Programação Orientada a Objetos

Prof. Wesin Ribeiro

Métodos construtores

- Inicializa os membros da classe quando o objeto é criado
- Possui o mesmo nome da classe
- Construtores não têm tipo de retorno
- Um construtor é chamado automaticamente quando um objeto é criado
- Deve ser colocado na seção pública da classe
- Se não especificarmos um construtor, o compilador C++ gera um construtor padrão para o objeto (não espera parâmetros e tem um corpo vazio)

```
class Vetor2d{
private:
  float x, y;
public:
  Vetor2d(float x_=0, float y_=0){
    x=x_; y=y_;
  }
  void setX(float x_);
  float getX(void);
  void setY(float y_);
  float getY(void);
};

#endif
```

Métodos destrutores

- são chamados automaticamente quando o objeto está para cessar de existir na memória da máquina.
- Estratégia valiosa para realizar liberação de recursos alocados pelos construtores
- Acionar outros mecanismos necessários durante a fase de destruição
- Só pode haver um destrutor por classe
- Não recebe parâmetros
- Recebe o mesmo nome da classe precedido de ~

```
#include <iostream>

using namespace std;

class Alo{
  int id;
public:
  Alo(int id_){
    id = id_;
    cout << "Construtor de Alo: elemento " << id << "\n";
  }
  ~Alo(){
    cout << "Destrutor de Alo: elemento " << id << "\n";
  }
};

int main(void){
  Alo x(1), y(2), z(3);
}
```

Herança

O que é herança?

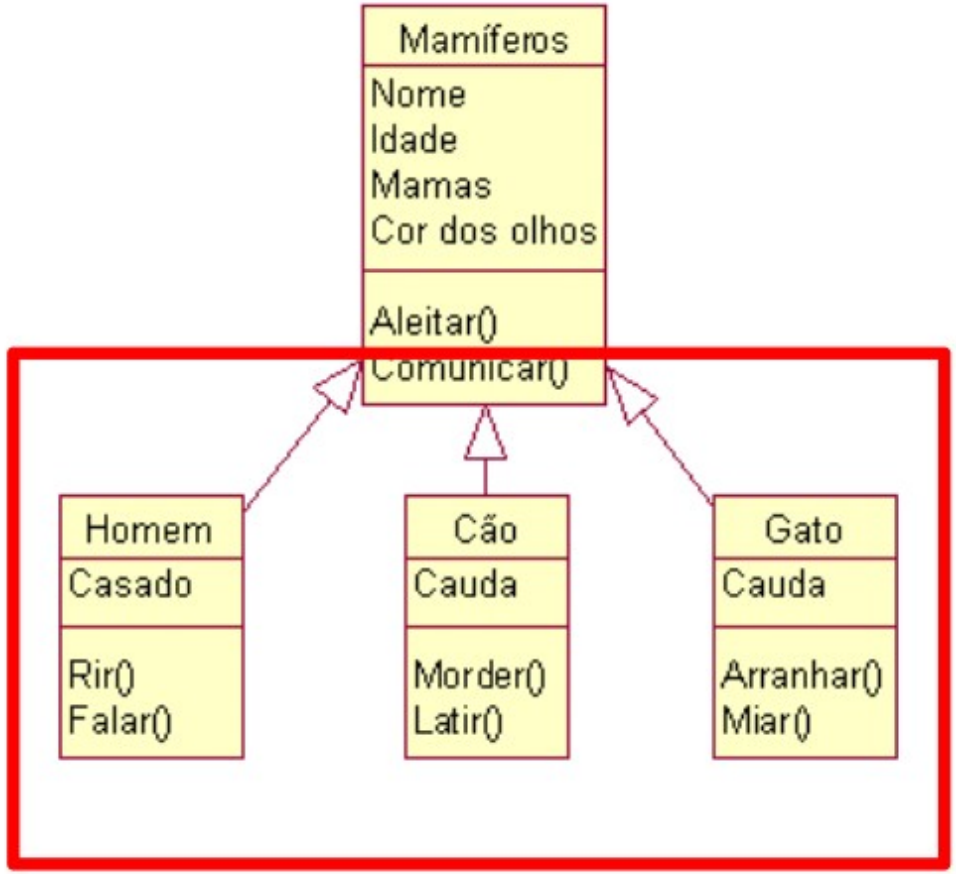
Herança é um mecanismo para reutilização de classes

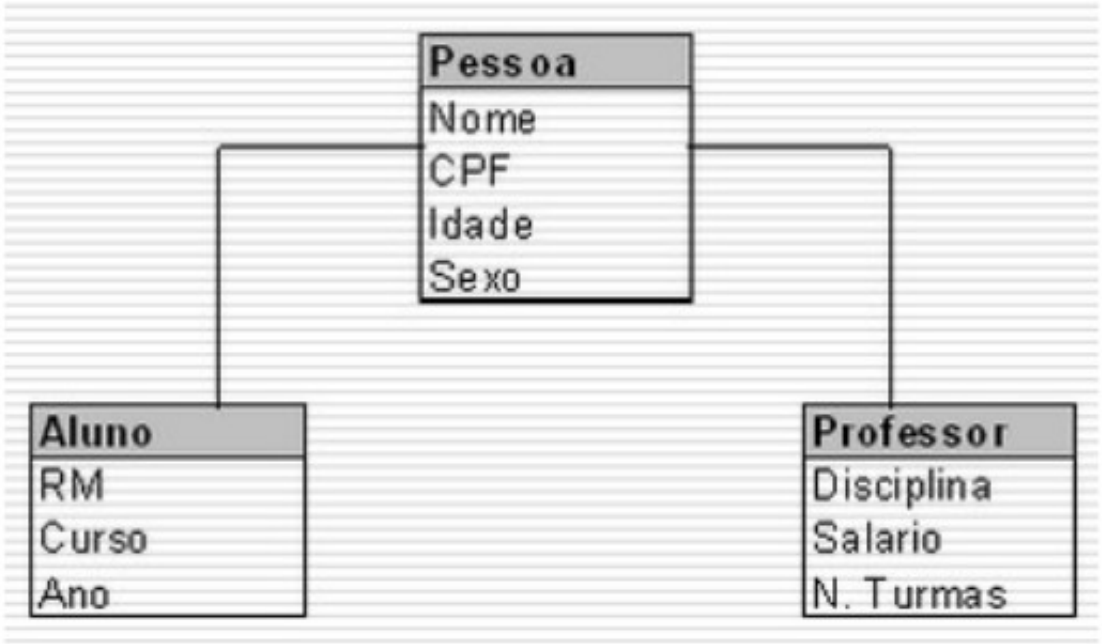
- É comum haver similaridades entre diferentes classes.
- Frequentemente, duas ou mais classes irão compartilhar os mesmos atributos e/ou métodos.
- Como nenhum de nós deseja reescrever várias vezes o mesmo código, seria interessante se algum mecanismo pudesse tirar proveito dessas similaridades.
- Por intermédio da herança, é possível modelar relacionamentos do tipo "é" ou "é semelhante", o que nos permite reutilizar rotinas e dados já existentes.

A classe filha herda os "bens" da classe mãe

- 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.

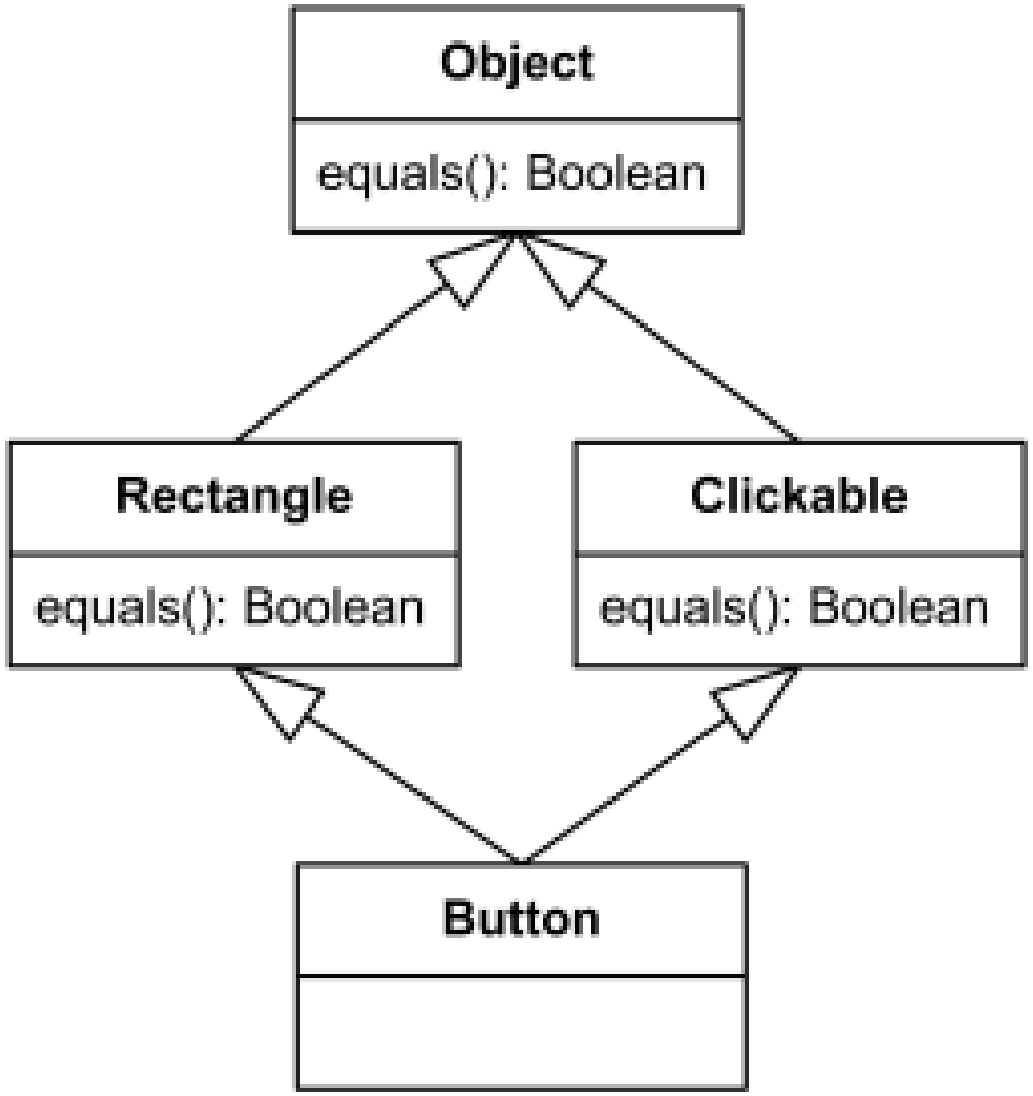
Exemplos de herança





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



Sintaxe do mecanismo de herança

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

class carro : public veiculo
{
 int nrodas;
 ...
 ...
 int mover( int nkilometros );

};
```

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

Sintaxe do mecanismo de herança

```
#include <iostream>
using namespace std;
class veiculo_rodoviario // Define uma classe base veículos.
   {
        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()
```

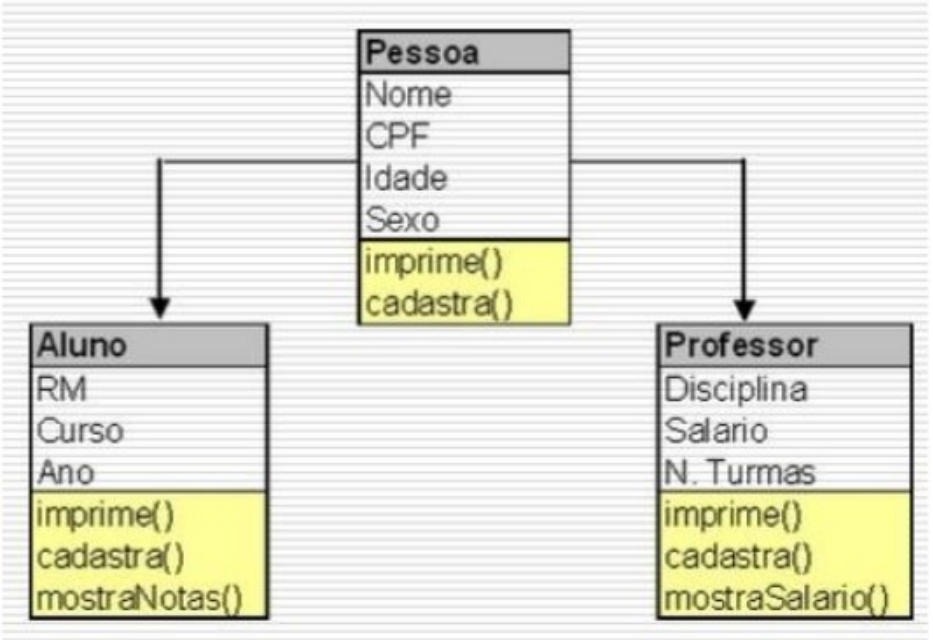
- Temos a classe mãe veiculo_rodoviario e duas classes filhas “:” caminhao e automovel.
- Características comuns estão na classe mãe.
- Características exclusivas estão nas classes filhas.
- Eventuais modificações na classe mãe são estendidas a todos os objetos criados.
- Repare que as duas classes filhas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

Polimorfismo

O que é polimorfismo?

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.



Programação Orientada a Objetos

Prof. Wesin Ribeiro