

Uma introdução à Orientação por Objetos

Generalização (herança)

Maratona de Programação 28 e 29/nov

Necessário ser realizada presencialmente, no exato horário do grupo matriculado:

G1: Quinta, 7h

G2: Quinta, 8h50

G3: Sexta, 7h

G4: Sexta, 8h50

Hoje: Bênção da Sede do *Campus* Lourdes

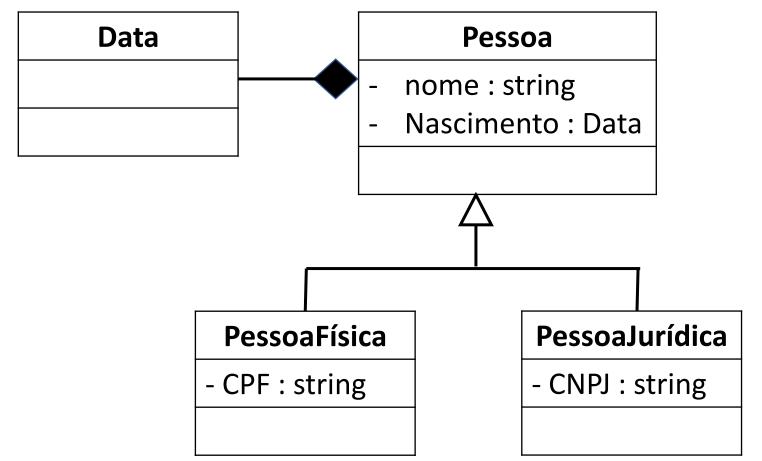
Sede do *Campus* Lourdes Rua Alvarenga Peixoto, 159

Grão Chanceler da PUC Minas, Dom Walmor Magnífico Reitor da PUC Minas, Pe. Luís Henrique Aula anterior: introdução à generalização (herança)

Generalização (herança)

Uma classe (base) pode generalizar as propriedades comuns de outras (derivadas)
Uma classe (base) pode se especializar em outras (derivadas)
Uma classe (derivada) pode herdar as propriedades descritas em outra (base)

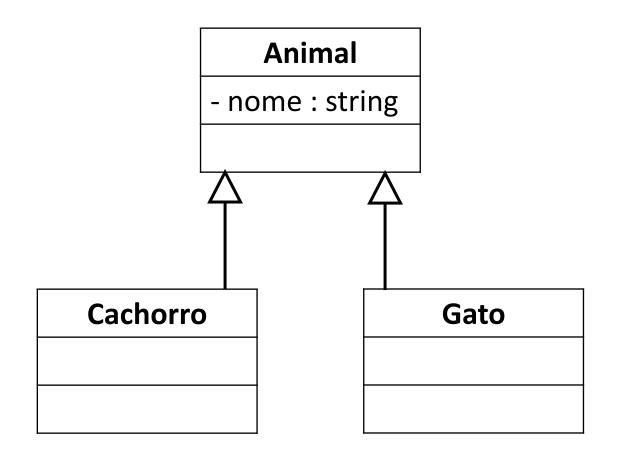
Relacionamento do tipo: is a

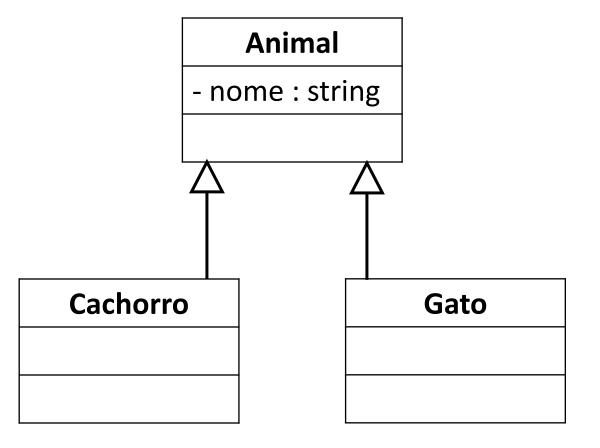


```
Pessoa*
          P = new Pessoa();
PessoaFisica* PF = new PessoaFisica();
PessoaJuridica* PJ = new PessoaJuridica();
     nome
     nome
                           nome
     CPF
                           CNPJ
```

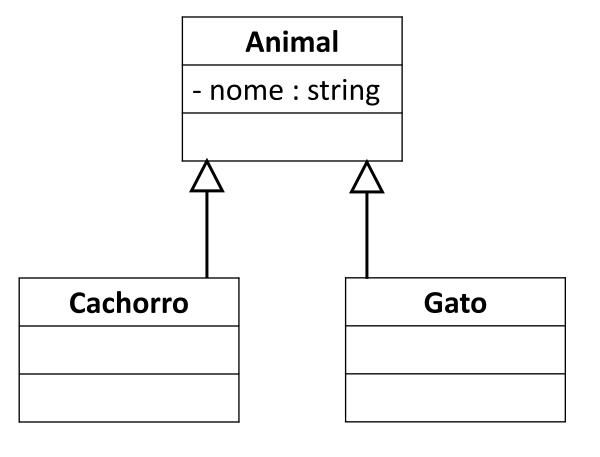
```
class Pessoa
  private : string nome;
  public: ...
};
class PessoaFisica: public Pessoa
  private : string CPF;
  public: ...
};
class PessoaJuridica: public Pessoa
  private : string CNPJ;
  public: ...
};
```

Implemente a generalização expressa nas três classes abaixo:





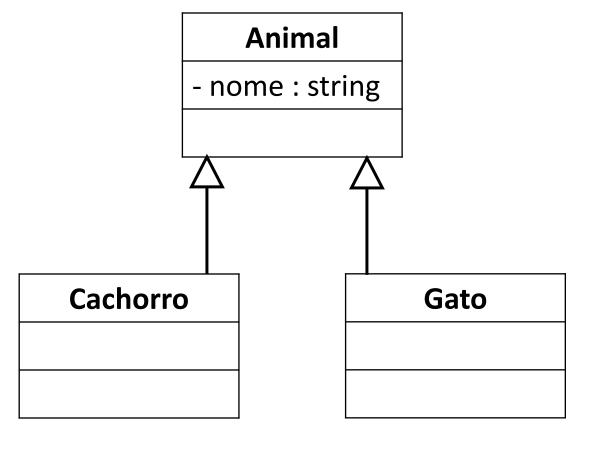
```
class Animal
{
    private :
        string nome;
    public :
};
```



```
class Animal
                private:
                    string nome;
                public:
              };
class Cachorro: Animal
```

public:

};

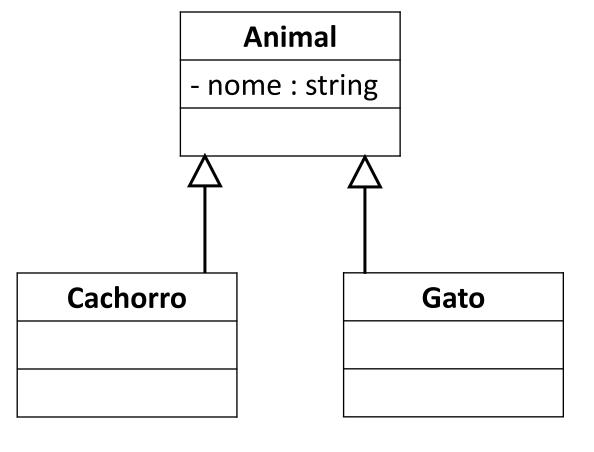


```
class Animal
                private:
                    string nome;
                public:
              };
                                 class Gato: Animal
class Cachorro: Animal
                                   public:
```

};

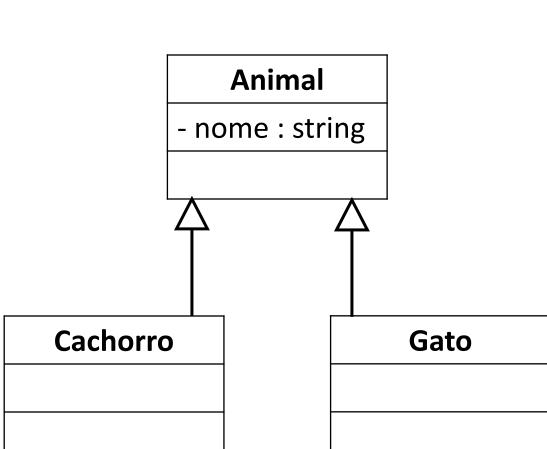
public:

};



```
class Animal
{
    private :
        string nome;
    public :
};
```

```
Animal animal;
...
Cachorro cachorro;
...
Gato gato;
...
```



```
class Animal
{
    private :
        string nome;
    public :
};
```

```
Animal animal;
...
Cachorro cachorro;
...
Gato gato;
...
```

```
class Cachorro : Animal {
   public :
```

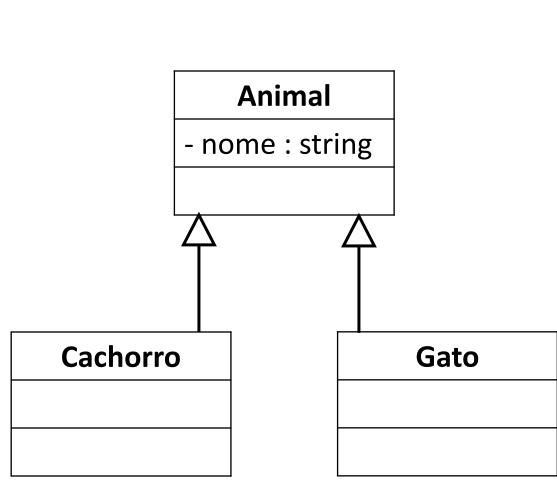
```
class Gato : Animal
{
   public :
};
```

nome ____

animal

nome ____

nome ____



```
class Animal
  private:
     string nome;
  public:
};
```

```
class Cachorro: Animal
  public:
```

nome

```
Animal animal;
Cachorro cachorro;
Gato gato;
cout << animal.nome;</pre>
```

```
class Gato: Animal
  public:
};
```

animal cachorro

nome

gato



```
Animal
         - nome : string
Cachorro
                          Gato
```

```
class Animal
{
    private :
        string nome;
    public :
};
```

class Cachorro: Animal

public:

};

```
Animal animal;
  Cachorro cachorro;
  Gato gato;
cout << animal.nome; //Privado
class Gato: Animal
  public:
};
```

```
animal
```





```
Animal
       - nome : string
      + getNome(): string
Cachorro
                          Gato
```

```
class Animal
{
    private :
        string nome;
    public :
};
```

```
Animal animal;
...
Cachorro cachorro;
...
Gato gato;
...
cout << animal.getNome();
```

nome

class Gato : Animal
{
 public :
};
 cachorro gato

nome

```
Animal
       - nome : string
      + getNome(): string
Cachorro
                          Gato
```

```
class Animal
  private:
     string nome;
  public:
     string getNome()
        return nome;
```

};

```
Animal animal;
Cachorro cachorro;
Gato gato;
cout << animal.getNome();</pre>
```

```
class Gato: Animal
class Cachorro: Animal
                               public:
  public:
                            };
```

nome

animal cachorro

nome

gato

```
Animal
       - nome : string
      + getNome(): string
Cachorro
                          Gato
```

```
class Animal
{
    private :
        string nome;
    public :
        string getNome()
        {
            return nome;
        }
};
```

```
...
Animal* animal;
...
Cachorro* cachorro;
...
Gato* gato;
...
cout << animal.getNome();</pre>
```

gato

```
Animal
       - nome : string
      + getNome(): string
Cachorro
                          Gato
```

```
class Animal
{
    private :
        string nome;
    public :
        string getNome()
        {
            return nome;
        }
};
```

```
Animal* animal;
...
Cachorro * cachorro;
...
Gato * gato;
...

cout << animal.getNome();
```

gato

```
Animal
       - nome : string
      + getNome(): string
Cachorro
                          Gato
```

```
class Animal
{
    private :
        string nome;
    public :
        string getNome()
        {
            return nome;
        }
};
```

```
Animal* animal;
...
Cachorro * cachorro;
...
Gato * gato;
...

cout << animal->getNome();
```

```
class Gato : Animal
{
   public :
};

cachorro gato
```

```
Animal
       - nome : string
      + getNome(): string
Cachorro
                          Gato
```

```
class Animal
{
    private :
        string nome;
    public :
        string getNome()
        {
            return nome;
        }
};
```

```
Animal* animal;
animal = new Animal;
...

Cachorro * cachorro;
cachorro = new Cachorro;
...

Gato * gato;
gato = new Gato;
...

cout << animal->getNome();
```

```
class Cachorro : Animal { public :
```

};

```
class Gato : Animal
{
   public :
};
```

```
nome ____
```

```
cachorro nome
```



Construtores em casos de herança

```
Animal
     - nome : string
     + getNome(): string
Cachorro
                          Gato
```

```
class Animal
  private:
     string nome;
  public:
     string getNome()
        return nome;
};
```

};

```
class Cachorro: Animal
                               public:
  public:
                             };
```

animal

nome Bichinho

```
Animal* animal;
animal = new Animal("Bichinho");
Cachorro * cachorro;
cachorro = new Cachorro;
Gato * gato;
gato = new Gato;
cout << animal->getNome();
```

```
class Gato: Animal
  cachorro
                  gato
```

nome

```
Animal
     - nome : string
     + getNome(): string
Cachorro
                          Gato
```

```
class Animal
{
    private :
        string nome;
    public :
        string getNome()
        {
            return nome;
        }
};
```

```
class Cachorro : Animal class Gato : {
   public : public :
};
```

nome Bichinho

animal

```
Animal* animal;
animal = new Animal("Bichinho");
Cachorro * cachorro;
cachorro = new Cachorro;
Gato * gato;
gato = new Gato;
cout << animal->getNome();
```

```
class Gato : Animal
{
   public :
};
cachorro gato
```

nome

```
Animal
     - nome : string
     + Animal(string)
     + getNome(): string
Cachorro
                          Gato
```

```
class Animal
  private:
     string nome;
  public:
     string getNome()
        return nome;
```

};

```
Animal* animal;
animal = new Animal("Bichinho");
Cachorro * cachorro;
cachorro = new Cachorro;
Gato * gato;
gato = new Gato;
cout << animal->getNome();
```

```
class Cachorro: Animal
                               public:
  public:
                             };
```

animal

nome Bichinho

```
class Gato: Animal
  cachorro
                 gato
    nome
                    nome
```

```
Animal
     - nome : string
     + Animal(string)
     + getNome(): string
Cachorro
                          Gato
```

```
class Animal
  private:
     string nome;
  public:
     Animal(string nome)
        this->nome= nome;
     string getNome()
        return nome;
   class Cachorro: Animal
     public:
```

animal

nome Bichinho

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro;
Gato * gato;
gato = new Gato;
cout << animal->getNome();
```

```
Animal
     - nome : string
     + Animal(string)
     + getNome(): string
Cachorro
```

```
class Animal
  private:
     string nome;
  public:
     Animal(string nome)
        this->nome= nome;
     string getNome()
        return nome;
    class Cachorro: Animal
      public:
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato("lua");
cout << animal->getNome();
```

```
Gato
{
    public:
    };
    animal
```

nome Bichinho

nome Lua

nome Sol

```
Animal
     - nome : string
     + Animal(string)
     + getNome(): string
Cachorro
```

```
class Animal
 private:
   string nome;
 public:
   Animal(string nome)
       this->nome= nome;
   string getNome()
      return nome;
};
     class Cachorro: Animal
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol"); //erro
Gato * gato;
gato = new Gato("lua"); //erro
cout << animal->getNome();
```

```
Gato class Cach { public :
```

```
class Gato : Animal
{
   public :
};
```

animal cachorro

nome Bichinho nome Sol

gato
nome Lua

Animal

- nome : string
- + Animal(string)
- + getNome(): string

Gato

Cachorro

+ Cachorro(string)

```
class Animal
 private:
   string nome;
 public:
   Animal(string nome)
      this->nome= nome;
   string getNome()
      return nome;
};
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato;
cout << animal->getNome();
```

class Cachorro : Animal
{
 public :

animal nome

class Gato : Animal
{
 public :
};

cachorro

nome

gato

Animal - nome : string

- + Animal(string)
- + getNome(): string

Gato

Cachorro

+ Cachorro(string)

```
class Animal {
 private:
   string nome;
 public:
   Animal(string nome) {
      this->nome= nome;
   string getNome() {
      return nome;
};
      class Cachorro: Animal
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato;
cout << animal->getNome();
```

```
class Cacnorro : Animal
{
    public :
        Cachorro(string nome)
        {
            this->nome= nome;
        }
};
```

nome

class Gato : Animal
{
 public :
}:

cachorro

gato nome

```
Animal
- nome : string
+ Animal(string)
+ getNome() : string
```

Gato

+ Cachorro(string)

Cachorro

```
class Animal {
 private:
   string nome;
 public:
   Animal(string nome) {
       this->nome= nome;
   string getNome() {
      return nome;
};
       public:
```

};

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato;
cout << animal->getNome();
```

cachorro

nome

gato

nome

animal

Escopo de visibilidade

Público:

Visível a todos que têm acesso ao objeto

Privado:

Visível apenas aos elementos da mesma classe

Protegido:

Visível apenas aos elementos da mesma classe e aos de suas especializações

Uma alternativa para o problema

Assim,

uma alternativa para o problema é o atributo nome, da classebase, ter o escopo de sua visibilidade definido como protected

Animal

nome: string

- + Animal(string nome);
- + getNome(): string

```
class Animal {
  protected :
    string nome;
  public :
    Animal(string nome) {
      this->nome= nome;
    }
  string getNome() {
      return nome;
    }
};
```

```
Animal
   # nome : string
    + Animal(string)
    + getNome(): string
Cachorro
                         Gato
```

+ Cachorro(string)

```
class Animal {
  protected :
    string nome;
  public :
    Animal(string nome) {
       this->nome= nome;
    }
  string getNome() {
      return nome;
    }
};
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato;
cout << animal->getNome();
```

```
class Cachorro : Animal
{
  public :
    Cachorro(string nome)
    {
     this->nome= nome;
    }
};
animal
nome Bichinho
```

```
class Gato : Animal
{
   public :
};
```

gato

nome Lua

cachorro

nome 5d

```
Generalização
              Animal
       # nome : string
       + Animal(string)
       + getNome() : string
    Cachorro
                          Gato
+ Cachorro(string)
```

```
class Animal {
                             Animal* animal;
 protected:
                             animal = new Animal("bichinho");
   string nome;
 public:
                             Cachorro * cachorro;
   Animal(string nome) {
                             cachorro = new Cachorro("sol");
      this->nome= nome;
                             Gato * gato;
   string getNome() {
                             gato = new Gato;
      return nome;
                             cout << animal->getNome();
};
                             cout << cachorro->getNome();
     class Cachorro: Animal
                                          class Gato: Animal
       public:
                                            public:
         Cachorro(string nome)
          this->nome= nome;
                   animal
                                    cachorro
                                                    gato
     };
```

nome

nome

Será autorizada a execução do método público *getNome()* sobre a instância da classe Cachorro?

```
Animal* animal;
                                          protected:
                                                                      animal = new Animal("bichinho");
                                            string nome;
                                          public:
                                                                      Cachorro * cachorro;
                                            Animal(string nome) {
                                                                      cachorro = new Cachorro("sol");
                                               this->nome= nome;
                  Animal
                                                                      Gato * gato;
         # nome : string
                                            string getNome() {
                                                                      gato = new Gato;
                                               return nome;
         + Animal(string)
         + getNome() : string
                                                                      cout << animal->getNome();
                                         };
                                                                   cout << cachorro->getNome(); //erro
                                               class Cachorro: Animal
                                                                                  class Gato: Animal
                                                public:
                                                                                    public:
     Cachorro
                                 Gato
                                                  Cachorro(string nome)
                                                   this->nome= nome;
+ Cachorro(string)
                                                           animal
                                                                            cachorro
                                               };
                                                              nome
                                                                               nome
```

gato

nome

class Animal {

```
protected:
                                            string nome;
                                          public:
                                            Animal(string nome) {
                                               this->nome= nome;
                  Animal
                                                                     Gato * gato;
         # nome : string
                                            string getNome() {
                                                                     gato = new Gato;
                                               return nome;
         + Animal(string)
         + getNome(): string
                                        };
                                              class Cachorro: Animal
                                                public:
     Cachorro
                                 Gato
                                                 Cachorro(string nome)
                                                   this->nome= nome;
+ Cachorro(string)
                                                           animal
                                                                            cachorro
                                              };
                                                              nome
```

class Animal {

```
Animal* animal;
  animal = new Animal("bichinho");
  Cachorro * cachorro;
  cachorro = new Cachorro("sol");
  cout << animal->getNome();
💢 cout << cachorro->getNome(); //Herança
               class Gato: Animal
                 public:
                         gato
            nome
                           nome
```

```
protected:
                                             string nome;
                                          public:
                                                                      Cachorro * cachorro;
                                            Animal(string nome) {
                                                this->nome= nome;
                  Animal
                                                                      Gato * gato;
         # nome : string
                                             string getNome() {
                                                                      gato = new Gato;
                                               return nome;
          + Animal(string)
         + getNome() : string
                                         };
                                               class Cachorro : Animal // herança privada
                                                public:
     Cachorro
                                  Gato
                                                  Cachorro(string nome)
                                                   this->nome= nome;
+ Cachorro(string)
                                                           animal
                                                                            cachorro
                                               };
                                                              nome
```

class Animal {

```
Animal* animal;
  animal = new Animal("bichinho");
  cachorro = new Cachorro("sol");
  cout << animal->getNome();
cout << cachorro->getNome(); //erro
                      class Gato: Animal
                        public:
                        gato
           nome
                           nome
```

```
Animal
        # nome : string
        + Animal(string)
        + getNome() : string
    Cachorro
                              Gato
+ Cachorro(string)
```

```
class Animal {
 protected:
   string nome;
 public:
   Animal(string nome) {
      this->nome= nome;
   string getNome() {
      return nome;
};
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato;
cout << animal->getNome();
cout << cachorro->getNome();
```

nome

nome

```
class Cachorro : public Animal
                                     class Gato: public Animal
 public:
                                       public:
   Cachorro(string nome)
    this->nome= nome;
             animal
                               cachorro
                                                gato
};
```

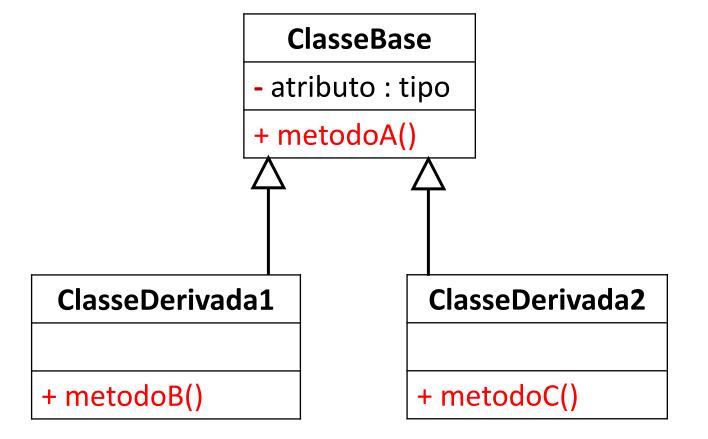
nome

Escopo de visibilidade

E qual alguma outra alternativa para o caso de manter o atributo com escopo de visibilidade igual a privado?

Escopo de visibilidade

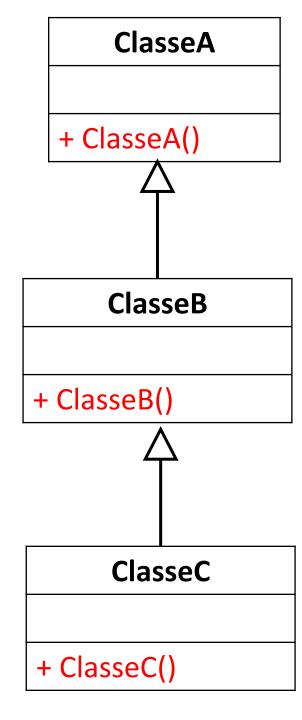
Para o caso de manter o elemento privado, há alguma outra alternativa para acessá-lo?



Construtores em hierarquia

E qual alguma outra alternativa para o caso de manter o atributo com escopo de visibilidade igual a privado?

Um construtor pode, explicitamente, solicitar a execução do construtor de sua classe base (superclasse).



```
Animal
        - nome : string
        + Animal(string)
        + getNome(): string
    Cachorro
                          Gato
+ Cachorro(string)
```

```
class Animal {
 private:
   string nome;
 public:
   Animal(string nome) {
       this->nome= nome;
   string getNome() {
      return nome;
};
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato;
cout << animal->getNome();
cout << cachorro->getNome();
```

```
class Cachorro : public Animal
{
   public :
        Cachorro(string nome) : Animal(nome)
        { }
};
```

cachorro

nome

gato

nome

animal

nome

Animal - nome : string + Animal(string) + getNome(): string Cachorro Gato + Cachorro(string)

```
class Animal {
 private:
   string nome;
 public:
   Animal(string nome) {
      this->nome= nome;
   string getNome() {
      return nome;
};
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato("lua");
cout << animal->getNome();
cout << cachorro->getNome();
cout << gato->getNome();
```

```
class Cachorro : public Animal
{
    public :
        Cachorro(string nome) : Animal(nome)
        {
        }
};
```

cachorro

nome

gato

nome

animal

nome

Animal - nome : string + Animal(string) + getNome(): string **Cachorro** Gato + Cachorro(string) + Gato(string)

```
class Cachorro: public Animal {
 public:
   Cachorro(string nome): Animal(nome)
};
```

```
class Animal {
 private:
   string nome;
 public:
   Animal(string nome) {
       this->nome= nome;
   string getNome() {
      return nome;
};
       class Gato: public Animal
         public:
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato("lua");
cout << animal->getNome();
cout << cachorro->getNome();
cout << gato->getNome();
```

```
};
```

animal nome cachorro nome gato nome

Animal - nome : string + Animal(string) + getNome(): string **Cachorro** Gato + Cachorro(string) + Gato(string)

```
class Cachorro : public Animal {
   public :
        Cachorro(string nome) : Animal(nome)
        {
      }
};
```

```
class Animal {
 private:
   string nome;
 public:
   Animal(string nome) {
      this->nome= nome;
   string getNome() {
      return nome;
};
```

```
Animal* animal;
animal = new Animal("bichinho");
Cachorro * cachorro;
cachorro = new Cachorro("sol");
Gato * gato;
gato = new Gato("lua");
cout << animal->getNome();
cout << cachorro->getNome();
cout << gato->getNome();
```

cachorro

nome

gato

nome

```
class Gato : public Animal {
   public :
      Gato(string nome) : Animal(nome)
      { }
};
```

nome

animal

Trabalho Prático Final Implementar Generalização

Menu de Opções

- 0 Sair do programa
- 1 Cadastrar uma pessoa
- 2 Listar todas as pessoas
- 3 Pesquisar por nome
- 4 Pesquisar por CPF
- 5 Excluir pessoa
- 6 Apagar todas as pessoas cadastradas
- 7 Aniversariantes do mês

Submenu a cada opção do menu principal

- 1 Cadastrar uma pessoa
 - 1.1 Cadastrar Professor
 - 1.2 Cadastrar Aluno
- 2 Listar todas as pessoas
 - 2.1 Listar Professores
 - 2.2 Listar Alunos
- 3 Pesquisar por nome
 - 3.1 Pesquisar Professores por nome
 - 3.2 Pesquisar Alunos por nome
- 4 Pesquisar por CPF
 - 4.1 Pesquisar Professores por CPF
 - 4.2 Pesquisar Alunos por CPF
- 5 Excluir pessoa
 - 5.1 Excluir Professor (pelo CPF)
 - 5.2 Excluir Aluno (pelo CPF)
- 6 Apagar todas as pessoas cadastradas
 - 6.1 Excluir todos os Professores
 - 6.2 Excluir todos os Alunos
- 7 Aniversariantes do mês
 - 7.1 Informar o mês a ser pesquisado
 - 7.1 Listar os Professores aniversariantes do mês
 - 7.2 Listar os Alunos aniversariantes do mês