PROGRAMAÇÃO EM PYTHON

Fernando F. Santos

python

SENAI

AULA 14

20/03/2023

HERANÇA MÚLTIPLA

Herança múltipla é quando uma classe deriva de duas ou mais classes existentes.

É benéfico derivar todos os dados de uma vez. Ainda assim, por outro lado, ele vem com complexidade e ambiguidade de uso. É ambíguo dizer qual característica é de qual pai, se vários pais possuem a mesma característica. Ocorre quando a herança múltipla não é usada ou implementada corretamente.

EXEMPLO 1

```
class Father():
    def dirigir(self):
        print("O pai leva o filho a escola")
class Mother():
    def cozinhar(self):
        print("A mamãe adora cozinha pra seu filho")
class Son(Father, Mother):
    def ama(self):
        print("Ama seus pais")
c = Son()
c.dirigir()
c.cozinhar()
c.ama()
```

A classe filha Son é derivada das classes pai Father e Mother, o que permite usar as funções **dirigir()** e **cozinhar()** para fornecer a saída desejada.

EXEMPLO 2

```python
class Terrestre(object):
    anda_na_terra = True
    def __init__(self, velocidade):
        self.velocidade = velocidade
```

```python
from terrestre import Terrestre

class Carro(Terrestre):
    rodas = 4
    def __init__(self, velocidade_em_terra, qtd_portas):
        self.qtd_portas = qtd_portas
        super().__init__(velocidade_em_terra)
```

```python
class Aquatico(object):
    anda_na_agua = True
    def __init__(self, velocidade):
        self.velocidade = velocidade
```

```python
from aquatico import Aquatico

class Barco(Aquatico):
    def __init__(self, velocidade_na_agua, helices):
        self.helices = helices
        super().__init__(velocidade_na_agua)
```

```python
from carro import Carro
from barco import Barco

class Anfibio(Carro, Barco):
    def __init__(self, velocidade_em_terra, velocidade_na_agua,
                 qtd_portas, helices):
        self.velocidade_em_terra = velocidade_em_terra
        self.velocidade_na_agua = velocidade_na_agua
        self.qtd_portas = qtd_portas
        self.helices = helices

        Carro.__init__(self, velocidade_em_terra, qtd_portas)
        Barco.__init__(self, velocidade_na_agua, helices)
```

```python
from anfibio import Anfibio

meu_anfibio = Anfibio(120, 25, 2, 4)
print(meu_anfibio.velocidade_na_agua)
print(meu_anfibio.velocidade_em_terra)
print(meu_anfibio.qtd_portas)
print(meu_anfibio.helices)
print(meu_anfibio.anda_na_agua)
print(meu_anfibio.anda_na_terra)
print(meu_anfibio.rodas)
```

A FUNÇÃO SUPER()

- A função **super()** refere-se à classe pai ou irmã na classe filha herdada e retorna um objeto temporário que permite à classe filha usar todos os métodos da superclasse.

- Isso geralmente é feito em caso de ambiguidade, quando a herança começa a se cruzar, ou seja, quando as duas classes pai também são derivadas da superclasse de base.

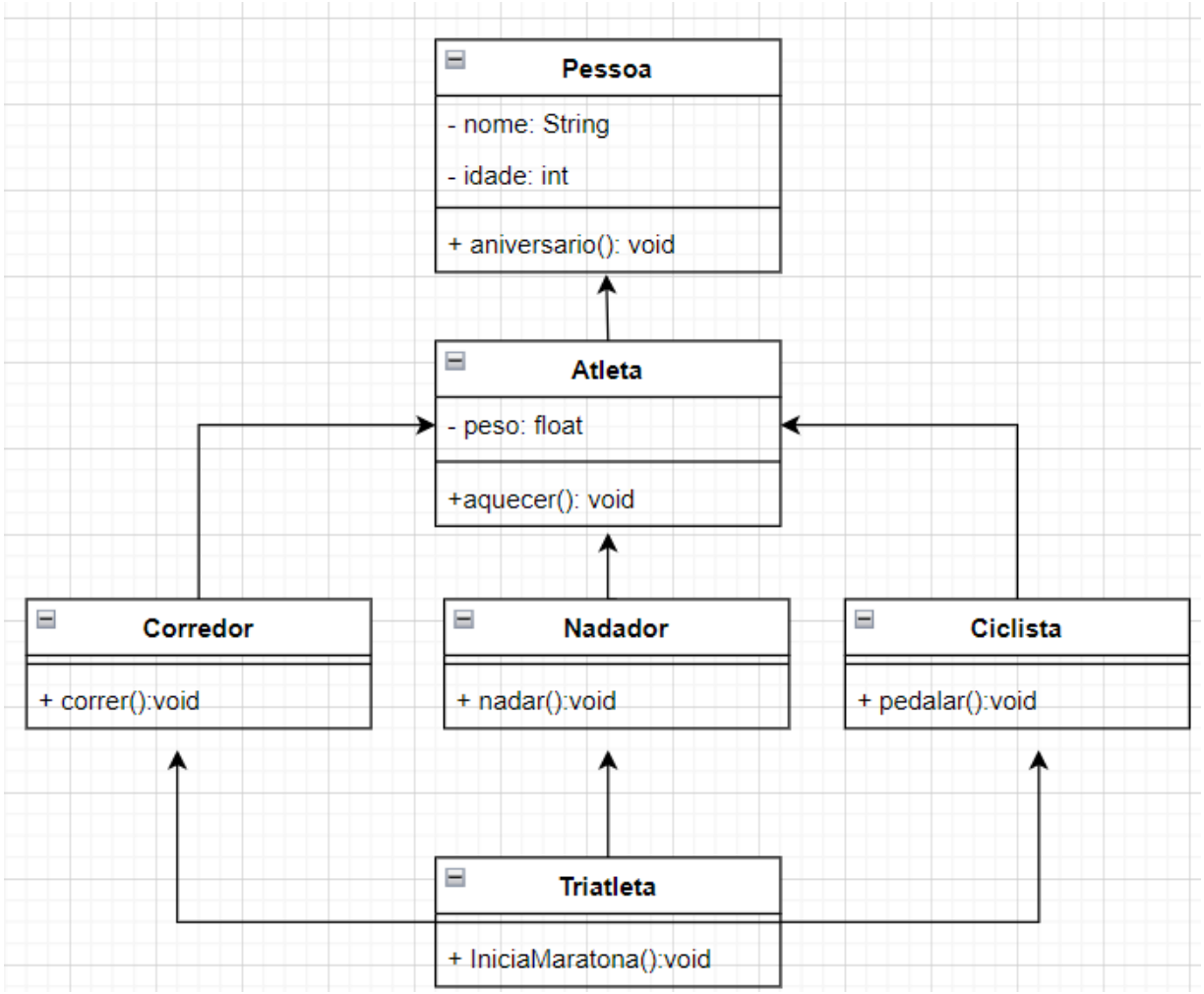
FUNÇÃO SUPER()

A função **super()** acessa os métodos herdados sobrescritos em uma classe. A função **super()** é usada na classe filha com herança múltipla para acessar a função da próxima classe pai ou superclasse. A função super() determina a próxima classe pai usando a Ordem de Resolução de Método (MRO). Como se o MRO for **C -> D -> B -> A -> object**, para D, a função super() irá procurar a próxima classe pai ou método da superclasse na sequência **D -> B -> A -> object**.

EXERCÍCIO 1

A. Construa as classes em python conforme o diagrama ao lado.

B. Os métodos apenas imprimem a ação especifica.



EXERCÍCIO 2

- Crie classes de forma a representar o diagrama a seguir:

A. A classe ContaEspecial herda da classe ContaCorrente.

B. Clientes que possuem conta especial possuem um limite de crédito. Dessa forma, podem fazer saques até esse valor limite, mesmo que não possuam saldo suficiente na conta.

C. O construtor da classe ContaEspecial deve receber como parâmetro, além dos parâmetros da superclasse, o limite que o banco disponibiliza para o cliente.

D. Sobrescreva o método sacar na classe ContaEspecial, de modo que o cliente possa ficar com saldo negativo até o valor de seu limite. Note que o atributo saldo da classe ContaCorrente deve ser do tipo protected para que possa ser modificado na subclasse.