Pesquisa - BackEnd

Orientação a objetos

Giovanna Marques de Lima.

Turma: DS-TB (18)

Herança:

Herança na programação orientada a objetos, especialmente em Python, é uma forma de criar uma nova classe usando outra como base. Ou seja, você tem uma "classe mãe" com certos comportamentos e características, e uma "classe filha" pode herdar tudo isso automaticamente. Isso evita repetir código e ainda permite que você adicione ou modifique coisas na classe filha, se precisar.
É como pegar um modelo pronto e fazer ajustes, sem ter que começar do zero. Isso torna o código mais organizado, reaproveitável e fácil de manter.

Exemplo de Herança aplicada em Python:

```python
# Classe base
class Pessoa:  2 usages
    def dizer_oi(self):  2 usages
        print("Oi!")

# Classe filha que herda de Pessoa
class Estudante(Pessoa):  1 usage
    pass

# Usando as classes
p = Pessoa()
p.dizer_oi()  # Saída: Oi!

e = Estudante()
e.dizer_oi()  # Saída: Oi!
```

A classe **Estudante** herda da classe **Pessoa.**
Mesmo sem definir nada dentro de **Estudante,** ela já pode usar o método **'dizer_oi'** que veio da classe **Pessoa.**

Ou seja, com herança, a classe **Estudante** automaticamente ganha os comportamentos da classe **Pessoa** sem precisar reescrevê-los

Vantagens de uso de Herança na arquitetura de sistemas POC

1. Reaproveitamento de código
Podemos reutilizar métodos e atributos da classe base nas classes derivadas, evitando duplicação de código e tornando o sistema mais limpo.
Ex: Se várias classes compartilham comportamentos comuns (como salvar() ou exibir()), isso pode ficar em uma classe-pai.

2. Facilidade de manutenção
Ao concentrar funcionalidades comuns em uma única classe, mudanças ou correções feitas na classe base se refletem automaticamente nas subclasses, facilitando a manutenção.

3. Organização lógica
A Herança ajuda a estruturar o sistema do nosso código de forma hierárquica.
Por exemplo: Cachorro é um tipo de Animal.

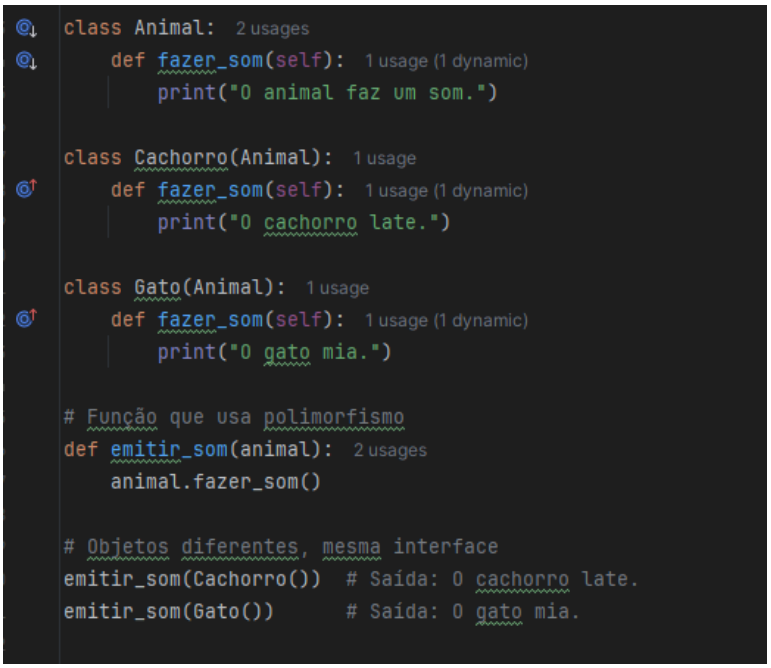
4. Extensibilidade
Podemos adicionar novos comportamentos em subclasses sem mexer no código original da classe base. Isso torna o sistema mais flexível e fácil de expandir.

5. Redução de complexidade
Herança permite dividir responsabilidades em diferentes níveis da hierarquia de classes, tornando o sistema mais modular e fácil de entender.

Polimorfismo

Polimorfismo é um conceito da programação orientada a objetos que permite que diferentes classes implementem métodos com o mesmo nome, mas com comportamentos distintos. Ele possibilita que objetos de diferentes tipos sejam tratados de forma uniforme, desde que obedeçam a uma interface comum ou a uma estrutura compatível. Dessa forma, uma mesma operação pode agir de maneira específica conforme o tipo do objeto que a invoca, promovendo flexibilidade, reutilização e extensibilidade no design de sistemas.

Exemplo de Polimorfismo aplicado em Python:



```python
class Animal:  2 usages
    def fazer_som(self):  1 usage (1 dynamic)
        print("O animal faz um som.")

class Cachorro(Animal):  1 usage
    def fazer_som(self):  1 usage (1 dynamic)
        print("O cachorro late.")

class Gato(Animal):  1 usage
    def fazer_som(self):  1 usage (1 dynamic)
        print("O gato mia.")

# Função que usa polimorfismo
def emitir_som(animal):  2 usages
    animal.fazer_som()

# Objetos diferentes, mesma interface
emitir_som(Cachorro())  # Saída: O cachorro late.
emitir_som(Gato())      # Saída: O gato mia.
```

Cachorro e **Gato** são classes diferentes, mas ambas têm o método **fazer_som().**
A função **emitir_som** aceita qualquer objeto que tenha esse método, sem se importar com o tipo específico.

Vantagens de uso de Polimorfismo na arquitetura de sistemas POC

1. Facilidade de extensão sem quebrar o código existente
Com polimorfismo, novas funcionalidades são adicionadas criando novas classes, e não mexendo nas antigas. Isso é ótimo em POCs, pois evita que estrague algo que já estava funcionando ao testar uma nova hipótese.

2. Melhor Organização do Código
Permite que cada classe tenha responsabilidade clara e única, ajudando a manter o código organizado e mais fácil de entender, mesmo em uma POC rápida.

3. Flexibilidade no Design
O polimorfismo permite que você trate objetos de diferentes classes de forma uniforme, contanto que implementem a mesma interface ou herdem da mesma superclasse.

Relação e Diferenças entre Herança e Polimorfismo

Herança permite que uma classe herde métodos e atributos de outra, criando uma base comum entre elas.
Polimorfismo aproveita essa base comum para permitir que objetos de diferentes classes respondam ao mesmo método de formas diferentes.

Ou seja, **a herança cria o "formato comum"** (como um método falar() em Animal), e **o polimorfismo permite que cada subclasse (Cachorro, Gato) implemente esse método do seu jeito**, mesmo sendo tratadas como Animal.

A herança fornece a estrutura compartilhada. O polimorfismo dá flexibilidade para que cada classe se comporte de forma única dentro dessa estrutura.

Referências utilizadas:

Oracle Java Tutorials – Object-Oriented Programming Concepts

Python Docs – Classes