Algoritmos e Estruturas de Dados I
Aula04

Herança

Prof. MSc. Adalto Selau Sparremberger


assparremberger@senacrs.com.br

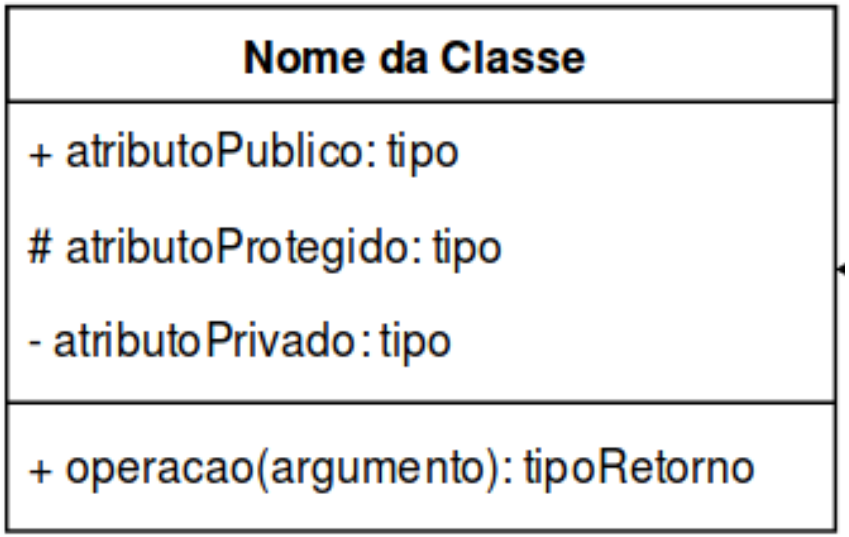

@adaltoss


/assparremberger

Fecomércio RS | Senac

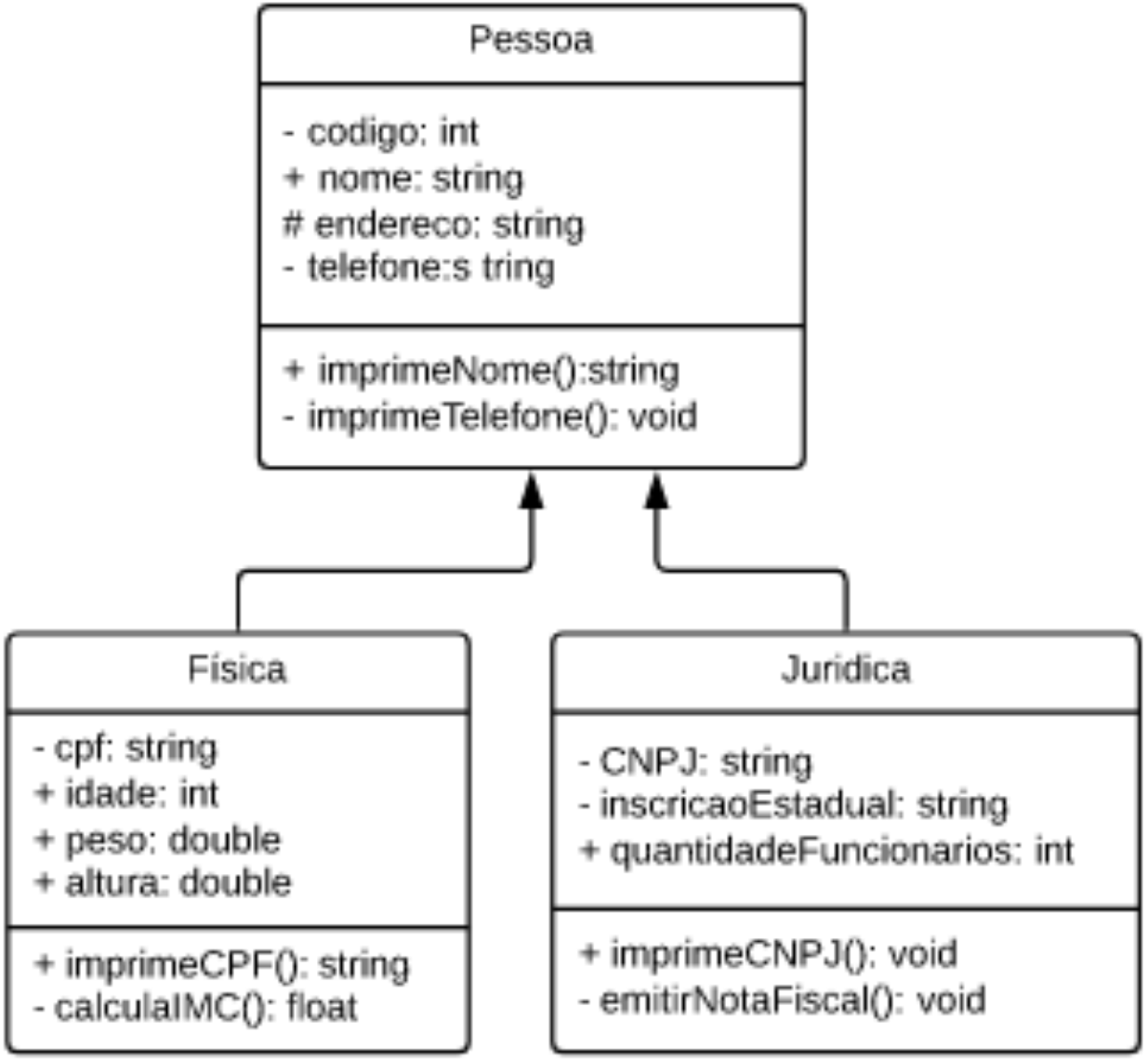
Classe (recapitulando...)

- Uma classe é um molde para objetos.
- Um objeto é uma instância de uma classe.
- Uma classe possui **atributos** (características) e **métodos** (funções/ações)



Herança

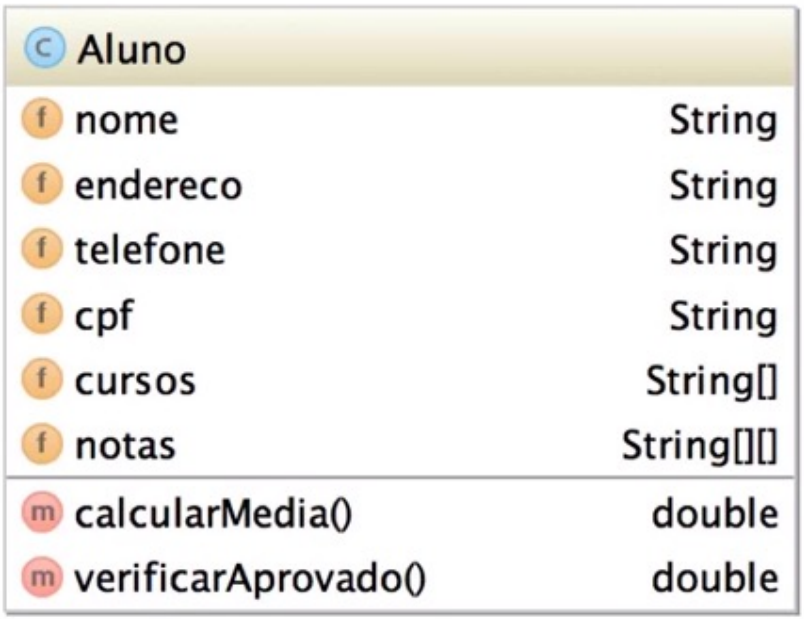
- Herança nos permite definir uma classe que herda todos os atributos e métodos de outra classe.
- **Classe pai** (Super Class) é a classe que está sendo herdada também chamada de classe base ou classe genérica.
- **Classe filha** (Sub Class) é a classe que herda de outra classe, também chamada de classe derivada ou classe especializada.

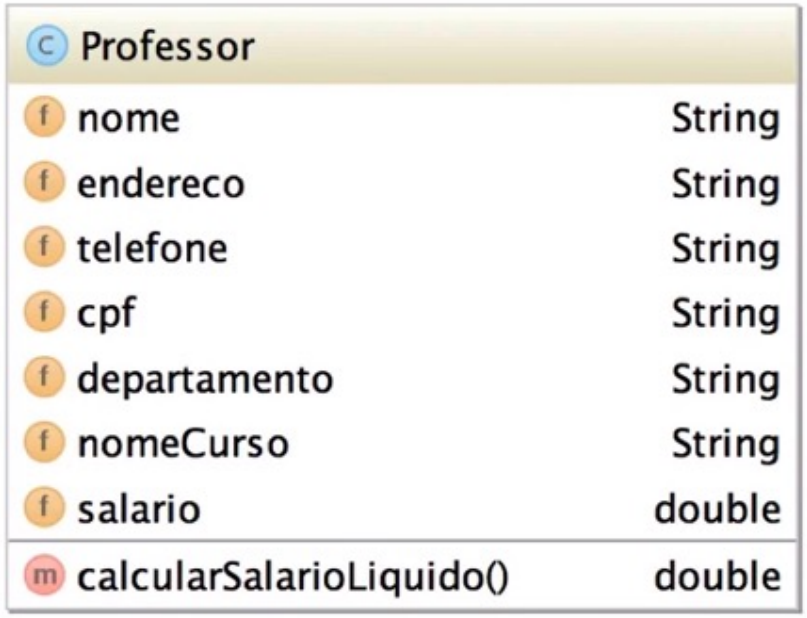

Pessoa
- codigo: int
+ nome: string
# endereco: string
- telefone:s tring
+ imprimeNome():string
- imprimeTelefone(): void
Física
- cpf: string
+ idade: int
+ peso: double
+ altura: double
+ imprimeCPF(): string
- calculaIMC(): float
Juridica
- CNPJ: string
- inscricaoEstadual: string
+ quantidadeFuncionarios: int
+ imprimeCNPJ(): void
- emitirNotaFiscal(): void

Herança

- A herança é um princípio da POO que permite a criação de novas classes a partir de outras previamente criadas.
- Essas novas classes são chamadas de subclasses, ou classes derivadas; e as classes já existentes, que deram origem às subclasses, são chamadas de superclasses, ou classes base.
- Uma subclasse herda métodos e atributos de sua superclasse; apesar disso, pode escrevê-los novamente para uma forma mais específica de representar o comportamento do método herdado.

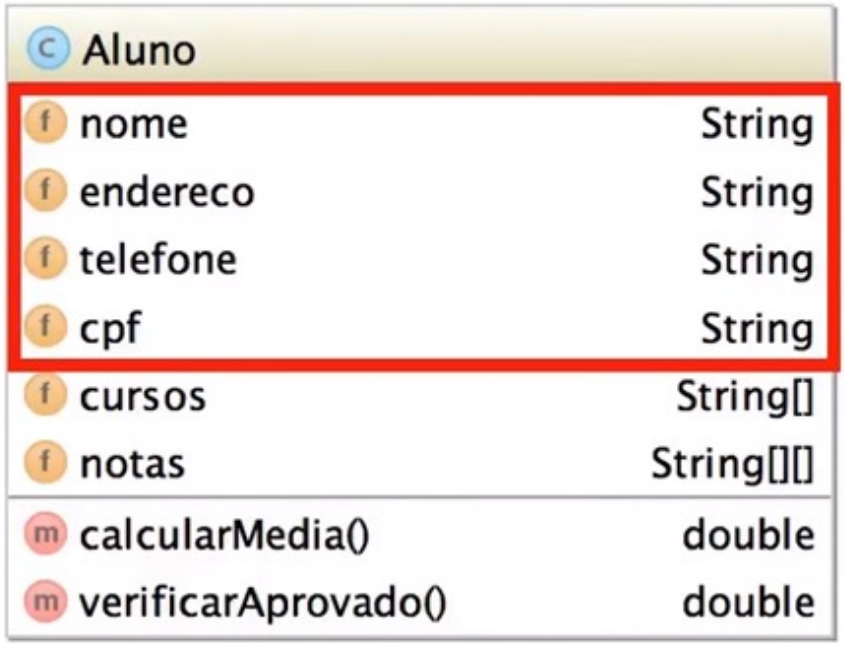
Herança

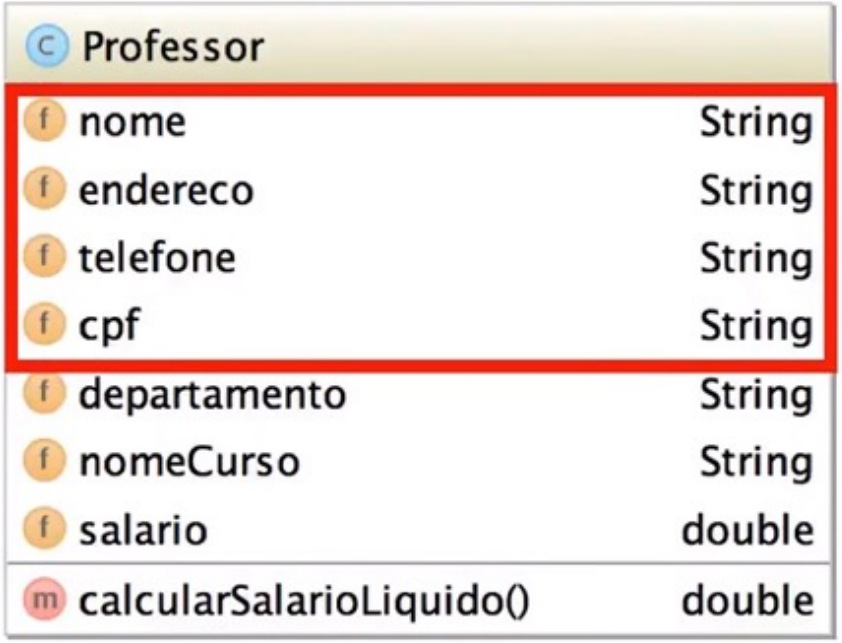

Aluno
nome String
endereco String
telefone String
cpf String
cursos String[]
notas String[][]
calcularMedia() double
verificarAprovado() double


Professor
nome String
endereco String
telefone String
cpf String
departamento String
nomeCurso String
salario double
calcularSalarioLiquido() double

Herança

- Analisando professores e alunos, vemos que todos podem (devem) ter nome, endereço, telefone e cpf; portanto, nada mais justo que criar subclasses de Pessoa para representa-los.

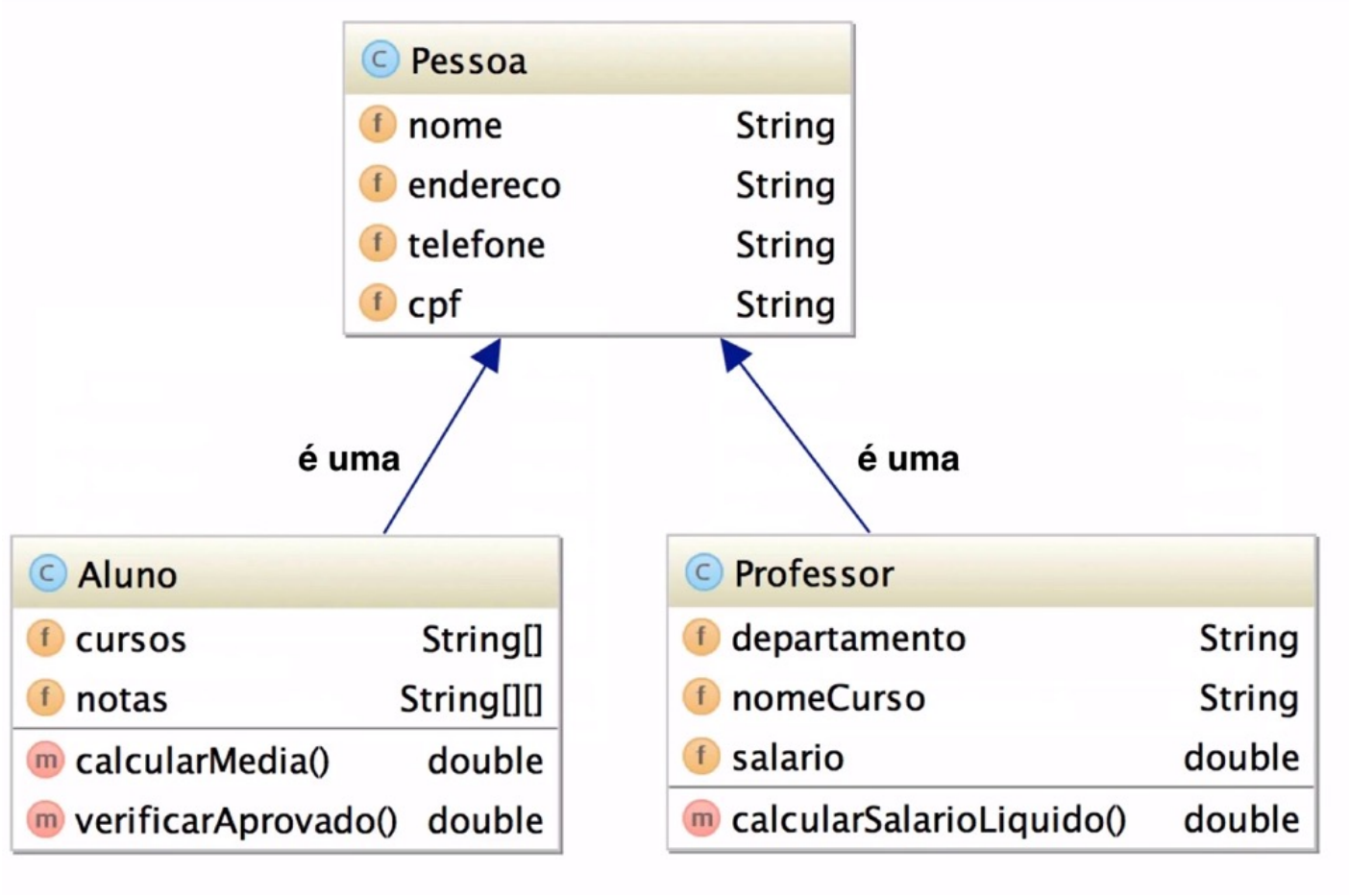




Herança

- Porém, existem algumas características que os alunos tem que um professor não tem: nota, por exemplo.
- Também existem características que os professores tem que os alunos não tem: departamento, salário, etc.
- Para resolver esse problema, e deixar a aplicação MUITO MAIS ORGANIZADA, vamos fazer duas classes: “Aluno" e “Professor".
- Cada uma dessas irá herdar a classe “Pessoa", pois também são um tipo de pessoa.
- Dizemos que “Pessoa" é a superclasse, “Aluno" e “Professor" são subclasses.

Herança



- Para saber quando usar, e detectar o uso de herança, use a relação 'é um'.
  No exemplo acima:
  “Aluno" é uma “Pessoa", e “Professor" é uma “Pessoa".

Herança em Python

Pessoa.py

```
class Pessoa:
  def __init__(self, codigo, nome, end, fone):
    self.codigo = codigo
    self.nome = nome
    self.endereco = end
    self.fone = fone

  def imprimir(self):
    print( "Nome: " , self.nome )
    print( "End: " , self.endereco)
```

Fisica.py

```
class Fisica(Pessoa):
    def __init__(self, codigo, nome, end, fone, cpf):
        Pessoa.__init__(self, codigo, nome, end, fone)
        self.cpf = cpf
```

Herança em Python

Pessoa.py

```
class Pessoa:
  def __init__(self, codigo, nome, end, fone):
    self.codigo = codigo
    self.nome = nome
    self.endereco = end
    self.fone = fone

  def imprimir(self):
    print( "Nome: " , self.nome )
    print( "End: " , self.endereco)
```

Fisica.py

```
class Fisica(Pessoa):
    def __init__(self, codigo, nome, end, fone, cpf):
        Pessoa.__init__(self, codigo, nome, end, fone)
        self.cpf = cpf
```

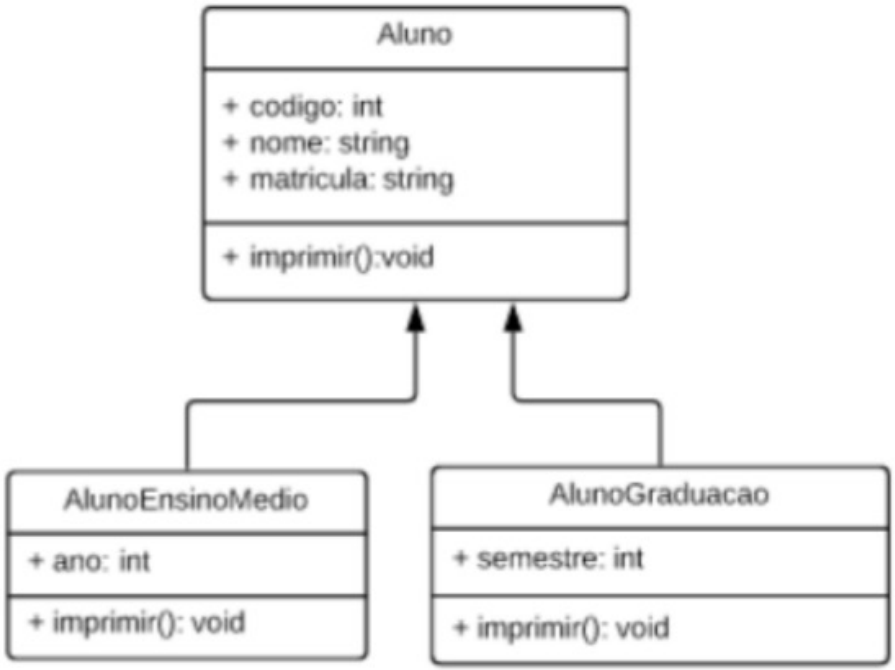
main.py

```
x = Fisica( None, "João", "Rua Um", "(51) 1234", "000.111.222-33" )
x.imprimir()
```

```
Nome:  João
End:  Rua Um
```

Exercício

- Construa um algoritmo para implementar a classe Aluno (código, nome, matrícula). A classe Aluno possui duas subclasses, a classe AluEnsinoMedio(ano) e AlunoGraduacao(semestre). As 3 classes possuem o método construtor o também o método imprimir(), que imprime na tela os valores de todos os atributos da sua classe



Referências

- BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. Rio de Janeiro: Elsevier, 2007.
- W3Schools: https://www.w3schools.com/python/default.asp