FICHA 5

CLASSES

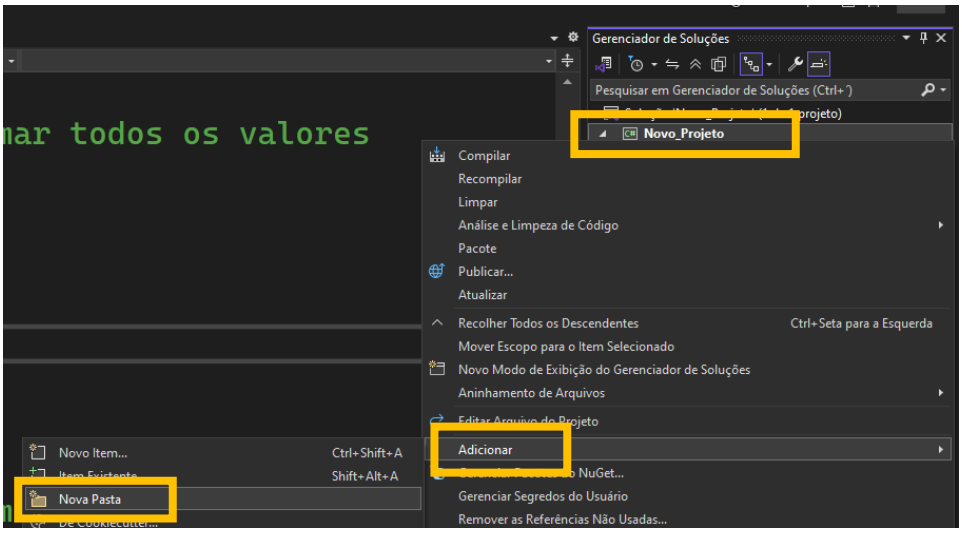
MODALIDADE:	Aprendizagem +	Escolha um item.	
CURSO:	Tecnologias de Programação de Sistemas de Informação		
UC:	Desenvolver programas complexos em linguagem de programação estruturada	CÓDIGO UC:	UC00607
FORMADOR/A:	Bruno Silva	DATA:	

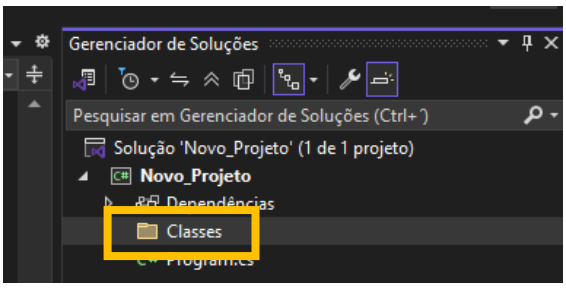
OBJETIVOS

- Perceber a implementação das classes com a linguagem C#

Uma ***classe*** é um conceito que define uma ***estrutura para a criação de objeto***. Uma classe ***define um objeto com base*** nas suas propriedades (os campos ou variáveis) e as suas ações ou métodos que definem o seu comportamento (funções).

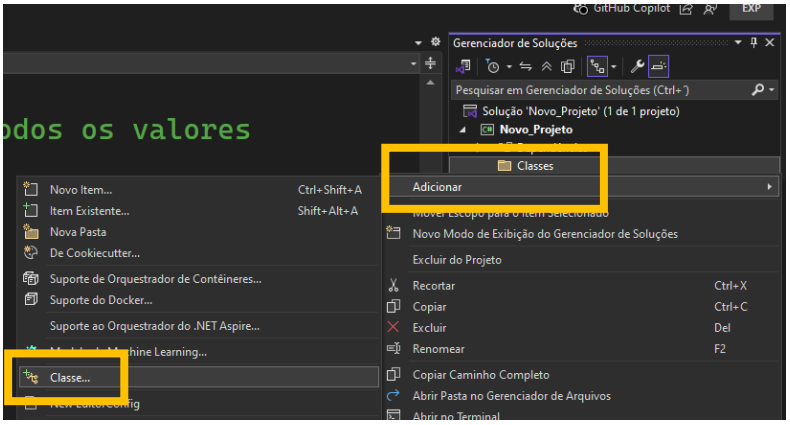
Para contruir a classe do exercício mais abaixo (e posteriormente as próximas classes), vamos criar uma pasta para colocar o conteúdo e fazer a separação dos outros ficheiros:



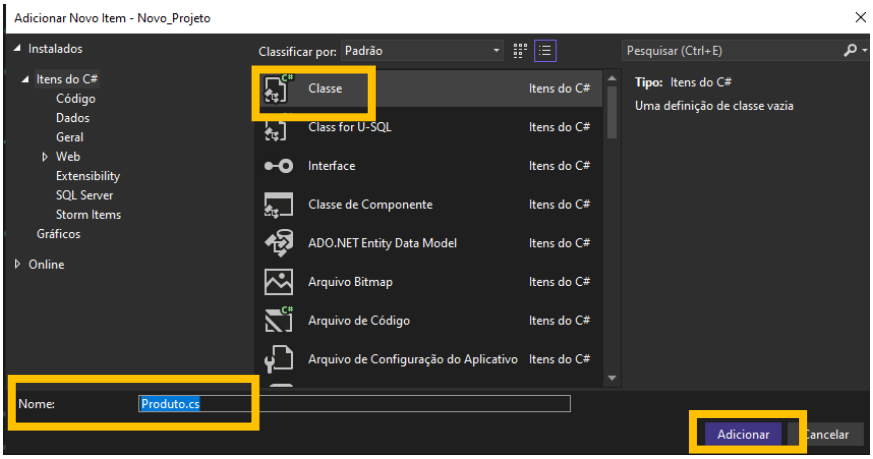


Construção de uma classe

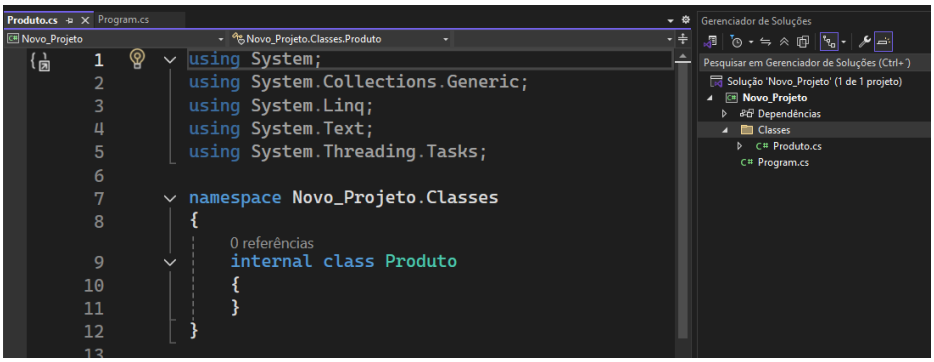
Passo 1 – Na nova pasta, vamos criar um novo ficheiro do tipo Classe:



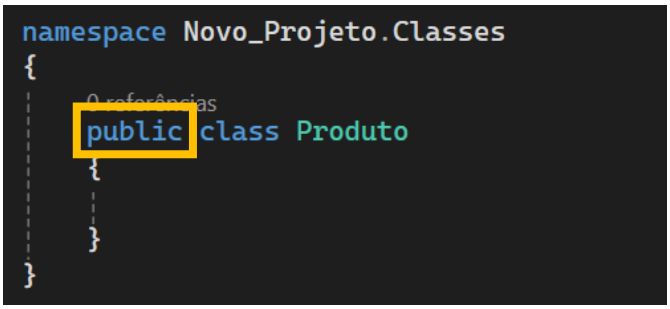
Passo 2 – De seguida, vai ser exibida a página de criação de ficheiros e recursos do C# e com a opção classe pré-selecionada. Mais abaixo, vamos dar o nome da classe, que neste exemplo, será o Produto. Se reparar com atenção, o programa já coloca por defeito a extensão .cs para que o sistema e o projeto saber que aquele ficheiro e do tipo de programação C#:



Passo 3 – No final, será construído o novo ficheiro dentro da pasta e com a seguinte estrutura:



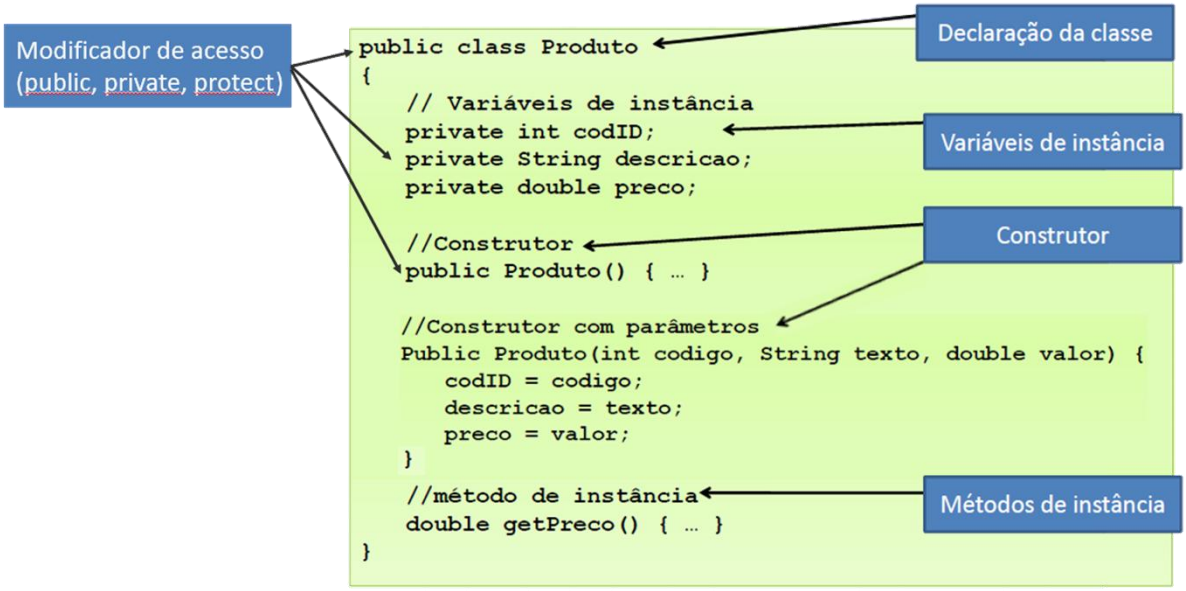
Passo 4 – Para deixar a classe como pública e disponível em todo o projeto, alterar o modificador de acesso de **internal** para **public**:



- A ***declaração de classes*** em C# devem ***começar com letra Maiúscula***;
- O ficheiro onde está a definição da ***classe tem de ter o mesmo nome da classe*** e com ***extensão*** ".cs" (como já referido anteriormente);
- As palavras ***public*** / ***private*** são ***modificadores de acesso*** que ***controlam*** que parte do programa ***pode ou não utilizar a classe***.
  - ***Os modificadores de acesso controlam*** que parte do programa ***pode ou não utilizar a classe***.

    Estes podem ser de *3 tipos*:
    - ***Public*** (público) - é o nível de acesso que os atributos ou métodos podem ser acedidos em qualquer outro ponto do projeto e por outras classes;
    - ***Private*** (privado) - é o nível de acesso mais restrito e permite definir que somente a própria classe pode aceder aos atributos ou métodos. Ou seja, nenhuma outra parte do projeto, nem mesmo as classes filhas (heranças), podem aceder a esse atributo ou método;
    - ***Protected*** (protegido) – nível de acesso protegido que somente a própria classe e as classes que herdam da mesma é que podem aceder os atributos ou métodos. Dessa forma, ao instanciar a classe os elementos protegidos (protected) não podem ser acedidos diretamente, como ocorre com o public;

Exemplo de uma classe:



Variáveis de instância

As **variáveis de instância** permitem representar as características de uma classe:

- A ***declaração de variáveis de instância*** em C# é feita ***dentro do bloco da definição de classe*** e ***nunca*** dentro de ***métodos*** ou ***construtores***;
- Para ***prevenir*** que outras classes ***possam aceder às variáveis de instância***, devemos utilizar a palavra reservada ***private*** antes da declaração.
- Neste exemplo, temos a classe **Produto** que vai ter 3 características: o ID, uma descrição e um preço;

```
0 referências
public class Produto
{
    //variaveis de instância
    private int codID;
    private string descricao;
    private double preco;
```

Construtores

O ***construtor é um método especial***, por norma é o ***primeiro "método"*** que se ***constrói dentro da classe***. Não são métodos de instância porque apenas são utilizados para **criar instâncias/objetos de uma dada classe**;

Nos construtores ***vamos ter parâmetros de entrada no construtor*** e que vai ***receber/passar a informação dos parâmetros de entrada*** para as ***variáveis de instância*** da classe:

```
//Construtor com parâmetros
Public Produto(int codigo, String texto, double valor) {
    codID = codigo;
    descricao = texto;
    preco = valor;
}
```

Se não especificarmos um construtor, o C# ***cria um construtor por defeito*** que ***inicializa todas as variáveis de instância*** para ***valores por defeito*** (numéricos a 0, boolean a false/true e objetos a null)

- Utilizamos a palavra reservada ***this*** para ***identificar*** o que é a **variável de instância do parâmetro** da informação **do parâmetro de entrada do construtor**;
    - A palavra reservada ***this*** permite ter acesso a partir do código do objeto às ***variáveis***, ***métodos*** e ***construtores*** do próprio objeto.
    - A palavra reservada ***this*** é útil por exemplo, quando as variáveis dos parâmetros de entrada de um método são coincidentes com as variáveis de instância.

```
1 referência
public class Produto
{
    //variaveis de instância
    private int codID;
    private string descricao;
    private double preco;

    //construtor com parametros de entrada
    0 referências
    public Produto(int codID, string descricao, double preco)
    {
        this.codID = codID;
        this.descricao = descricao;
        this.preco = preco;
    }
}
```

Métodos de instância: Get e Set

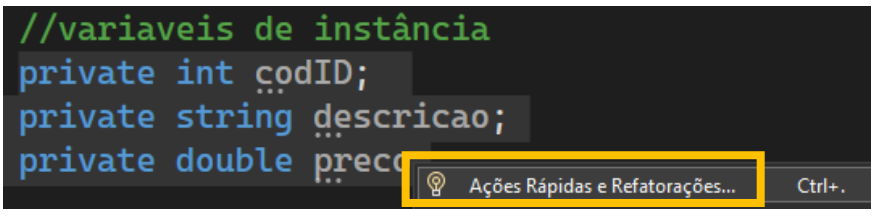
- ***Métodos de instância*** são responsáveis ***pelo comportamento das instâncias/objetos quando recebem/mostram os(as) valores/mensagens provenientes de alguma parte do programa***;
- ***Para que outras classes tenham acesso ao método*** é preciso designar o método com acesso público com a ***palavra reservada public***;
- ***Um método que não retorne qualquer tipo de dado*** deve ser colocada a palavra reservada ***void*** no ***tipo de dados de retorno***;
- Utilizamos a palavra reservada ***this*** para ***identificar*** o que é a **variável de instância do parâmetro** da informação **do parâmetro de entrada do construtor**;
    - A palavra reservada ***this*** permite ter acesso a partir do código do objeto às ***variáveis***, ***métodos*** e ***construtores*** do próprio objeto.
    - A palavra reservada ***this*** é útil por exemplo, quando as variáveis dos parâmetros de entrada de um método são coincidentes com as variáveis de instância.

```
public void setCodID(int codID) {
        this.codID = codID; //codID = codID não faz sentido
}
```

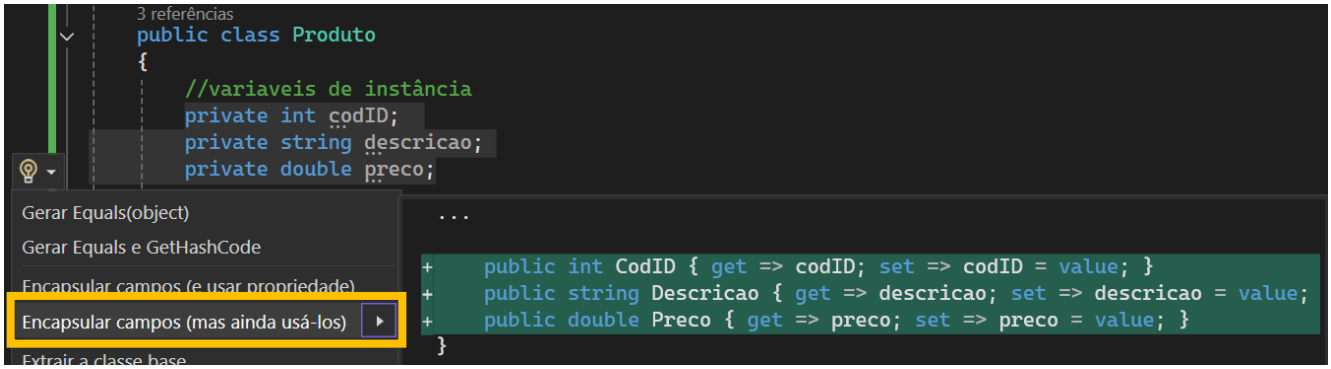
- É boa prática na codificação das classes C# construir métodos de instância ***set*** e ***get*** para ***cada variável de instância*** que existe definida na classe, em que:
    - O ***método get*** é utilizado para obter as informações das variáveis de instância do objeto criado a partir da classe;
    - O ***método set*** é utilizado para receber e alterar as informações das variáveis de instância do objeto criado a partir da classe;

Para **gerar** os **gets** e **sets** para cada variável de instância, vamos fazer os seguintes passos:

- **Passo 1** – **Selecione todas as variáveis de instância da classe** e **clique** em cima da seleção com **o lado direito do rato**. Será exibido o menu de opções e **deve selecionar a opção "Ações rápidas e Refatorações"**:



- **Passo 2 –** De seguida, selecione a opção " Encapsular campos (mas ainda usá-los)":



Resultado Final:

```
3 referências
public class Produto
{
    //variaveis de instância
    private int codID;
    private string descricao;
    private double preco;

    //construtor com parametros de entrada
    1 referência
    public Produto(int codID, string descricao, double preco)
    {
        this.codID = codID;
        this.descricao = descricao;
        this.preco = preco;
    }

    //métodos de instância (gerados a partir das variáveis de instância)
    0 referências
    public int CodID { get => codID; set => codID = value; }
    2 referências
    public string Descricao { get => descricao; set => descricao = value; }
    0 referências
    public double Preco { get => preco; set => preco = value; }
}
```

Métodos adicionais

Podemos criar **métodos adicionais** para designar comportamentos específicos, como por exemplo, **mostrar todos os dados das variáveis de instância de um objeto** criado com a classe Produto:

```
//Métodos adicionais

//Exemplo: mostrar todos os dados das variáveis de instância para um objeto
1 referência
public string Mostra_Produto()
{
    return "ID: " + this.codID + ", Descrição: " + this.descricao + ", Preço: " + this.preco;
}
```

Exemplo da classe Produto (Completo):

```
3 referências
public class Produto
{
    //variaveis de instância
    private int codID;
    private string descricao;
    private double preco;

    //construtor com parametros de entrada
    1 referência
    public Produto(int codID, string descricao, double preco)
    {
        this.codID = codID;
        this.descricao = descricao;
        this.preco = preco;
    }

    //métodos de instância (gerados a partir das variáveis de instância)
    0 referências
    public int CodID { get => codID; set => codID = value; }
    2 referências
    public string Descricao { get => descricao; set => descricao = value; }
    0 referências
    public double Preco { get => preco; set => preco = value; }

    //Métodos adicionais

    //Exemplo: mostrar todos os dados das variáveis de instância para um objeto
    1 referência
    public string Mostra_Produto()
    {
        return "ID: " + this.codID + ", Descrição: " + this.descricao + ", Preço: " + this.preco;
    }
}
```

Instância de Classes: Criação de novos objetos a partir de uma Classe

Quando pretendemos usar as classes para criar objetos e armazenar informação, temos de saber como invocar as mesmas. Como tal, as classes são tipos de referência para uma variável de um objeto de uma classe que contém uma referência ao endereço do objeto.

Desta forma, temos de fazer a criação da instância de uma classe para uma variavel, por forma a criar um novo objeto a partir dessa classe e colocar as informações pedidas no construtor dessa classe.

Para realizar as instâncias de classes utilizamos a palavra reservada **new**.

- A palavra reservada ***new*** trata-se de um ***operador que vai instanciar a classe***, ***alocando para isso espaço em memória para o objeto*** e retorna a referência/endereço desse objeto.
- ***Exemplo***: Vamos criar uma instância de uma classe, isto é, um novo objeto.

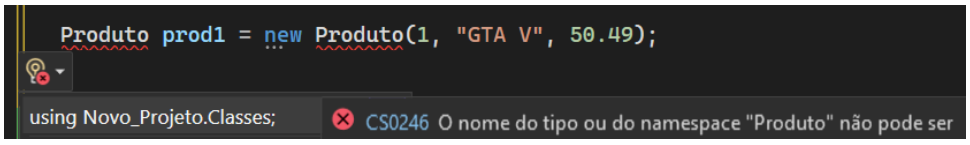
  Empregado eng1 = ***Empregado***("Magda Dionísio", 25);

- O operador ***new*** também tem a responsabilidade de:
  - ***invocar o construtor da classe;***
  - ***para inicializar um objeto;***

Exemplo da criação de uma variável com base na classe Produto:

```
Produto prod1 = new Produto(1, "GTA V", 50.49);
```

Nota Importante: Se a palavra que designa como classe estiver a dar erro, significa que a classe na está no mesmo sítio do ficheiro do ficheiro onde está a testar a criação do objeto. Como tal, temos de importar o local ou pasta de onde está o ficheiro da classe:

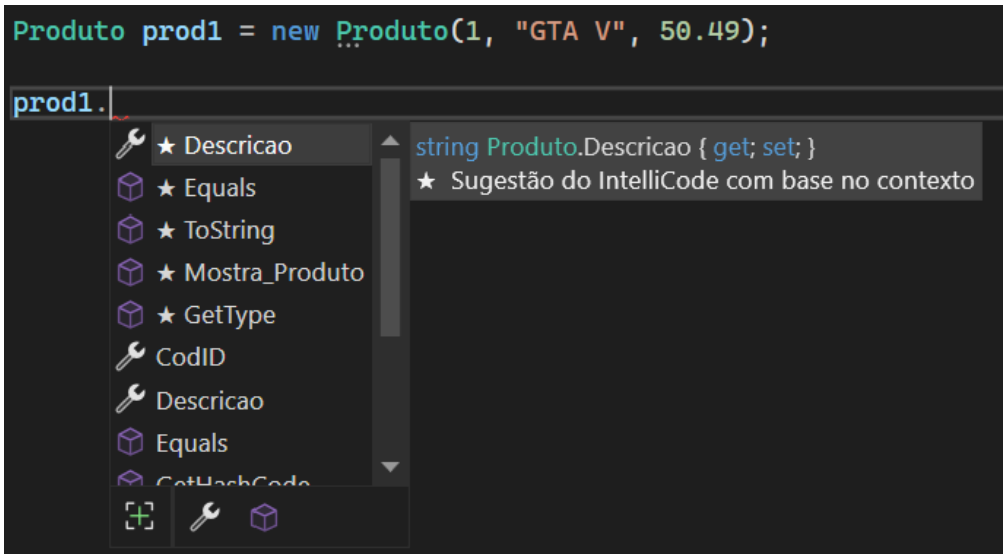


Resultado da importação automática:

```
using Novo_Projeto.Classes;

Produto prod1 = new Produto(1, "GTA V", 50.49);
```

Depois de criar o objeto em causa, basta invocar o nome da variável baseada na classe e poderá aceder às propriedades da mesma (como tal, basta colocar o símbolo **. (ponto final)** depois do nome da variável para obter a informação da classe):



Podemos assim fazer várias operações, como por exemplo:

Mostrar a informação de uma variável de instância do objeto (pelos métodos de instância **get**):

```
Produto prod1 = new Produto(1, "GTA V", 50.49);

Console.WriteLine("Valor do ID do prod1: " + prod1.CodID);
Console.WriteLine("Valor da descrição do prod1: " + prod1.Descricao);
```

```
Console de Depuração do Microsoft Visual Studio
Valor do ID do prod1: 1
Valor da descrição do prod1: GTA V
```

Alterar a informação de uma variável de instância do objeto (pelos métodos de instância **set**):

```
Produto prod1 = new Produto(1, "GTA V", 50.49);

//Alterar o valor da descricao do prod1
prod1.Descricao = "Super Mario";

Console.WriteLine("Valor da descrição do prod1: " + prod1.Descricao);
```

```
Console de Depuração do Microsoft Visual Studio
Valor da descrição do prod1: Super Mario
```

Mostrar a informação de todas as variáveis de instância do objeto com o método Mostra_Produto:

```
Produto prod1 = new Produto(1, "GTA V", 50.49);

Console.WriteLine("Dados prod1: " + prod1.Mostra_Produto());
```

