

### Notas:

- Introdução à definição de classes e criação de objetos (instanciação de classes).
- Todos os identificadores de tipos de dados, variáveis e métodos **terminam em \_NA**, por exemplo: Livro\_NA, titulo\_NA, livro\_NA, onde **\_NA** são as iniciais do seu primeiro **N**ome e último **A**pellido.

### Crie um projeto T02\_NomeApellido

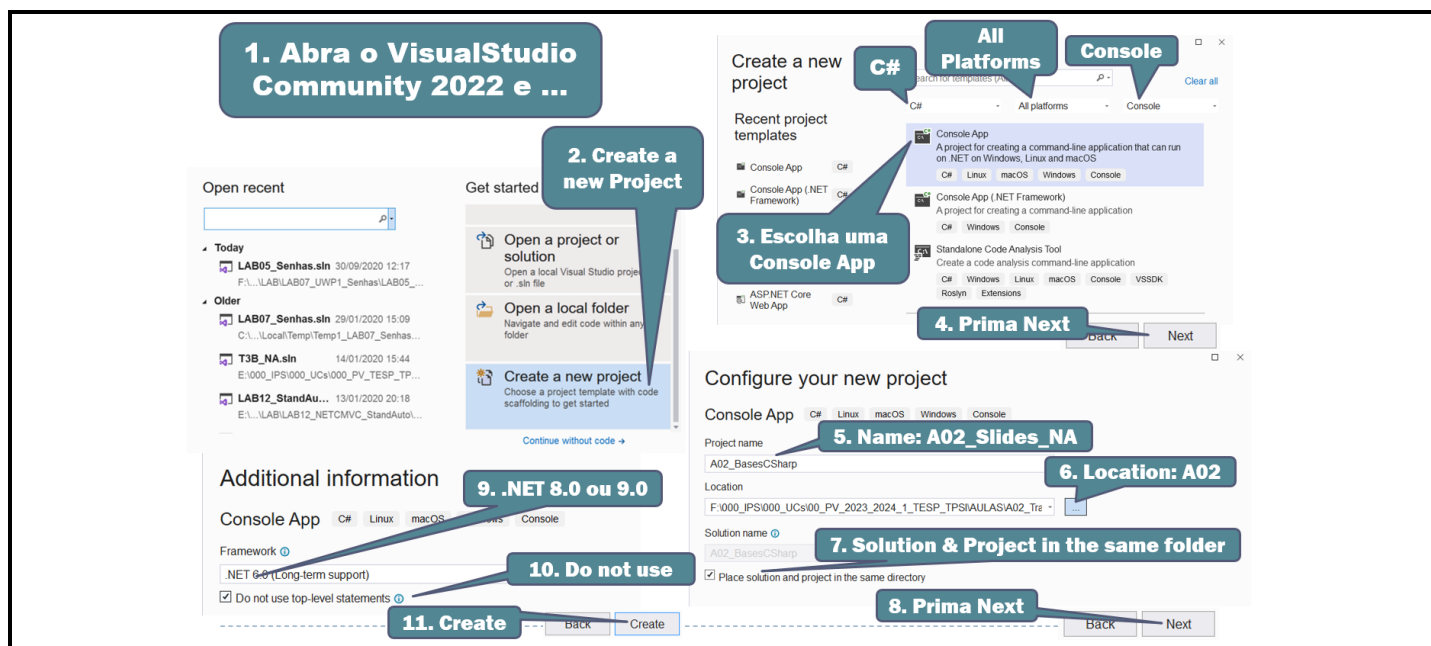


Figura 0.1: Passos para criar um projeto no Visual Study Community 2022.

## 1 Tipos enumerados (enums ou enumerativos)

### 1.1 Crie e use vários tipos enumerados:

1. Com o cursor do rato sobre o nome do projeto no Solution Explorer, prima o botão direito do rato e escolha: Add -> Class. -> Name: Season0.cs (para que a classe se chame Season0) tal como na Figura 1.1A.

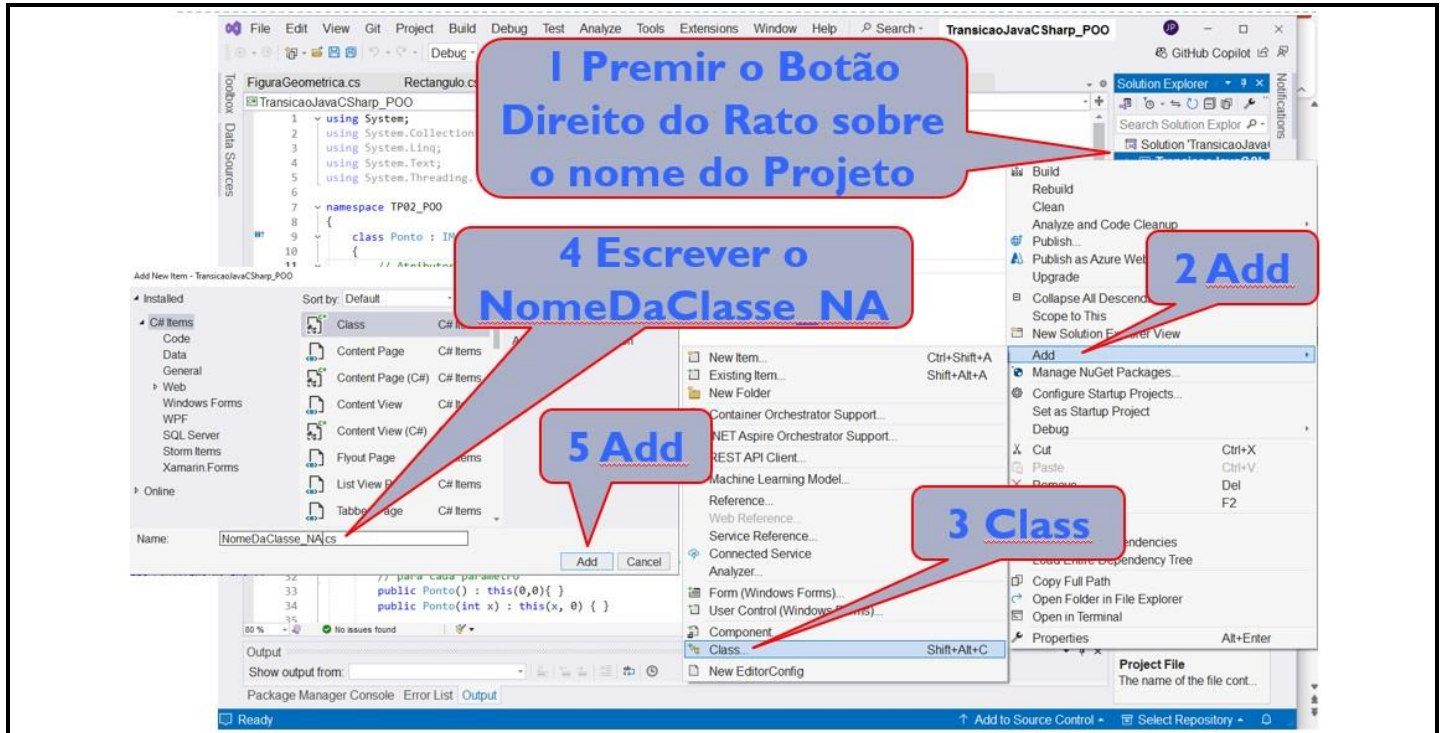
**NOTE QUE:** em c# tal como em Java, cada classe ou tipo enumerado deve estar no seu ficheiro que deve ter como nome o nome do tipo de dados (classe ou enumerado).

2. A partir do Solution Explorer, abra o ficheiro Season0.cs se ainda não estiver aberto e adicione-lhe o código a negro na figura 1.1B.

**NOTE QUE:** o nome do namespace do ficheiro Season01 é igual ao namespace do ficheiro com a classe Program que contém o método Main.

3. Procedendo como em 1 e 2, crie um enumerado Season1 cujo tipo subjacente seja inteiro e comece em 3 usando o código da Figura 1C.
4. Procedendo como em 1 e 2, crie um enumerado Season2 cujo tipo subjacente seja byte e comece em 0 usando o código da Figura 1D.
5. Por analogia e procedendo como em 1 e 2 crie um enumerado Season3 cujo tipo subjacente seja long e comece em 5.
6. No método Main digite o código da Figura 1E, execute e deverá obter uma saída semelhante à da Figura R1.1.

## Tutorial 2 – Transição java -> C# (Classes e Objetos)



**Figura 1.1A: Passos para criar uma classe no Visual Study Community 2022.**

```

espace LABS02_NA
{
    // por omissão: tipo subjacente é int começa em 0
    enum Season0 {
        Spring, Summer, Fall, Winter
    };
}
    
```

**Figura 1.1B: código para o ponto 2**

```

namespace LABS02_NA
{
    // por omissão: tipo subjacente é int começa em 3
    enum Season1 {
        Spring= 3, Summer, Fall, Winter
    };
}
    
```

**Figura 1.1C: código para o ponto 3**

```

namespace LABS02_NA
{
    // por omissão: tipo subjacente é byte e começa em 0
    enum Season2 : byte {
        Spring, Summer, Fall, Autumn = Fall, Winter
    };
}
    
```

**Figura 1C: código para o ponto 4**

```
Season0 estação0 = Season0.Fall;
Console.WriteLine(estação0);      // escreve 'Fall'
Console.WriteLine((int)estação0); // escreve '2'

Season1 estação1 = Season1.Fall;
Console.WriteLine(estação1);      // escreve 'Fall'
Console.WriteLine((int)estação1); // escreve '5'

Season2 estação2 = Season2.Autumn;
Console.WriteLine(estação2);      // escreve 'Autumn'
Console.WriteLine((int)estação2); // escreve '2'

Season3 estação3 = Season3.Fall;
Console.WriteLine(estação3);      // escreve 'Fall'
Console.WriteLine((int)estação3); // escreve '5'

Console.WriteLine("\nprima return para continuar");
Console.ReadKey();
```

Figura 1D: código para o ponto 6

## 2 Classes

Uma classe é um Tipo de Dados que abstrai todos os objetos (instâncias/ocorrências) desse tipo.

### 2.1 Crie a classe Livro\_NA e declare os seus campos

- Com o cursor do rato sobre o nome do projeto no Solution Explorer, prima o botão direito do rato e escolha: Add -> Class. -> Name: Livro\_NA.cs (para que a classe se chame Livro\_NA). Ou veja os passos na Figura 1.1A.
- Logo abaixo do cabeçalho da classe Livro\_NA declare os campos de instância (atributos de instância do Java) escrevendo as **linhas 13 a 17 da Figura 2.1**.

**NOTE QUE:** em C#, por omissão, os membros de instância (campos e métodos de instância) de uma classe são privados pelo que não podemos aceder-lhes (ler/alterar) fora da classe.

### 2.2 Praticar a definição de classes e a declaração de atributos

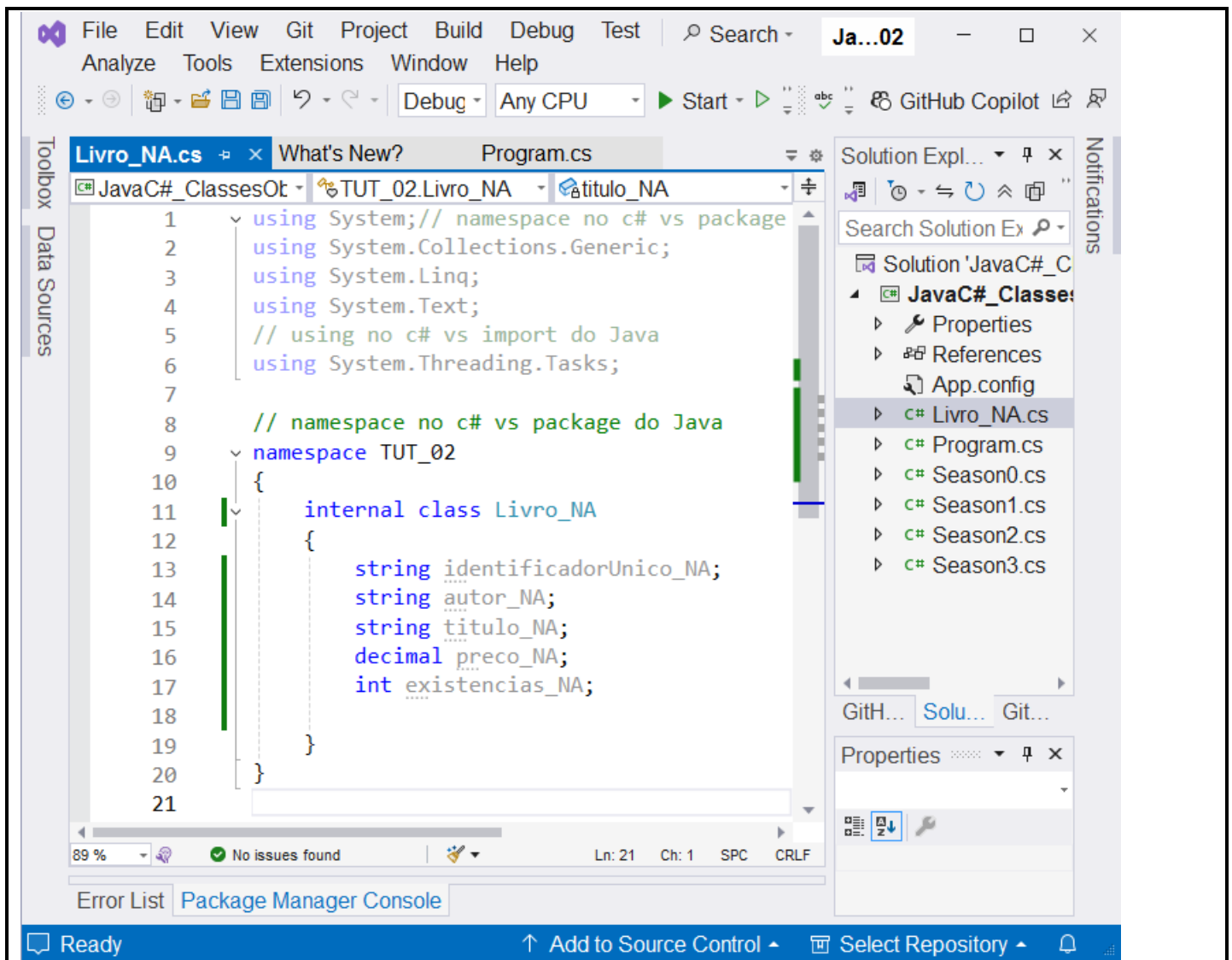
- Repita os passos 7 e 8 para criar, por analogia, a classe Cd\_NA com campos de instância (atributos de instância) identificadorUnico\_NA (String), titulo\_NA (String), interprete (String), duracao\_NA (int), preco\_NA (decimal), existencias\_NA (int).
- Repita os passos 7 e 8 para criar, por analogia, a classe Dvd\_NA com campos de instância identificadorUnico\_NA (String), titulo\_NA (String), realizador (String), duracao\_NA (int), preco\_NA (double), existencias\_NA (int).

## 3 Criar Objetos

Um objeto (instância da classe) representa uma entidade do mundo real ou do domínio do problema

### 3.1 Criar um objeto da classe Livro\_NA

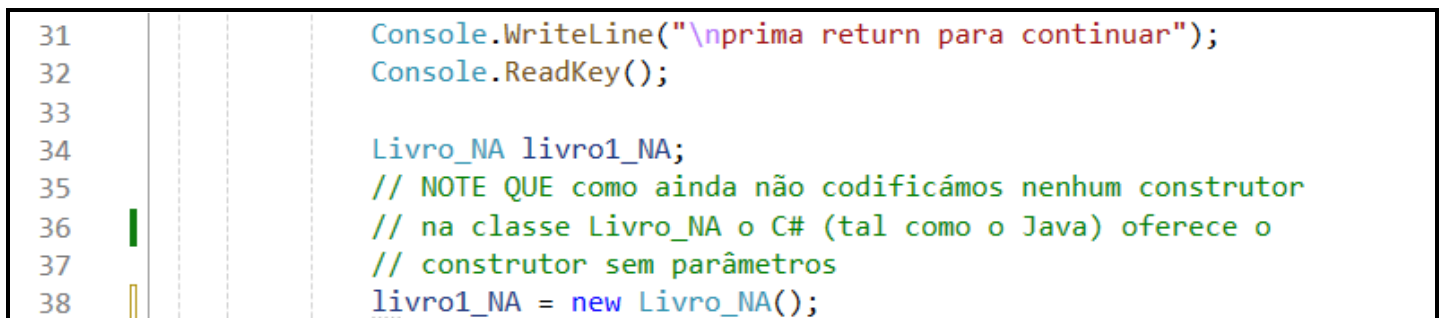
- No método main escreva o código (e os comentários) das **linhas 33 a 38 da Figura 3.1**.



**Figura 2.1: Definição da classe Livro\_NA**

### 3.2 Praticar a criação de objetos

12. Por analogia com o código das linhas 34 e 38 da Figura 3.1, crie mais dois livros denominados livro2 e livro3. Experimente declarar e inicializar as variáveis na mesma linha.
13. Por analogia com o código das linhas 34 e 38 da Figura 3.1, crie um objeto do tipo Cd\_NA denominado cd1\_NA e outro do tipo Dvd\_NA denominado dvd1\_NA.



**Figura 3.1: Criação de objetos e escrita/leitura de campos public**

### 4 Métodos Construtores

Os métodos construtores de uma classe servem para criar objetos dessa classe.

Observando as Boas Práticas de Programação (BPP), por convenção, na UC de POO todos os atributos são inicializados nos métodos construtores.

#### 4.1 Codificação de métodos construtores

Vamos usar um *snippet* <sup>1</sup> do VS, para gerar o construtor sem parâmetros e em seguida codificá-lo para inicializar todos os campos, para tal:

- Vá para a janela que tem a classe Livro\_NA e, logo abaixo dos atributos que definiu no ponto 9 (Figura 2.1, linha 18), insira o cursor, **escreva `ctor` e prima a tecla `tab`**. Deve obter um código semelhante ao da Figura 4.1. (caso tal não aconteça, escreva o código da Figura 4.2 (mais tarde terá oportunidade de treinar o uso deste e de outros snippets).

```

19      public Livro_NA()
20      {
21      }
22  
```

Figura 4.1: Construtor sem parâmetros gerado pelo snippet `ctor` na classe Livro\_NA

- Codifique e formate a lista de parâmetros do construtor gerado por forma a mostrar os parâmetros tal como nas linhas 19 a 30 da Figura 4.2.
- Prima o botão para guardar o ficheiro.  
**NOTE QUE** surgiram 3 erros sintáticos nas linhas 38 a 40 do método Main (Figura 4.3). Tal deve-se ao facto de termos codificado um construtor e, por isso, o C# deixar de nos oferecer o construtor sem parâmetros que usámos para criar os objetos livro1, 2 e 3. O erro deixará de existir assim que criarmos o nosso próprio construtor sem parâmetros. Para tal:
- Na janela da classe Livro\_NA, insira o cursor na linha abaixo da chave que fecha o construtor com parâmetros e **escreva o código das linhas 32 a 38 da figura 4.4** para codificar um construtor que não aceita argumentos e, por isso, deve invocar o construtor que codificou no ponto 16, usando a sintaxe : `this ( )` para lhe passar argumentos por omissão.

```

19      public Livro_NA(string identificadorUnico,
20                      string autor,
21                      string titulo,
22                      decimal preco,
23                      int existencias)
24      {
25          identificadorUnico_NA = identificadorUnico;
26          autor_NA = autor;
27          titulo_NA = titulo;
28          preco_NA = preco;
29          existencias_NA = existencias;
30      }

```

Figura 4.2: Construtor com parâmetros da classe Livro\_NA

<sup>1</sup> <https://learn.microsoft.com/en-us/visualstudio/ide/visual-csharp-code-snippets?view=vs-2022>. Snippets, traduzido à letra significa trecho (pequeno pedaço de texto de um contexto maior).

## Tutorial 2 – Transição java -> C# (Classes e Objetos)

```

38     livro1_NA = new Livro_NA();
39     Livro_NA livro2_NA = new Livro_NA();
40     Livro_NA livro3_NA = new Livro_NA();
41
42     Cd_NA cd1_NA = new Cd_NA();
43     Dvd_NA dvd1_NA = new Dvd_NA();
    
```

Figura 4.3: Erros sintáticos no método Main provocados pelo desaparecimento do construtor sem parâmetros quando codificámos um construtor com parâmetros.

```

32     public Livro_NA() : this("IU_000000000",
33                               "Sem Autor Definido",
34                               "Sem Título Atribuido",
35                               0.0M,
36                               0)
37     {
38     }
39
    
```

Figura 4.4: Construtor sem parâmetros que invoca o construtor codificado usando a sintaxe : this( ).

### 4.2 Praticar a codificação de métodos construtores

18. Repita os passos 14 a 17 para criar, por analogia, construtores com e sem parâmetros para a classe **Cd\_NA**.
19. Repita os passos 14 a 17 para criar, por analogia, construtores com e sem parâmetros para a classe **Dvd\_NA**.

### 4.3 Criação de objetos com diferentes construtores<sup>2</sup>

20. No método main **escreva o código das linhas 42 a 55 da Figura 4.5** para atribuir às variáveis livro2\_NA e Livro3\_NA dois objetos criados com o construtor com parâmetros.

```

42     Cd_NA cd1_NA = new Cd_NA();
43     Dvd_NA dvd1_NA = new Dvd_NA();
44
45     livro2_NA = new Livro_NA("IU_000000001",
46                               "Memorial do Convento",
47                               "José Saramago",
48                               25.5M,
49                               3);
50
51     livro3_NA = new Livro_NA("IU_000000002",
52                               "O Conde de Abranhos",
53                               "Eça de Queirós",
54                               15.3M,
55                               2);
    
```

Figura 4.5: Inicialização de variáveis com objetos criados com diferentes construtores

<sup>2</sup> Como terá notado na secção 4.1, em C# (tal como em Java) pode ter diferentes métodos construtores com o mesmo nome desde que tenham diferentes listas de parâmetros



## Tutorial 2 – Transição java -> C# (Classes e Objetos)

### 4.4 Praticar a utilização de métodos construtores

21. Por analogia com o que fez no passo 20 atribua às variáveis `cd1_NA` e `dvd1_NA`, que criou na secção 4.2, dois objetos criados com os construtores com parâmetros das respetivas classes (use valores de uma música e de um filme que conheça).

## 5 Métodos Acessores<sup>3</sup>

Sendo os campos (atributos) da classe sempre privados, a forma de obter os seus valores passa por codificar métodos acessores (Getters) que retornem o valor desse campo.

### 5.1 Codificação de métodos acessores (Getters)

22. Abra a janela com a classe `Livro_NA` e escreva o código das linhas 40 a 44 da Figura 5.1.

**NOTE QUE:** os acessores são normalmente públicos, retornam um tipo de dados do mesmo tipo do campo a que acedem e não aceitam argumentos. Sendo públicos, para respeitar a notação do C# escrevem-se em PascalCase.

```

40 public string GetIdentificadorUnico_NA() { return identificadorUnico_NA;}
41 public string GetAutor_NA() { return autor_NA; }
42 public string GetTitulo_NA() {return titulo_NA;}
43 public decimal GetPreco_NA() {return preco_NA;}
44 public int GetExistencias_NA() { return existencias_NA; }
```

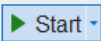
Figura 5.1: Métodos acessores (Getters) para os campos (atributos) da classe `Livro_NA`

### 5.2 Praticar a codificação de métodos acessores (Getters)

23. Por analogia com o que fez no passo 22 crie métodos assessores (Getters) para todos os campos das classes `Cd_NA` e `Dvd_NA`.

### 5.3 Utilização (invocação) de métodos acessores (Getters)

Tratando-se de métodos instância a invocação de métodos assessores exige a prévia criação de um objeto da classe e em seguida a invocação do método para esse objeto através do **operador ponto**.

24. Para imprimir na consola os valores dos campos dos objetos `livro1_NA` e `livro2_NA` **escreva o código das linhas 57 a 67 da Figura 5.2.**
25. Para compilar e executar o programa prima o botão  da barra de ferramentas e verifique se obtém um resultado semelhante ao da Figura R5.1, caso contrário corrija o código que escreveu e volte a correr o programa.

### 5.4 Praticar a utilização (invocação) de métodos assessores

26. Por analogia com o código que escreveu no passo 24 imprima na consola os valores dos campos dos objetos `cd1_NA` e `dvd1_NA` que criou no ponto 23.

<sup>3</sup>**NOTE QUE:** o C# oferece um mecanismo mais apropriado e transparente para lidar com o encapsulamento, ou mais propriamente com o encobrimento, de dados (campos/atributos) baseado em **Propriedades** públicas. Em breve passaremos a usá-lo, e deixaremos de lado os getters e setters que aqui são abordados apenas no contexto da transição da linguagem de programação Java para C#.

## Tutorial 2 – Transição java -> C# (Classes e Objetos)

```

57 Console.WriteLine("*****\nLIVROS EM STOCK:");
58 Console.WriteLine("\n*** LIVRO: " + livro1_NA.GetIdentificadorUnico_NA()
59     + "\n* " + livro1_NA.GetTitulo_NA()
60     + "\n* " + livro1_NA.GetAutor_NA()
61     + "\n* " + livro1_NA.GetPreco_NA()
62     + "\n* " + livro1_NA.GetExistencias_NA());
63 Console.WriteLine("\n*** LIVRO: " + livro2_NA.GetIdentificadorUnico_NA()
64     + "\n* " + livro2_NA.GetTitulo_NA()
65     + "\n* " + livro2_NA.GetAutor_NA()
66     + "\n* " + livro2_NA.GetPreco_NA()
67     + "\n* " + livro2_NA.GetExistencias_NA());
68 Console.WriteLine("\n*** LIVRO: " + livro3_NA.GetIdentificadorUnico_NA()
69     + "\n* " + livro3_NA.GetTitulo_NA()
70     + "\n* " + livro3_NA.GetAutor_NA()
71     + "\n* " + livro3_NA.GetPreco_NA()
72     + "\n* " + livro3_NA.GetExistencias_NA());
73
74 Console.WriteLine("\nprima return para continuar");
75 Console.ReadKey();
    
```

Figura 5.2: Saída para a consola dos valores dos campos de dois objetos.

## 6 Métodos Modificadores

Sendo os campos (atributos) da classe sempre privados, a forma de modificar os seus valores passa por codificar métodos modificadores (Seters) que recebem um novo valor para o campo e lho atribuem.

### 6.1 Codificação de métodos modificadores (seters)

27. Para codificar métodos modificadores (geters) da classe Livro\_NA, na janela da classe Livro\_NA e escreva o código das linhas 46 a 64 da Figura 6.1.

```

46 public void SetIdentificadorUnico_NA(string identificadorUnico) {
47     identificadorUnico_NA = identificadorUnico;
48 }
49
50 public void SetAutor_NA(string autor) {
51     autor_NA = autor;
52 }
53
54 public void SetTitulo_NA(string titulo) {
55     titulo_NA = titulo;
56 }
57
58 public void SetPreco_NA(decimal preco) {
59     preco_NA = preco;
60 }
61
62 public void SetExistencias_NA(int existencias) {
63     existencias_NA = existencias;
64 }
    
```

Figura 6.1: Métodos modificadores (seters) para os campos (atributos) da classe Livro\_NA



## Tutorial 2 – Transição java -> C# (Classes e Objetos)

### 6.2 Praticar a codificação de métodos Modificadores (Seters)

28. Por analogia com o que fez no passo 27 crie métodos modificadores (Seters) para todos os campos das classes Cd\_NA e Dvd\_NA.

**NOTE QUE:** os modificadores são normalmente públicos, não retornam nada (void) e aceitam um argumento do mesmo tipo do campo (atributo) que irão alterar. Sendo públicos, para respeitar a notação do C# escrevem-se em PascalCase

### 6.3 Utilização (invocação) de métodos Modificadores (Seters)


Tratando-se de métodos instância a invocação de métodos modificadores exige a prévia criação de um objeto da classe e em seguida a invocação do método para esse objeto através do **operador ponto**.

29. Para alterar os valores dos campos dos objetos livro1\_NA e livro2\_NA **escreva o código das linhas 77 a 100 da Figura 6.2,**

```

74 Console.WriteLine("\nprima return para continuar");
75 Console.ReadKey();
76
77 livro1_NA.SetTitulo_NA("Os Lusíadas");
78 livro1_NA.SetAutor_NA("Luís Vaz de Camões");
79 livro1_NA.SetPreco_NA(30.99M);
80 livro1_NA.SetExistencias_NA(1);
81
82 livro2_NA.SetPreco_NA(35.99M);
83 livro2_NA.SetExistencias_NA(5);
84 livro3_NA.SetPreco_NA(20.89M);
85 livro3_NA.SetExistencias_NA(1);
86
87 Console.WriteLine("\n*** LIVRO: " + livro1_NA.GetIdentificadorUnico_NA()
88     + "\n* " + livro1_NA.GetTitulo_NA()
89     + "\n* " + livro1_NA.GetAutor_NA()
90     + "\n* " + livro1_NA.GetPreco_NA()
91     + "\n* " + livro1_NA.GetExistencias_NA());
92 Console.WriteLine("\n*** LIVRO: " + livro2_NA.GetIdentificadorUnico_NA()
93     + "\n* " + livro2_NA.GetPreco_NA()
94     + "\n* " + livro2_NA.GetExistencias_NA());
95 Console.WriteLine("\n*** LIVRO: " + livro3_NA.GetIdentificadorUnico_NA()
96     + "\n* " + livro3_NA.GetPreco_NA()
97     + "\n* " + livro3_NA.GetExistencias_NA());
98
99 Console.WriteLine("\nprima return para continuar");
100 Console.ReadKey();
    
```

**Figura 5.2: Alteração dos valores de alguns dos campos e dois objetos e saída para a consola dos valores dos campos modificados.**

30. Para compilar e executar o programa prima o botão  da barra de ferramentas e verifique se obtém um resultado semelhante ao da Figura R6.1, caso contrário corrija o código que escreveu e volte a correr o programa.

### 6.4 Praticar a utilização (invocação) de métodos Modificadores (Seters)

31. Por analogia com o código que escreveu no ponto 29, modifique os valores dos campos dos objetos cd1\_NA e dvd1\_NA e veja os valores na consola.

## Tutorial 2 – Transição java -> C# (Classes e Objetos)

### 7 O Método ToString ( )

O método ToString é muito utilizado para confirmar os valores dos campos em tarefas de eliminação de erros (debugging).

**NOTE QUE:** Se diferencia do toString do Java por, sendo público, o seu identificador começa com uma maiúscula (PascaCase) e ter de ser antecedido da clausula **override**, que no java é opcional.

#### 7.1 Codificação de método ToString ( )

32. Abra a janela com a classe Livro\_NA e escreva o código das linhas 40 a 44 da Figura 7.1.

```

66      override
67      public String ToString()
68      {
69          String tmp = "**** LIVRO: " + identificadorUnico_NA
70                  + " | Tit: " + titulo_NA
71                  + " | Aut: " + autor_NA
72                  + " | $$: " + preco_NA
73                  + " | Stock: " + existencias_NA;
74          return tmp;
75      }
    
```

Figura 7.1: Método ToString ( ) da classe Livro\_NA

#### 7.2 Praticar a codificação do método ToString ( )

33. Por analogia com o que fez no ponto 32 crie métodos ToString ( ) para as classes Cd\_NA e Dvd\_NA.

#### 7.3 Utilização (invocação) do método ToString ( )

Tal como qualquer outro método de instância, a invocação do ToString ( ) exige a prévia criação de um objeto da classe e em seguida a invocação do método para esse objeto através do **operador ponto** .

34. Para imprimir na consola os valores dos campos dos objetos livro1\_NA, livro2\_NA e livro3\_NA usando o ToString ( ) **escreva o código das linhas 77 a 100 da Figura 7.2.**

35. Verifique se o resultado obtido é igual ao da Figura R7.1 e, se for o caso, encontre e resolva os erros no programa que conduziram a esse erro “Semântico”, explique porque é que ele teve lugar e descreva a técnica que aprendeu no semestre passado em POO para evitar esse tipo de erros.

```

102      Console.WriteLine(livro1_NA
103                      + "\n" + livro2_NA
104                      + "\n" + livro3_NA);
    
```

Figura 7.2: Invocação no Main do método ToString ( ) dos objetos livro1\_NA a livro3\_NA

#### 7.1 Praticar a utilização (invocação) do método ToString ( )

36. Por analogia com o código que escreveu no ponto 34, use os ToString que codificou no ponto 33 para imprimir na consola os valores dos campos dos objetos Cd\_NA e Dvd\_NA.

### 8 Saídas na consola

```
Fall
2
Fall
5
Fall
2
Fall
7

prima return para continuar
```

Figura R1.1

```
*****
LIVROS EM STOCK:

*** LIVRO: IU_000000000
* Sem Título Atribuído
* Sem Autor Definido
* 0.0
* 0

*** LIVRO: IU_000000001
* José Saramago
* Memorial do Convento
* 25.5
* 3

prima return para continuar
```

Figura R5.1

```
prima return para continuar

*** LIVRO: IU_000000000
* Os Lusíadas
* Luis Vaz de Camões
* 30.99
* 1

*** LIVRO: IU_000000001
* 35.99
* 5

*** LIVRO: IU_000000002
* 20.89
* 1
```

```
prima return para continuar
```

Figura R6.1

*** LIVRO: IU_000000000	Tit: Os Lusíadas	Aut: Luis Vaz de Camões	\$\$: 30.99	Stock: 1
*** LIVRO: IU_000000001	Tit: José Saramago	Aut: Memorial do Convento	\$\$: 35.99	Stock: 5
*** LIVRO: IU_000000002	Tit: Eça de Queirós	Aut: O Conde de Abranhos	\$\$: 20.89	Stock: 1

Figura R7.1

# The End

.

.

# My Friend

# The End