Herança Simples

Programação Orientada a Objetos

Python

Prof. Diógenes Furlan

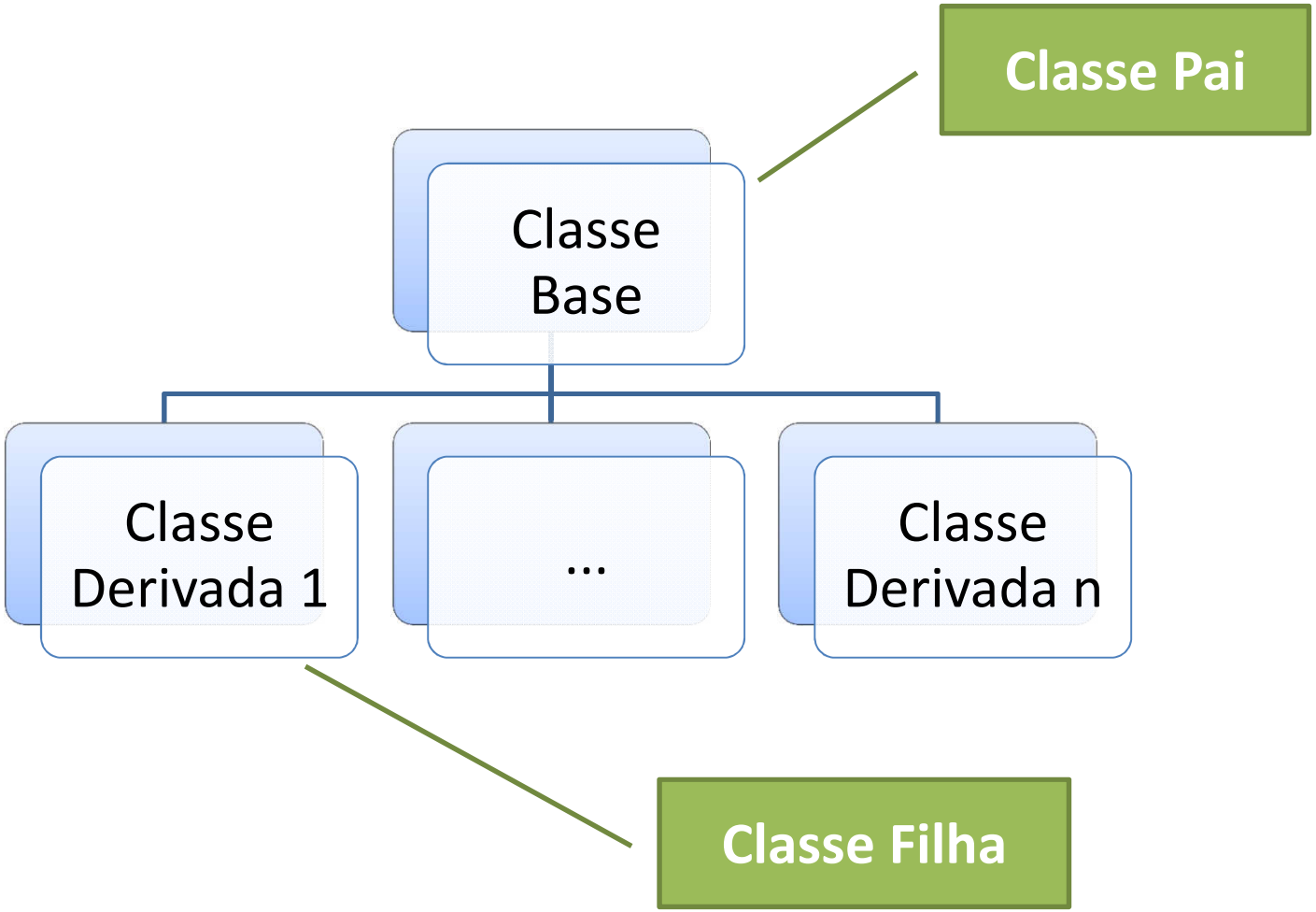
Sumário

- Atributos protegidos
- Construtor na herança
- Sobrescrita de métodos

Herança

- Processo de criação
  - de novas classes (classe derivada)
  - baseado em classes existentes (classe base)
- A classe derivada herda todas as características da classe base, além de incluir características próprias adicionais
- Reutilização de Código

Herança


Classe Pai
Classe Base
Classe Derivada 1
...
Classe Derivada n
Classe Filha

Ser X Ter

- Um Cliente é uma pessoa ou tem uma pessoa?
- Um Cliente é uma senha ou tem uma senha?
- Uma Conta Corrente é uma conta ou tem uma conta?
- Uma Pessoa é uma data de aniversário, ou tem uma data de aniversário?
- Uma Conta é um saldo ou tem um saldo?
- Uma Reta é um ponto ou tem um ponto?

Herança X Composição

- Herança (SER)
  - Um Cliente é uma Pessoa.
  - Um Funcionário é uma Pessoa.
  - Uma Conta Corrente é uma Conta.
- Composição (TER)
  - Uma Pessoa tem uma Data de Aniversário.
  - Um Cliente tem uma senha.
  - Uma Conta tem um saldo.
  - Uma Reta tem vários pontos.

Exemplo

- Criando as classes

PESSOA
- nome: string - dtNasc: data
+ Pessoa(nome) + getNome() : string + mostra()

FUNCIONÁRIO
- nome: string - dtNasc: data - salario: float
+ Funcionário(nome, salario) + getNome() : string + mostra() + getSalario() : float + getSalarioAnual() : float

Exemplo

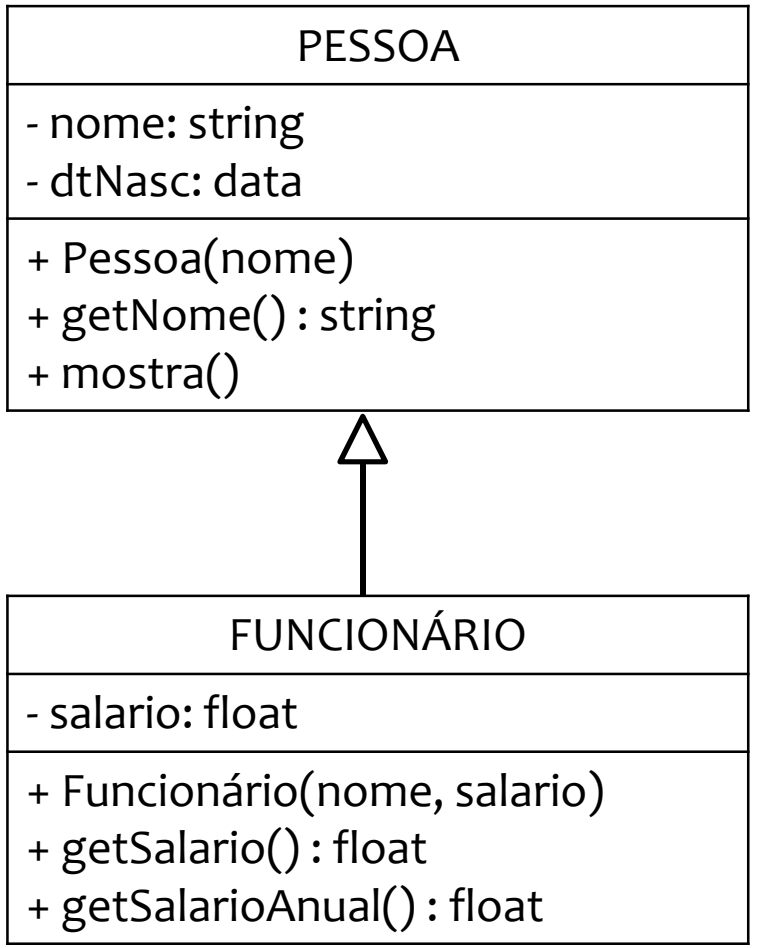
- Identificando as semelhanças

PESSOA
- nome: string - dtNasc: data
+ Pessoa(nome) + getNome() : string + mostra()

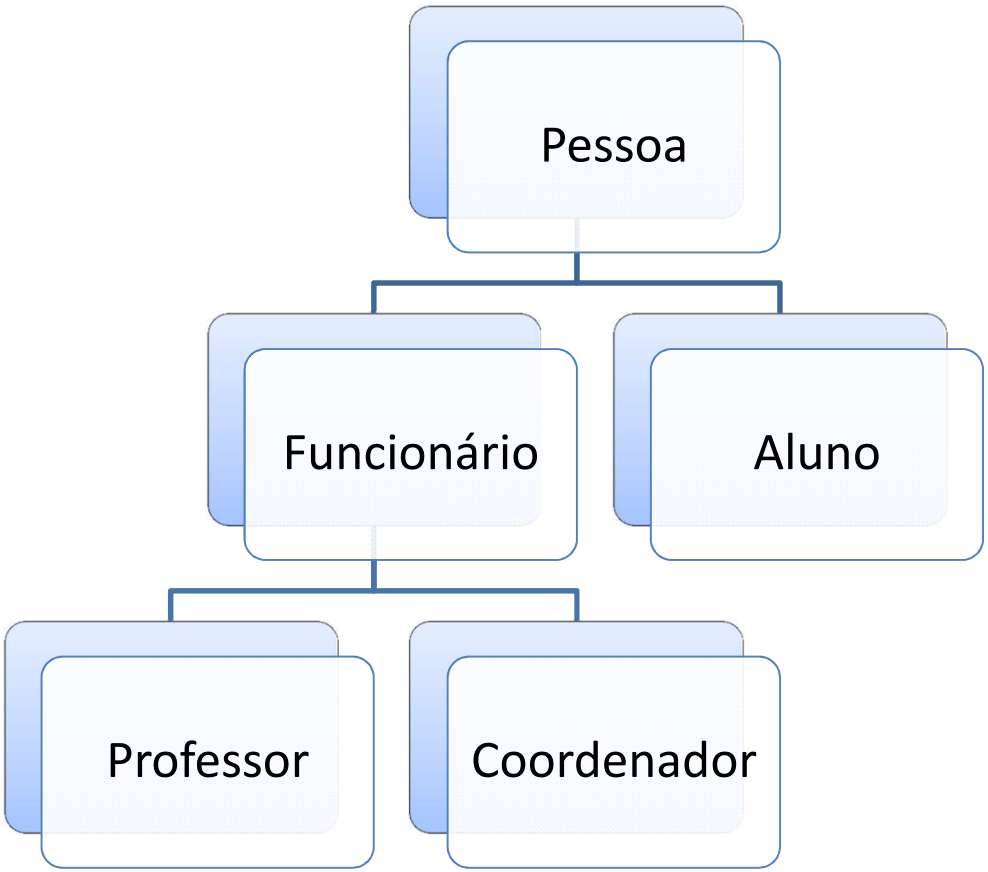
FUNCIONÁRIO
- nome: string - dtNasc: data - salario: float
+ Funcionário(nome, salario) + getNome() : string + mostra() + getSalario() : float + getSalarioAnual() : float

Exemplo

- Usando a herança



Hierarquia de Herança



Classe Pessoa

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def getNome(self):
        return self.nome

    def __str__(self):
        return 'Nome=' + self.nome + '\nIdade = ' + str(self.idade)
```

Classe Funcionário

Herança

```
class Funcionário(Pessoa):
    def __init__(self, nome, idade, salario):
        super().__init__(nome,idade)
        self.salario = salario

    def getSalario(self):
        return self.salario

    def getSalarioAnual(self):
        return self.salario * 12

# main
p1 = Pessoa('João',34)
p2 = Funcionário('Adalberto', 26, 1345.0)

print(p1)
print(p2)
```

super(): acesso à classe pai

Classe Base vs. Classe Derivada

Membros	Própria Classe	Classes Derivadas	Outros
Privados	X		
Protegidos	X	X	
Públicos	X	X	X

Membros **protegidos** são semelhantes a membros **privados**, exceto pelo fato de serem visíveis a todos os membros de uma classe derivada.

SOBRESCRITA

Sobrescrita (Override)

- As subclasses podem ter métodos com o mesmo **nome** que a classe pai (sobrecarga).
- As subclasses podem ter métodos com a mesma **assinatura** que a classe pai (sobrescrita).

- Finalidades
  - Reescrita
  - Extensão

Exercício

- Defina um método **__str__** para a classe Funcionário.

- Defina um método **bonificação** para as classes:
  - Funcionário = 10%
  - Professor = 15%

Classe Funcionário

```
class Funcionário(Pessoa):
    def __init__(self, nome, idade, salario):
        super().__init__(nome,idade)
        self.salario = salario

    def getSalario(self):
        return self.salario

    def getSalarioAnual(self):
        return self.salario * 12

    def __str__(self):
        return super().__str__() + '\nSalário = ' + str(self.salario)

    def getBonificação(self):
        return self.salario * 0.10
```

super(): acesso à classe pai

Classe Professor

```
class Professor(Funcionário):
    def getBonificação(self):
        return self.salario * 0.15

# main
f1 = Funcionário('Josias', 26, 3000)
p1 = Professor('Malaquias', 56, 5000)

print( f1.getBonificação() ) # 300
print( p1.getBonificação() ) # 500 ou 750
```

EXERCÍCIOS

Exercício 1

- Modele as seguintes classes com pelo menos um atributo e um método cada:
  - classe Aluno, derivada de Pessoa
  - classe Professor, derivada de Funcionário.
  - classe Coordenador, derivada de Funcionário.

- Teste todas as classes no main().

Modele a seguinte Hierarquia de Classes