Programação Orientada a Objetos

Herança

Parte 4

Programação Orientada a Objetos

Herança.

Herança nos permite criar uma versão modificada de uma classe existente, adicionando novos atributos e métodos.

Com herança podemos adaptar o comportamento de classes existentes sem termos que modificá-las.

Programação Orientada a Objetos

Herança.

A nova classe herda todos os métodos da classe existente.

A classe existente pode ser chamada de classe mãe, classe base ou *superclasse* e a nova classe, pode ser chamada de classe filha, classe derivada, ou *subclasse*.

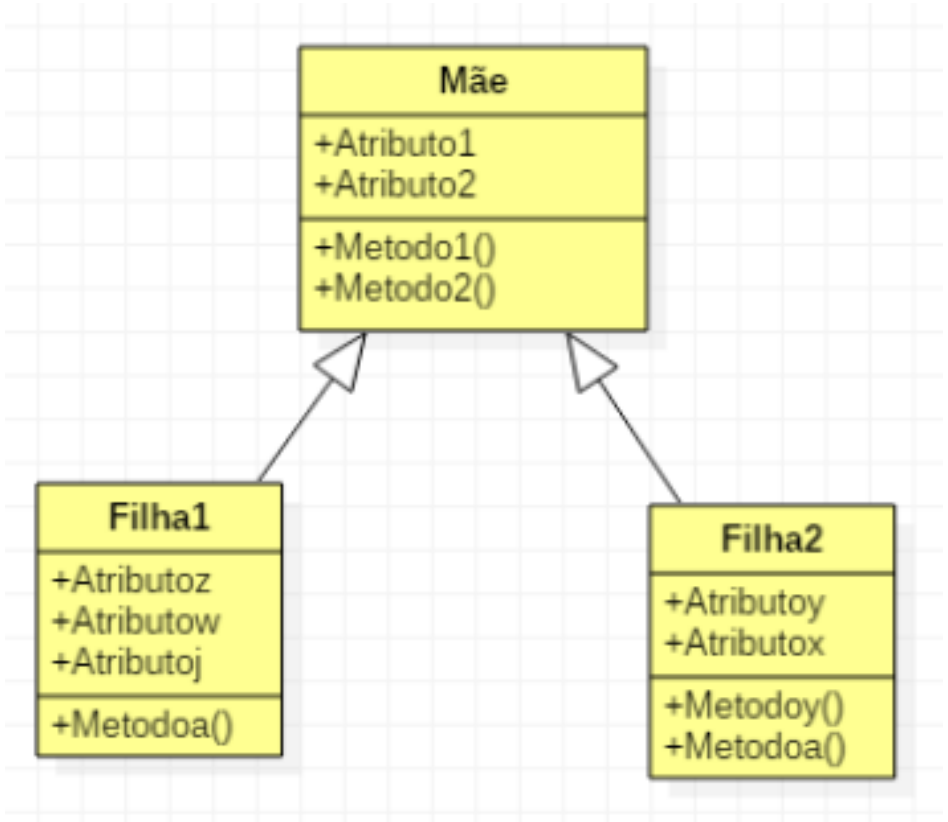
Herança facilita o reuso de código pois podemos adaptar o comportamento de classes existentes, sem ter que modificá-las.

Herança.

Para localizar os métodos e atributos, o Python procura na classe derivada, retornando pela cadeia de classes base até encontrá-los, similar ao que acontece nos namespaces local e global.

Programação Orientada a Objetos

Herança.



Programação Orientada a Objetos

Herança.

```
class Classe_base():
    def __init__(self, valor1, valor2):
        print("Método __init__ da classe base")
        self.valor1 = valor1
        self.valor2 = valor2

    def somar(self):
        return self.valor1 + self.valor2

    def subtrair(self):
        return self.valor1 - self.valor2
```

```
from classe_base import Classe_base

class Classe_derivada2(Classe_base):
    def multiplicar(self, num1, num2):
        return num1 * num2
```

```
from classe_base import Classe_base

class Classe_derivada1(Classe_base):
    def __init__(self, v1, v2):
        print("Método __init__ da classe derivada1.")
        super().__init__(v1, v2)

    def imprimir(self, texto):
        print(texto)
```

```
from classe_derivada1 import Classe_derivada1
from classe_derivada2 import Classe_derivada2

class Classe_teste():
    calculo = Classe_derivada1(10, 25)
    resultado = calculo.somar()
    print(resultado)
    resultado = calculo.subtrair()
    print(resultado)
    calculo.imprimir("Olá, este é um texto.")

    calc = Classe_derivada2(70, 85)
    resultado = calc.multiplicar(20, 10)
    print(resultado)
    resultado = calc.somar()
    print(resultado)
```

Programação Orientada a Objetos

Herança.

```
# Criando a classe base
class Pai():
    def __init__(self):
        print('Construindo a classe Pai')

# Classe filha herda da classe pai
class Filha(Pai):
    def __init__(self):
        Pai.__init__(self)  # Chamando o construtor da classe pai direto

# Criada classe mãe
class Mae():
    def __init__(self):
        print('Construindo a classe Mãe')

# Mudar a classe filha para herdar de Mae
class Filha(Mae):
    def __init__(self):
        Pai.__init__(self)  # Chamando o construtor
                            # da classe pai direto. E agora?

# Em vez de fixar a clase Pai, melhor seria usar super() para definir
# que o método __init__ chamado é o da classe base.
class Filha(Pai):
    def __init__(self):
        super().__init__(self)  # Chamando o construtor da classe pai direto
```

Programação Orientada a Objetos

Herança.

```
class Veiculo:
    """
    Classe de veículos
    """
    def __init__(self, possui_motor, quantidade_rodas):
        self.possui_motor = possui_motor
        self.quantidade_rodas = quantidade_rodas

    def ligar(self):
        if self.possui_motor:
            print("Ligou")
        else:
            print("Não tem motor.")

    def desligar(self):
        if self.possui_motor:
            print("Desligou")
        else:
            print("Não tem motor.")

    def andar(self):
        print("O veículo está andando.")

    def parar(self):
        print("O veículo foi parado.")
```

```
class Carro(Veiculo):
    """Classe carro herdando da classe Veiculo"""
    def __init__(self, quantidade_rodas):
        Veiculo.__init__(self, True, quantidade_rodas)

class Bicicleta(Veiculo):
    possui_guidao = True

    def __init__(self, quantidade_rodas):
        Veiculo.__init__(self, False, quantidade_rodas)

    def empinar(self):
        print("A bicicleta empinou.")
```

```
bike = Bicicleta(2)
print(bike.possui_guidao)
print(bike.quantidade_rodas)
bike.ligar()
bike.empinar()
bike.andar()
bike.parar()
bike.desligar()
```

```
carro = Carro(4)
print(carro.quantidade_rodas)
carro.ligar()
carro.andar()
carro.parar()
carro.desligar()
```

Programação Orientada a Objetos

Herança.

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

```
from pessoa import Pessoa

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

```
from pessoa import Pessoa

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

```
from pessoa_fisica import Pessoa_Fisica
from pessoa_juridica import Pessoa_Juridica

pf = Pessoa_Fisica("012.345.678-90", "Evaldo", 38)
pj = Pessoa_Juridica("01.000.123/0001-00", "Loja Teste", 5)

print(pf.nome)
print(pf.cpf)
print(pj.nome)
print(pj.cnpj)
```

CONTINUA...