



{ Engenharia
de Software
4.0 }



Curso de Programação em C#

Aula 02 - Programação Orientada a Objetos

17/03/23



Veridiano Barroso
VERIDIANO.FILHO@UFAC.BR



Apoio



Aula 2 - Programação Orientada a Objetos



Classe Fruta

Objetos Banana, Maça, Uva

Aula 2 - Programação Orientada a Objetos



Watch on YouTube

Aula 2 - Programação Orientada a Objetos

Carro	
Nome	Cannibal
Cor	Vermelho
Modelo	Ferrari 512 TR

Carro	
Nome	Razo
Cor	lilás
Modelo	Jaguar XJR-15

Carro	
Nome	Sidewinder
Cor	Branco
Modelo	Ferrari 288 GTO

Carro	
Nome	Yoz
Cor	azul
Modelo	Porsche 959

Aula 2 - Programação Orientada a Objetos

Abstração

Classe



```
public class Carro
```

```
{
```

```
    int id;  
    string nome;  
    double vel_max;  
    int marchas;
```

Métodos



```
public void trocarPneu(){
```

```
    ....  
}
```

```
}
```

```
Carro Jogador1 = new Carro()
```

Atributos



Objetos
Construtor

Exemplo 3: classe Carro

Criar uma console application e uma classe **Carro** com os seguintes atributos “**id**”, “**Nome**” e “**VeloxMax**”.”.

No Main crie uma **instância** de **Carro** atribuindo valor aos atributos;

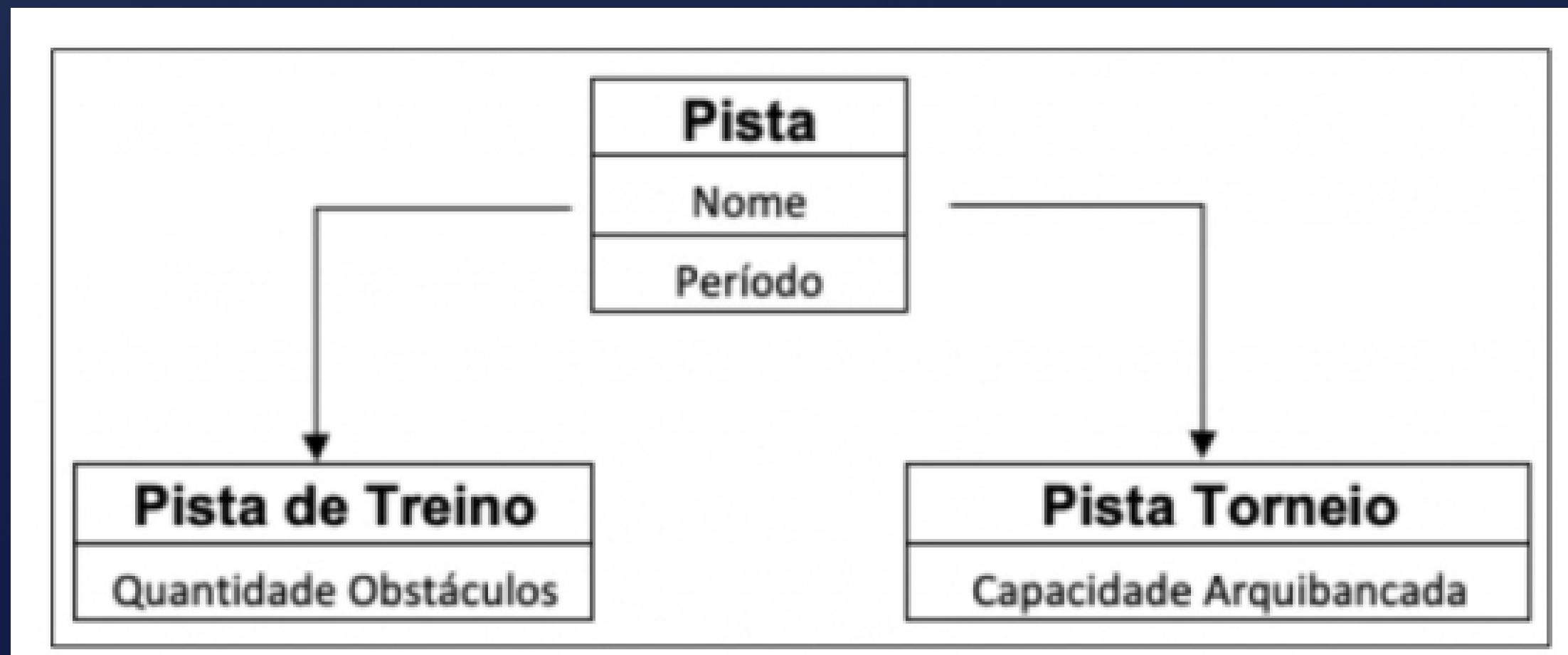
Crie um método void para **alterar** a velocidade máxima do objeto **Carro**.

De volta ao Main exiba no terminal a velocidade **alterada**.

Aula 2 - Programação Orientada a Objetos

Herança

Objetivo de reaproveitar o código



Exemplo 4: classe Pista

Criar uma console application e uma classe **Pista** com os seguintes atributos “**Nome**”, “**Clima**”. E uma classe **Treino**, herdada de **Pista** com o atributo **NumBarreiras**.

No Main crie uma **instância** de **Pista** e **Treino** atribuindo valor aos atributos;

De volta ao **Main** exiba seus valores no terminal.

Na orientação a objetos, esconder os detalhes de implementação de uma classe é um conceito conhecido como **encapsulamento**



Exemplo Conta

Aula 2 - Programação Orientada a Objetos

Encapsulamento

Public – deixa visível a classe ou membro para todas as outras classes, subclasses e pacotes do projeto, ou namespaces do projeto.

Private – deixa visível o atributo apenas para a classe em que este atributo se encontra.

Protected – deixa visível o atributo para todas as outras classes e subclasses que pertencem ao mesmo pacote ou namespace. O protected é um intermediário entre public e private.

Publico VS Private

- **Public** todas as classes podem usar os métodos e campos.
- **Private** apenas a classe podem usar os métodos e campos.

Por que Controlar o Acesso?

- Proteger informação privada.
- Esclarecer como outros programadores devem usar sua classe.
- Manter a implementação separado da interface.

Aula 2 - Programação Orientada a Objetos

Polimorfismo

O polimorfismo vem do grego e significa “muitas formas”.

Com o polimorfismo, podemos sobrescrever métodos das classes filhas para que se comportem de maneira diferente e ter sua própria implementação.

Em tempo de execução

- **Virtual** todas as classes podem usar os métodos e campos.
- **Override** apenas a classe podem usar os métodos e campos.

Em tempo de compilação

- NomeMetodo(a , b)
- NomeMetodo(a , b , c)

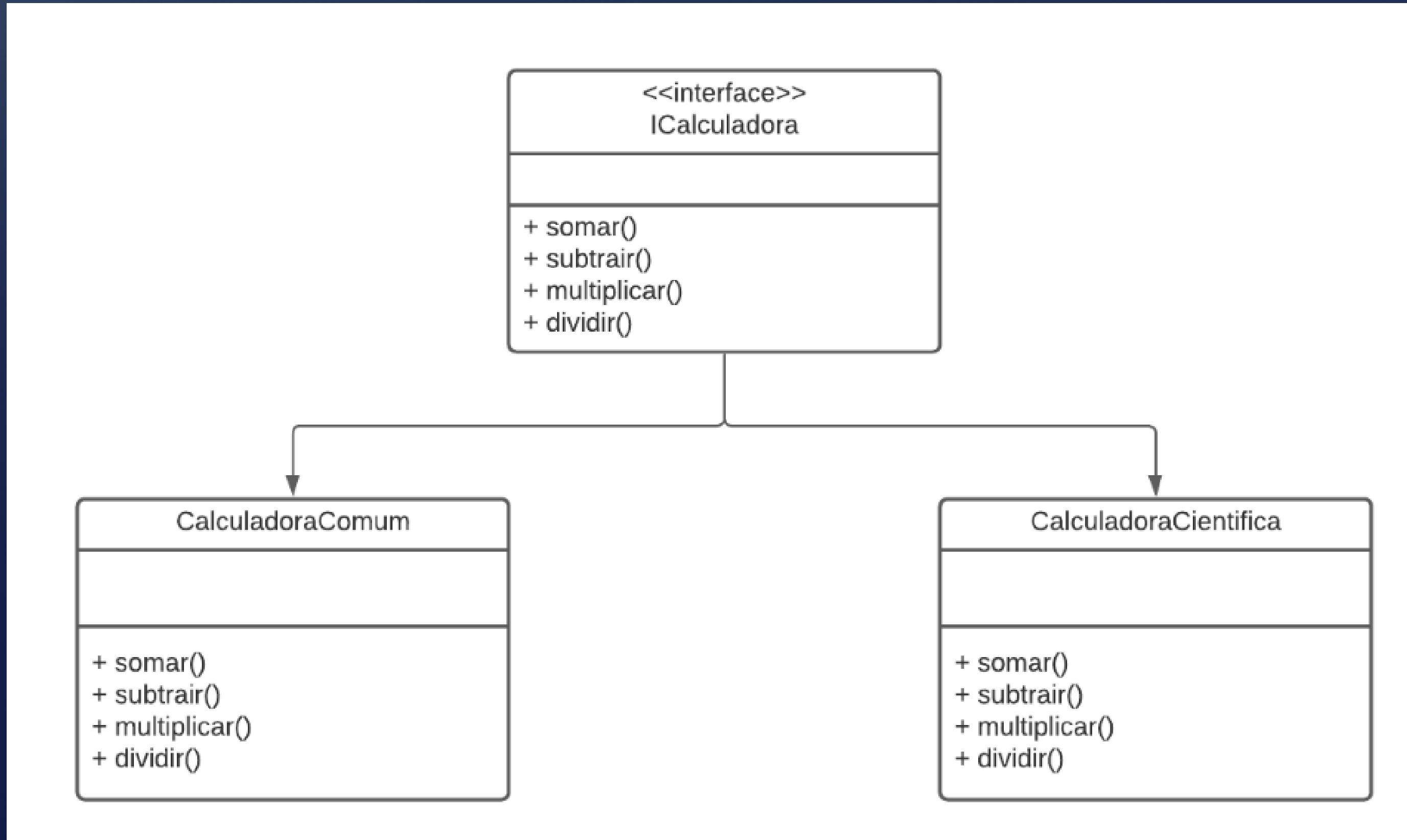
Uma interface é um contrato que pode ser implementado por uma classe.

É como se fosse uma classe abstrata, podendo definir métodos abstratos para serem implementados.

Assim como uma classe abstrata, uma interface não pode ser instanciada.

Aula 2 - Programação Orientada a Objetos

Interface



Vamos lá!

{ Engenharia
de Software
4.0 }

