



Programação Orientada a Objetos em C#

Aula 04 (21/03/2019)



COTI Informática
Escola de Nerds

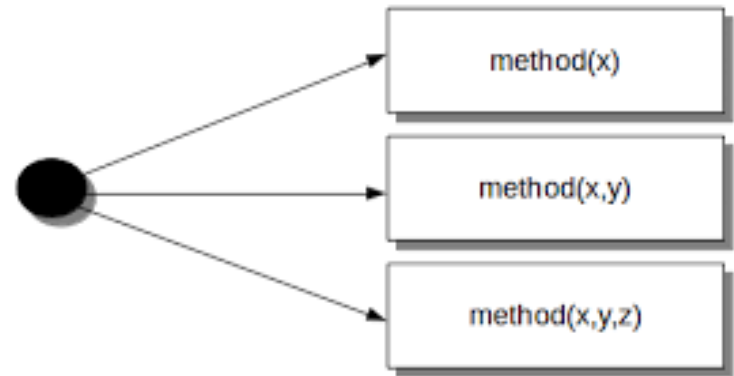
Sobrecarga de Métodos (Overloading)

- Recurso de programação orientada a objetos que permite ao desenvolvedor declarar métodos em uma classe com o mesmo nome, porem com entrada de argumentos diferentes.

Por exemplo:

```
public class Calculo
{
    public double Somar(double a, double b)
    {
        return a + b;
    }

    public double Somar(double a, double b, double c)
    {
        return a + b + c;
    }
}
```



Sobrescrita de Métodos (**Override**)



- Recurso de programação orientada a objetos que permite ao desenvolvedor em uma classe "filha" (subclasse) reprogramar métodos herdados de sua "superclasse"

```
public class A
{
    public virtual void Imprimir()
    {
        Console.WriteLine("Imprime A");
    }
}
```

```
public class B : A
{
    public override void Imprimir()
    {
        Console.WriteLine("Imprime B");
    }
}
```

virtual

É uma palavra reservada da linguagem utilizada para permitir que um método seja sobrescrito por uma subclasse.

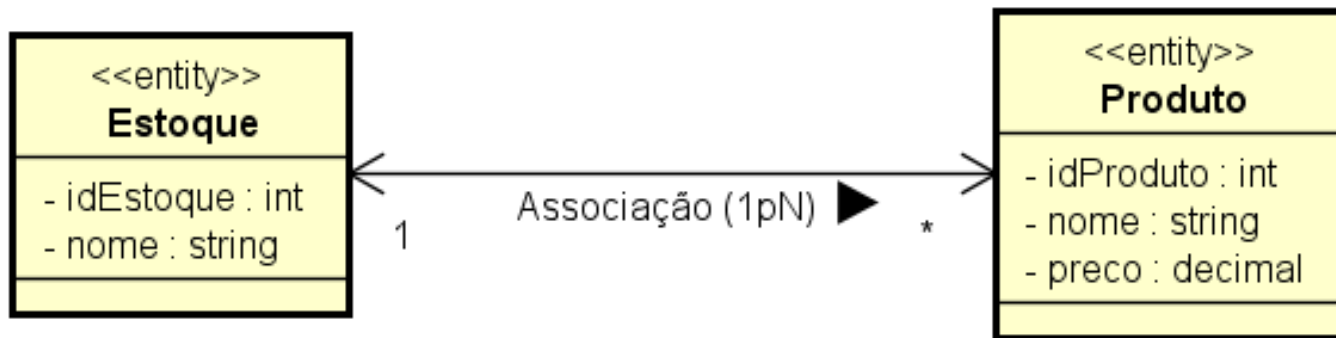
Para que haja sobrescrita de método, a superclasse sempre deverá declarar o método que seja permitir sobrescrever como "virtual"

Relacionamento de Associação

Associação (TER)

É um relacionamento baseado na abstração do verbo TER, representa o conceito de utilização (TODO/PARTE) ao invés de herança.

Exemplo:

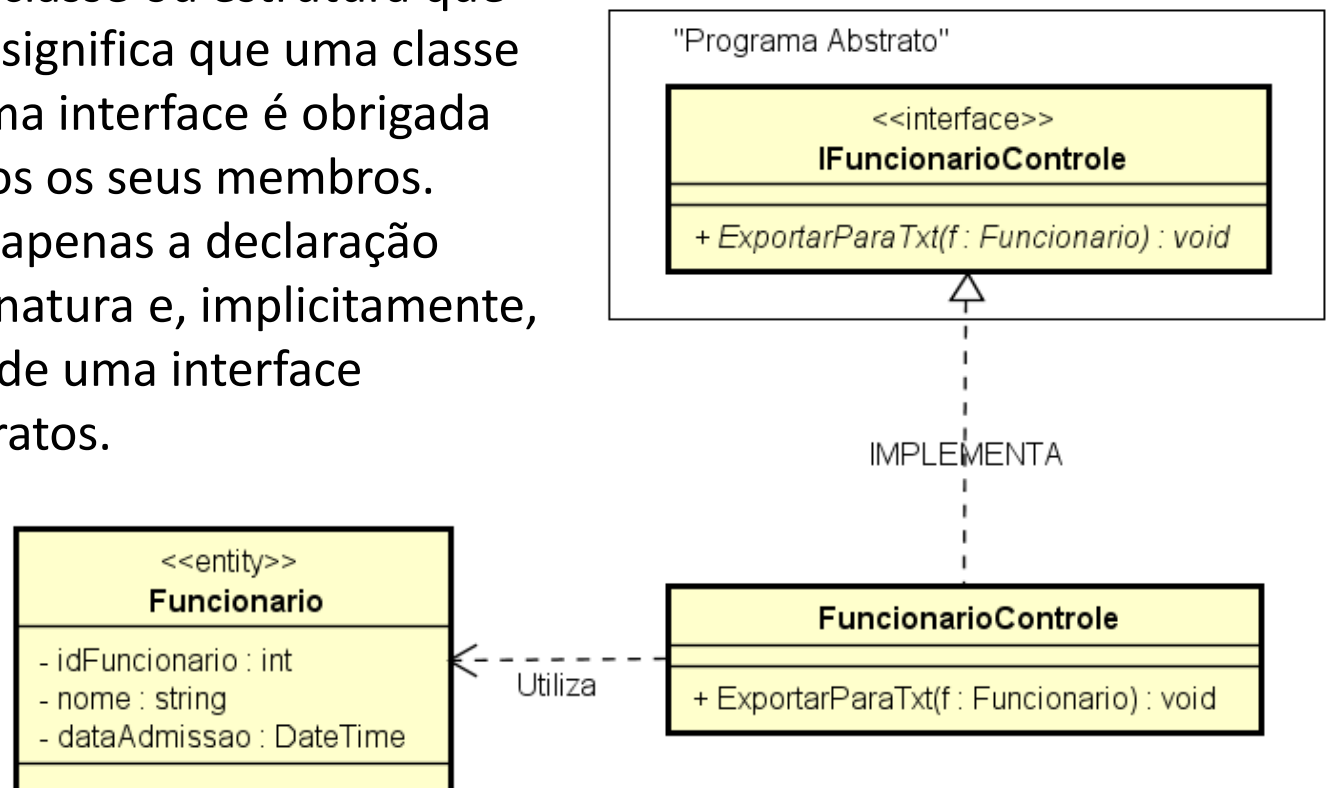


0..1	No mínimo zero e no máximo 1
1	1 e somente 1
0..*	No mínimo zero e no máximo muitos
1..*	No mínimo 1 e no máximo muitos
*	Muitos

Interfaces

Interface é uma ferramenta de programação orientada a objetos utilizado para definir padrões (contratos) que as classes deverão implementar.

Uma interface funciona como um contrato entre si e qualquer classe ou estrutura que a implementa. Isso significa que uma classe que implementa uma interface é obrigada a implementar todos os seus membros. Uma Interface tem apenas a declaração de membro ou assinatura e, implicitamente, todos os membros de uma interface são públicos e abstratos.

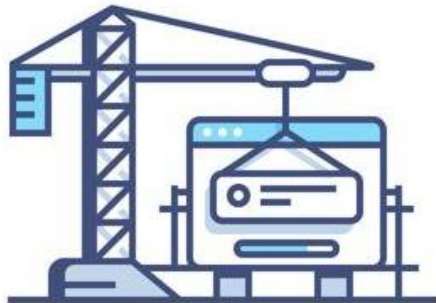


Regras sobre interfaces:

- Interfaces não podem ter atributos
- Interfaces não podem ter construtores
- Métodos de interface devem ser abstratos, ou seja, não podem ter corpo, apenas assinatura.
- Métodos de interface já são implicitamente públicos.

Regra principal:

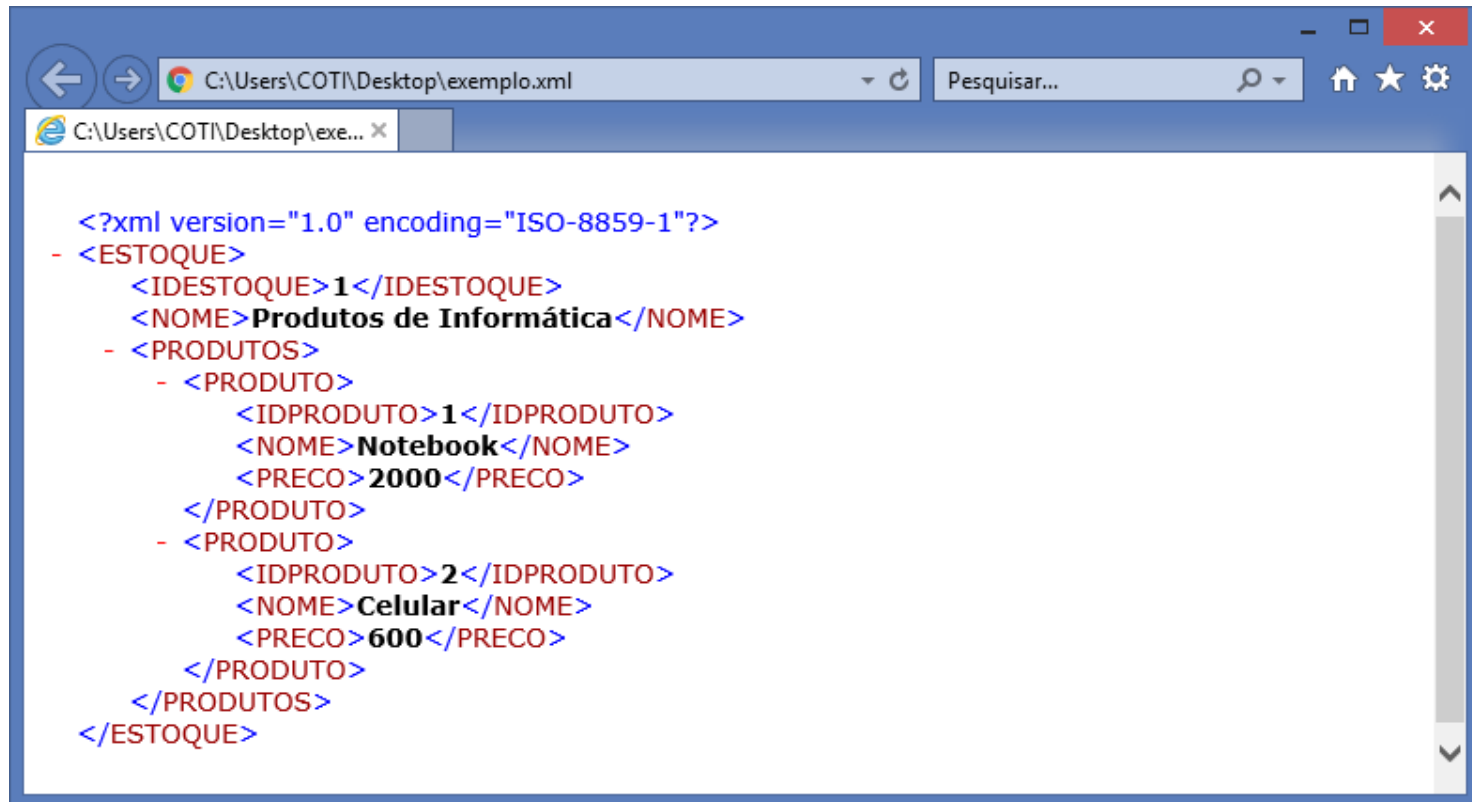
Quando uma classe HERDA uma interface, a classe é obrigada a implementar (fornecer corpo) para todos os métodos abstratos da interface.



Formato XML

eXtensible Markup Language

Formato para representação e armazenamento de dados baseado em TAGS e que é utilizado comumente por aplicações web que precisam realizar troca de dados entre si.

A screenshot of a web browser window showing an XML file. The address bar indicates the file path 'C:\Users\COTI\Desktop\exemplo.xml'. The browser's address bar also contains a search field labeled 'Pesquisar...'. The XML content is displayed with syntax highlighting: opening and closing tags are in blue, and text values are in red. The XML structure is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <ESTOQUE>
  <IDESTOQUE>1</IDESTOQUE>
  <NOME>Produtos de Informática</NOME>
  - <PRODUTOS>
    - <PRODUTO>
      <IDPRODUTO>1</IDPRODUTO>
      <NOME>Notebook</NOME>
      <PRECO>2000</PRECO>
    </PRODUTO>
    - <PRODUTO>
      <IDPRODUTO>2</IDPRODUTO>
      <NOME>Celular</NOME>
      <PRECO>600</PRECO>
    </PRODUTO>
  </PRODUTOS>
</ESTOQUE>
```

Tratamento de Exceções



São erros que podem ocorrer em um programa não durante a sua compilação mas sim durante a sua execução.

Estes erros em tempo de execução são chamados de **Exceções**. Para tratarmos estas exceções podemos utilizar um bloco de programação denominado **try / catch**

Exception

Classe que representa qualquer tipo de erro ocorrido em tempo de execução
(qualquer tipo de exceção.)

Hierarquia de Exceções

