



UnB

Algoritmos e Estrutura de Dados

Introdução à Orientação a Objetos

Prof. Vinícius Pereira Gonçalves

vpgvinicius@unb.br

Programação Orientada a Objetos?

3 PILARES

- Encapsulamento.
- Herança.
- Polimorfismo.



HERANÇA



Programação Orientada a Objetos?

O que é herança?

Programação Orientada a Objetos?

Herança

É comum haver similaridades entre diferentes classes.

Frequentemente, duas ou mais classes irão compartilhar os mesmos atributos e/ou métodos.

Como nenhum de nós deseja reescrever várias vezes o mesmo código, seria interessante se algum mecanismo pudesse tirar proveito dessas similaridades.

A herança é esse mecanismo.

Por intermédio da herança, é possível modelar relacionamentos do tipo "é" ou "é **semelhante**", o que nos permite reutilizar rotinas e dados já existentes.

Programação Orientada a Objetos?

Herança

A herança está relacionada às hierarquias e às relações entre os objetos.

É o mecanismo em que uma classe filha compartilha automaticamente todos os métodos e atributos de sua classe mãe.

A herança permite implementar classes descendentes implementando os métodos e atributos que se diferenciam da classe mãe.

Tipos de Herança

Simples

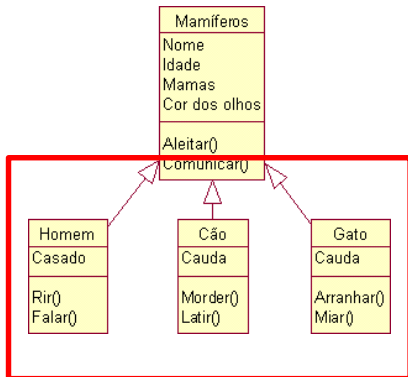
Quando uma classe herda as propriedades de uma única classe mãe.

Múltipla

Ocorre quando uma classe tem mais de uma mãe.

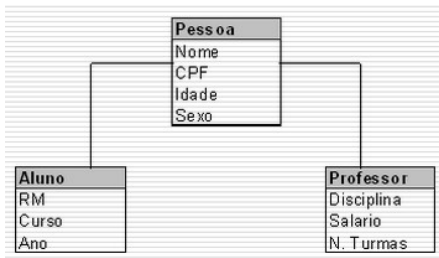
Programação Orientada a Objetos?

Herança



Programação Orientada a Objetos?

Herança



Programação Orientada a Objetos?

```
class veículo
{
    string cor;
    string combustivel;
    ...
    ...
};

class carro : public veículo
{
    int nrodas;
    ...
    ...
    int mover( int nkilometros );
};
```

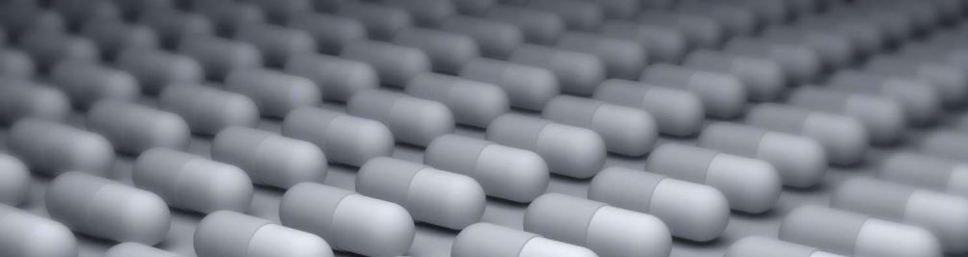
- A classe carro possui a extensão ": public veículo"
- Referindo-se a uma declaração de parentesco.
- Estamos informando ao compilador que a classe veículo é mãe da classe carro.
- A classe carro possui toda a estrutura da classe veículo além de seus próprios membros.

```

#include <iostream>
using namespace std;
class veiculo_rodoviario // Define uma classe base veiculos.
{
    int rodas;
    int passageiros;
public:
    void set_rodas(int num) { rodas = num; }
    int get_rodas() { return rodas; }
    void set_pass(int num) { passageiros = num; }
    int get_pass() { return passageiros; }
};
class caminhao : public veiculo_rodoviario // Define um caminhao.
{
    int carga;
public:
    void set_carga(int size) { carga = size; }
    int get_carga() { return carga; }
    void mostrar();
};
enum tipo {car, van, vagao};
class automovel : public veiculo_rodoviario // Define um automovel.
{
    enum tipo car_tipo;
public:
    void set_tipo(tipo t) { car_tipo = t; }
    enum tipo get_tipo() { return car_tipo; }
    void mostrar();
};
void caminhao::mostrar()

```

- Temos a classe mãe `veiculo_rodoviario` e duas classes filhas “:” `caminhao` e `automovel`.
- Características comuns estão na classe mãe.
- Características exclusivas estão nas classes filhas.
- Eventuais modificações na classe mãe são estendidas a todos os objetos criados.
- Repare que as duas classes filhas possuem o método `mostrar()`, mas uma não interfere na outra, caracterizando o polimorfismo



POLIMORFISMO



Programação Orientada a Objetos?

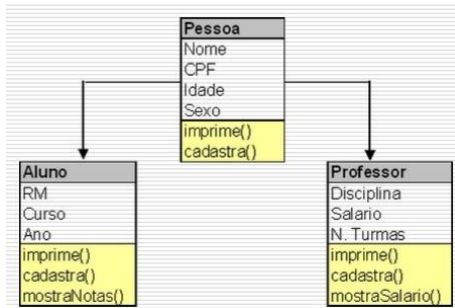
O que é polimorfismo?

Programação Orientada a Objetos?

Polimorfismo

Da mesma forma que podemos reaproveitar classes (em Herança), podemos reaproveitar métodos (através de Polimorfismo);

Ex: As classes aluno e professor, ambas possuem os métodos `imprime()` e `cadastra()`, cada um com seu objetivo, mas por estarem em classes diferentes, uma não interfere na outra. Mas o objetivo dos métodos é praticamente o mesmo.





UnB

Algoritmos e Estrutura de Dados

Introdução à Orientação a Objetos

Prof. Vinícius Pereira Gonçalves

vpgvinicius@unb.br