# - Estruturas de Selecção

```
switch (str){
    case "janeiro":
    case "março":
    case "maio":
    case "julho":
    case "agosto":
    case "outubro":
    case "dezembro": Console.WriteLine( "Tem 31 dias");
       break;
    case "abril":
    case "junho":
    case "setembro":
    case "novembro": Console.WriteLine( "Tem {0} dias", 30);
       break;
    default: Console.WriteLine( "Mês desconhecido");
       break;
    case "fevereiro": Console.WriteLine("Tem 28 ou 29 dias");
       break;
}
```

Quando Usar?
Sempre que possamos ...
Infelizmente só podemos quando str for uma String ou um enumerável.

se não usarem break a seguir a instrução dá erro sintático

11

C# - Estruturas de Selecção

```
if ( (str == "janeiro") ||
          (str == "março") ||
          (str == "maio") ||
          (str == "julho") ||
          (str == "agosto") ||
          (str == "outubro") ||
          (str == "dezembro") ||
              Console.WriteLine("Tem 31 dias");
else if ((str == "abril") ||
          (str == "junho") ||
          (str == "setembro") ||
          (str == "novembro") ||
              Console.WriteLine ("Tem {0} dias", 30);
else if (str == "fevereiro")
              Console.WriteLine("Tem {0} ou {1} dias", 28, 29);
else
   Console.WriteLine("Mês desconhecido");
```

Quando Usar?
Quando str for boolean ou não pudermos usar o switch

12

C# - Estruturas de Seleção

- Quando usar qual delas?
- Operador condicional ?
  - Sempre que o resultado da avaliação seja atribuível a uma variável e facilite a leitura.
- Switch
  - Quando str for uma String ou um enumerável.
- If/else
  - Quando não pudermos usar os dois anteriores

13

C# - Estruturas de Repetição

- **foreach (ler "para cada")**

Quando Usar?
Sempre que possamos!
Infelizmente só podemos usar para iterar toda uma coleção

```
string str = "abcdefgh";
foreach (char s in str)
   Console.Write(" {0}-", s);
```

- **for**

Quando Usar?
Sempre que possamos!
Infelizmente só podemos usar quando sabemos quantas vezes (ou o intervalo de valores) queremos repetir as instruções

```
for (int i = 0; i < 10; i++)
   Console.Write(" {0}-", i);
```


14

C# - Estruturas de repetição

- **do-while**

Quando Usar?
Quando o bloco deve ser executado pelo menos uma vez

```
int i=0;
do {
  Console.Write(" {0}-", i);
  i++;
} while (i < 20);
```

- **while**

Quando Usar?
Quando o bloco pode nunca ser executado

```
while (i < 30){
  Console.Write(" {0}-", i);
  i++;
}
```


15

C# - Tabelas / Arrays

- //Uni-dimensionais

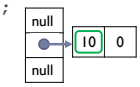
```
double[ ] values = new double [3];
values[1] = 2.0;
Console.WriteLine( values [2]);
```

- //Multi-dimensionais

```
int[,] vals1 = new int [3, 2];
vals1[2, 1] = 5;
Console.WriteLine(vals1 [2, 1]);
```

- //Tabelas de tabelas

```
byte[ ][ ] vals2 = new byte[3][ ];
vals2[1] = new byte[2];
vals2[1][0] = 10;
Console.WriteLine(vals2 [1][0]);
```



16

C# – Tipos Enumerados

- **Oferecem uma forma eficiente para definir um conjunto de constantes inteiras que podem ser atribuídas a variáveis.**

```
public enum Unidade
{
   EST_SETUBAL = 210, EST_BARREIRO = 262, IPE = 269
}
```

enum – palavra reservada para definir um tipo enumerativo (ou enumerado)

```
/////// Algures no main ///////
Unidade unidade = Unidade.EST_BARREIRO;
Console.WriteLine(Unidade.EST_SETUBAL);//-> EST_SETUBAL
Console.WriteLine((int)Unidade.EST_SETUBAL);//-> 210
Console.WriteLine(unidade); // -> EST_BARREIRO
```

Declaração de uma variável "unidade" do tipo Unidade e atribuição do valor EST_BARREIRO

17

C# - Tipos enumerados

- **Exemplos:**

Tipo subjacente – o tipo de dados usado para enumerar as constantes

- // por omissão: tipo subjacente é int começa em 0

```
enum Season0 { Spring, Summer, Fall, Winter };
```

- // tipo subjacente é int começa em 3

```
enum Season1 { Spring = 3, Summer, Fall, Winter };
```

- // tipo subjacente é byte começa em 0

```
enum Season2 : byte { Spring, summer, Fall, Winter };
```

- // tipo subjacente é long começa em 3
- // Fall = Autumn = 5

```
enum Season3 : long { Spring = 3, Summer, Fall, Autumn = Fall, Winter};
```


18

C# - Estruturas – não abordamos

```
struct tempo{
  public int horas, minutos, segundos;
  public tempo(int horas){
       this.horas = horas;
       minutos = 0;
       segundos = 0;
  }
}

tempo t = new tempo(2);
Console.WriteLine("Valor -> {0}", t.horas);
```


19

C# - Operadores e Instruções

Using

A instrução **using** permite usar uma memória local para uma determinada variável que é válida dentro do bloco do *using*. A classe dessa variável deverá implementar a interface **IDisposable** onde está declarado o método **Dispose()**. Este método é **sempre** chamado (implicitamente) no fim do bloco

```
string manyLines=@"This is line one
This is line two
Here is line three
The penultimate line is line four
This is the final, fifth line.";

using (var reader = new StringReader(manyLines))
{
   string? item;
   do {
      item = reader.ReadLine();
      Console.WriteLine(item);
   } while(item != null);
}
```


20

C# - Operadores e Instruções

Using

A partir do C# 8.0 podemos usar uma sintaxe alternativa que não exige parentesis

```
string manyLines=@"This is line one
This is line two
Here is line three
The penultimate line is line four
This is the final, fifth line.";

using (var reader = new StringReader(manyLines))
string? item;
do {
    item = reader.ReadLine();
    Console.WriteLine(item);
} while(item != null);
```

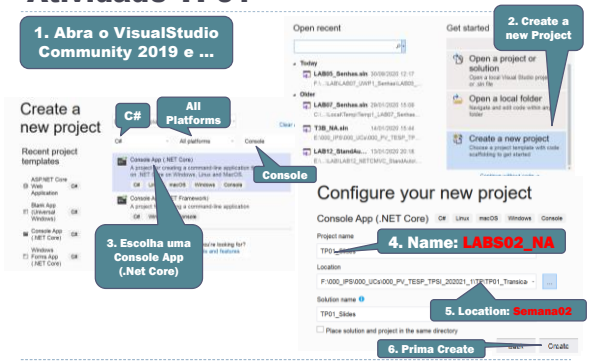

21

C# - Classes

- **Campos** – são os atributos do Java
  - Podem ser inicializados na declaração
  - **BPP**: são sempre privados e (por enquanto) usam os get e set para aceder
- **Métodos**
  - Código definido dentro da classe
  - Podem ter várias formas com o mesmo identificador
  - **BPP**: só são públicos se tiverem de ser acedidos no exterior da classe
- **Métodos Construtores**
  - O construtor por omissão inicializa todos os 'campos'
  - Podem existir outros construtores, podendo ser então necessário definir o construtor sem argumentos
  - **BPP**: só se escreve o código do construtor com mais parâmetros, os restante reutilizam esse construtor.

22

Atividade TP01



23