ECT2540 - Programação Orientada a Objetos

Herança

Prof. Bruno Marques F. da Silva

Escola de Ciências e Tecnologia
Universidade Federal do Rio Grande do Norte

29 de Abril de 2019

POO - Herança

Objetivo da aula:

- Apresentar o mecanismo de herança
  - O que é herança em POO
  - Como utilizá-la na linguagem Python

Os Quatro Pilares de POO

- Abstração
- Encapsulamento
- **Herança**: capacidade de uma classe herdar o comportamento definido por outra classe
- Polimorfismo

Herança I

- Mecanismo existente em linguagens de programação orientada a objetos que permite a reutilização de comportamento entre classes diferentes
- Com herança:
  - A reutilização de comportamento se dá por meio de reutilização de código
  - Um novo tipo de relação entre objetos é estabelecido
  - Esta relação acontece entre objetos genéricos e objetos específicos modelados a partir de objetos do mundo real

Herança II

Herança

- Relação do tipo "*é um*": *B é A* ou "um objeto B é um objeto A"
- Classe B herda o comportamento (atributos e métodos) da classe A
- A: classe base, classe mãe ou superclasse
- B: classe derivada, classe filha ou subclasse
- As superclasses devem oferecer comportamentos genéricos
- As subclasses devem oferecer comportamentos específicos

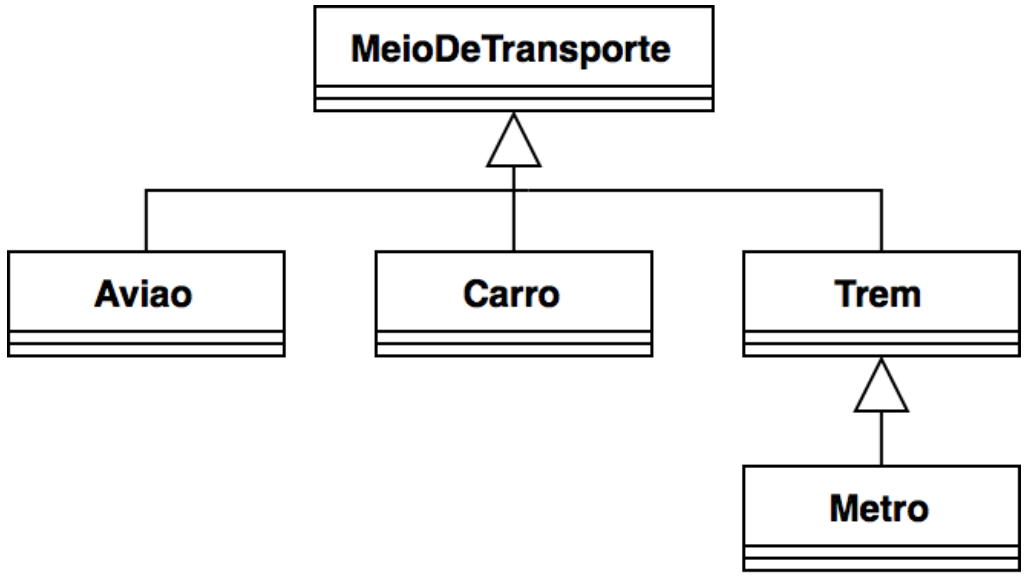
Herança III

Exemplos de relações de herança:

- `Pessoa` e `Aluno`: todo aluno (objeto específico) é uma pessoa (objeto genérico)
- `MeioDeTransporte` e `Carro`: todo carro (objeto específico) é um meio de transporte (objeto genérico)
- `Sensor` e `Sonar`: todo sonar (objeto específico) é um sensor (objeto genérico)
- `Poligono` e `Triangulo`: todo triângulo (objeto específico) é um polígono (objeto genérico)

Herança IV

Exemplo de hierarquia:



Herança V

A relação “é um” entre diferentes classes permite:

- Definir um comportamento em comum para objetos de uma mesma hierarquia
  - O código da classe base é reutilizado em todas as subclasses
  - Qualquer alteração no código da classe base é propagado para todas as subclasses
- Definir um comportamento específico para objetos especializados presentes em uma hierarquia. Isto deve ser implementado de duas maneiras:
  - Reescrevendo totalmente o comportamento da classe base
  - Estendendo o comportamento da classe base

Herança em Python I

Em Python, a indicação de que uma classe deve herdar o comportamento de outra é feito da seguinte forma:

```
# Classe A: classe base
class A:
...

# Classe B: classe derivada
# Classes base sao indicadas
# em uma tupla na definicao
# das classes derivadas
class B(A):
...
```

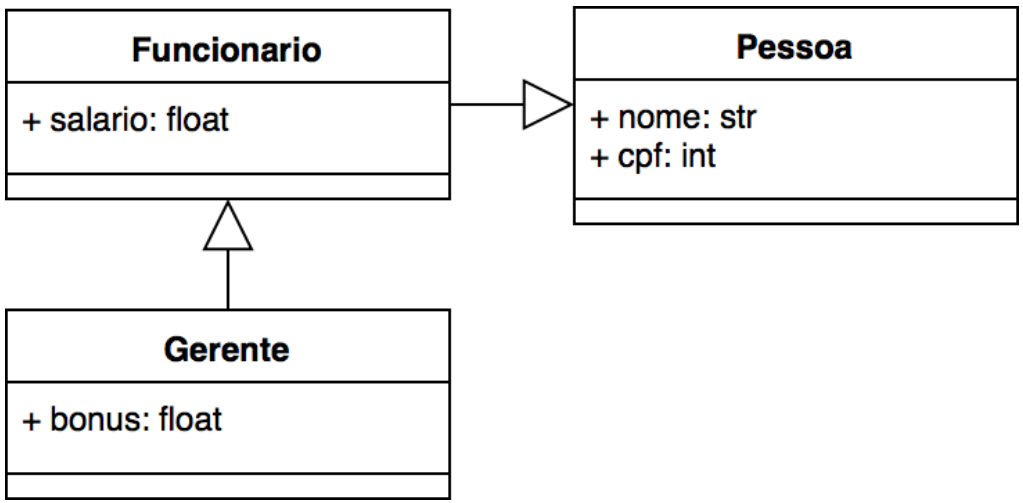
Herança em Python II

- Desta forma, todos os atributos e métodos da classe `A` estão na classe `B`:
  - Atributos de instância e de classe
  - Métodos de instância e de classe
- E se quisermos estender as funcionalidades de `A` na classe `B`?

Herança em Python III

Exemplo (ver código `heranca.py`):

- `Pessoa` tem nome e CPF
- `Funcionario` tem salário
- `Gerente` tem bônus aplicado ao salário



Herança em Python IV

Em resumo:

- Uma classe derivada contém sempre todos os atributos e métodos da classe base
  - Em Python, não há o modificador `protected`, que em outras linguagens impede que um atributo/método seja herdado
- Quando uma classe derivada estende a classe base com novos atributos, é necessário sobrescrever o inicializador `__init__` para declará-los e inicializá-los
- Ao invés de copiar e colar o código de um método da classe base ao estendê-la, este deve ser explicitamente chamado como `ClasseBase.metodo(self, ...)`
- Quando uma classe derivada sobrescreve um método, esta nova versão é chamada se invocada através de um objeto da classe derivada, mas não através de um objeto da classe base

Herança em Python V

- Python possui a função especial `isinstance`:
  - Sintaxe: `isinstance(obj, classe)`: retorna verdadeiro se `obj` for da classe `classe` ou falso caso contrário
- Faz a mesma coisa que a função `type`, com uma diferença: `isinstance` considera a hierarquia de classes. Ou seja:
  - Um gerente é uma pessoa. Para `obj` da classe `Gerente`, `isinstance(obj, Pessoa)` retorna verdadeiro
  - Nem toda pessoa é um gerente. Para `obj` da classe `Pessoa`, `isinstance(obj, Gerente)` retorna falso

Herança em Python VI

- Em Python, toda classe herda da classe `object` implicitamente
  - `isinstance(o, object)` é verdadeiro, seja quem for o objeto `o`

Exercício

Implemente um sistema para bancos com os requisitos a seguir:

- Existem 2 tipos de contas: conta corrente e conta poupança
- Toda conta deve conter os métodos `saque` e `deposito`
- Uma conta poupança não pode ficar com saldo negativo
- Uma conta poupança tem o método `rende`, que aplica a taxa de 0.95% sobre o saldo da poupança
- Todo saque em uma conta poupança tem uma taxa de R$2

Sumário

- Após esta aula, você deve saber:
  - O que é herança
  - Como implementar herança em Python
    - Para aproveitar o comportamento de classes
    - Para estender o comportamento de classes

Bibliografia I

Read the Docs. *Read the docs: object oriented programming in python*. 2018. `http://python-textbok.readthedocs.io/en/1.0/#`.

LUTZ, M. *Learning Python*. 5th. ed. Sebastopol, US: O'Reilly Media, 2013.

PHILLIPS, D. *Python 3 Object Oriented Programming*. 2nd. ed. Birmingham, UK: Packt Publishing, 2015.

SWAROOP, C. H. *A Byte of Python*. 1st. ed. US: CreateSpace Independent Publishing Platform, 2015.

bruno.silva@ect.ufrn.br