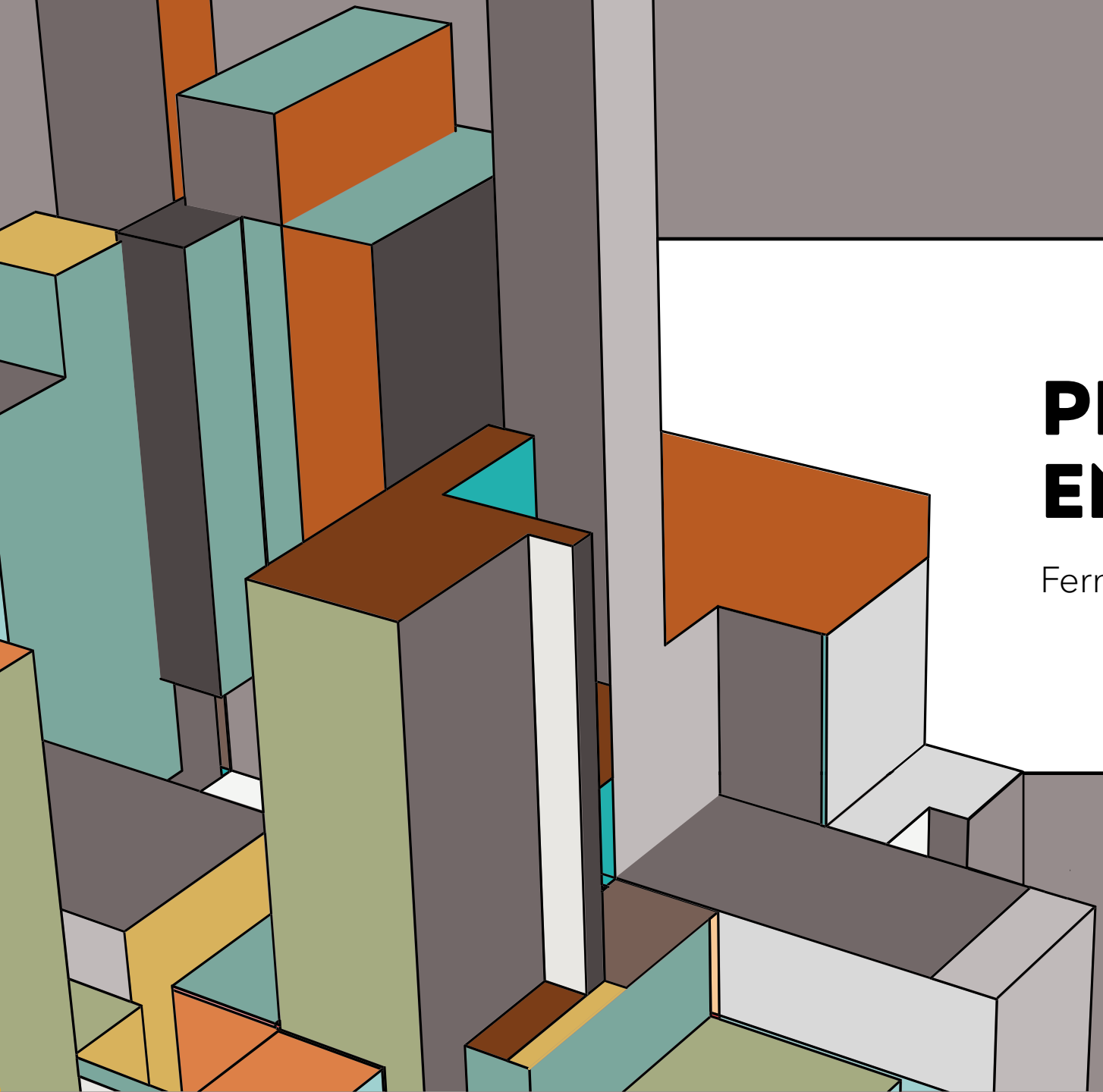


PROGRAMAÇÃO EM PYTHON

Fernando F. Santos

python

SENAI

AULA 15

22/03/2023

POLIMORFISMO

O polimorfismo é baseado nas palavras gregas Poli (muitas) e morphos (formas).

É a capacidade de um objeto adaptar o código ao tipo de dados que está processando.

Objetos podem ser de muitos tipos ou formas. Por exemplo, botões possuem formas diferentes, existem botões redondos, quadrados, botões com imagem, mas todos compartilham a mesma lógica, executam uma ação chamada “clique”. A capacidade de acessarmos o método “clique” independente do tipo de botão é chamada de polimorfismo.

TIPOS

- Um objeto oferece diferentes implementações do método de acordo com os parâmetros de entrada;
- A mesma **interface** pode ser usada por objetos de tipos diferentes.

INTERFACES - Classes com comportamentos semelhantes

EXEMPLO

```
class Cachorro(object):
    def som(self):
        print("Au au")
class Gato(object):
    def som(self):
        print("Miau")
def emitir_som(animal):
    animal.som()
cachorrinho = Cachorro()
gatinho = Gato()
emitir_som(cachorrinho)
emitir_som(gatinho)
```

Esta é uma forma de utilizarmos polimorfismo com função. Criamos duas classes: Cachorro e Gato, ambos podem fazer um som distinto. Em seguida, criamos duas instâncias e chamamos sua ação usando o mesmo método denominado “som”

EXEMPLO

```
class Cachorro(object):
    def som(self):
        print("Au au")
class Gato(object):
    def som(self):
        print("Miau")
def emitir_som(animal):
    animal.som()
cachorrinho = Cachorro()
gatinho = Gato()
emitir_som(cachorrinho)
emitir_som(gatinho)
```

A classe Cachorro possui um método chamado “som” que imprime o texto “Au au”

A classe Gato possui um método chamado “som” que imprime o texto “Miau”

A função emitir_som recebe um objeto chamado “animal” e executa o método “som” desse objeto.

Foram instanciados dois objetos, um da classe Cachorro e outro da classe Gato.

A função emitir_som foi executado duas vezes, na primeira passando um objeto do tipo Cachorro e na segunda, um objeto do tipo Gato.

EXEMPLO 2

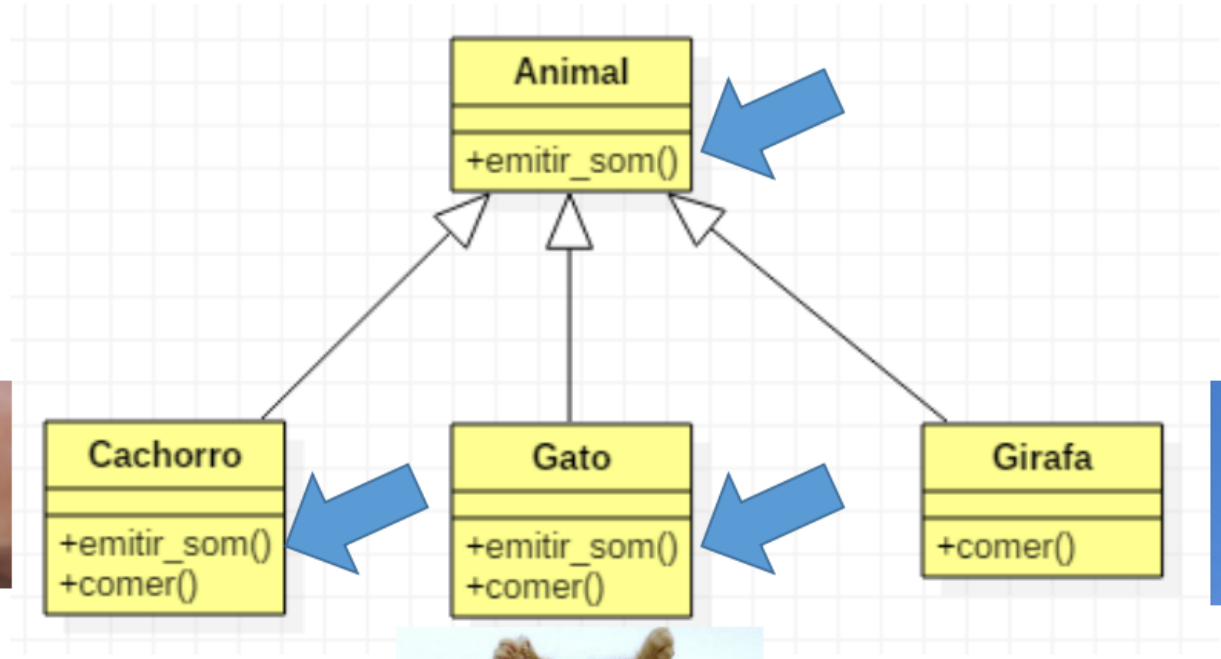
```
class Porta(object):
    def fechar(self):
        print("A porta foi fechada!")

    def abrir(self):
        print("A porta foi aberta!")

class Janela(object):
    def fechar(self):
        print("A janela foi fechada!")

    def abrir(self):
        print("A janela foi aberta!")
```

```
def realizar_abertura(o_que_abrir):
    o_que_abrir.abrir()

def realizar_fechamento(o_que_fechar):
    o_que_fechar.fechar()

porta = Porta()
janela = Janela()

realizar_abertura(porta)
realizar_abertura(janela)
realizar_fechamento(porta)
realizar_fechamento(janela)
```

CLASSES ABSTRATAS

- O mais comum é utilizarmos polimorfismo com **classes abstratas.**
- Uma classe abstrata é uma classe que não possui implementação, possuindo apenas uma estrutura com os métodos que devem ser implementados nas subclasses.
- Desta forma, os métodos devem ser obrigatoriamente implementados na subclasse.

EXEMPLO


Animal
+emitir_som()
Cachorro
+emitir_som()
+comer()
Gato
+emitir_som()
+comer()
Girafa
+comer()
Au, au
Miau
?

EXEMPLO

```
class Animal(object):
    def emitir_som(self):
        raise NotImplementedError("Não foi
implementado o método emitir_som.")
class Cachorro(Animal):
    def emitir_som(self):
        print("Au au")
class Gato(Animal):
    def emitir_som(self):
        print("Miau")
```

```
class Girafa(Animal):
    def comer(self):
        print("A girafa está comendo.")
def barulho(animal):
    animal.emitir_som()
cachorrinho = Cachorro()
gatinho = Gato()
girafinha = Girafa()
barulho(cachorrinho)
barulho(gatinho)
barulho(girafinha)
```

CORRIGINDO O EXEMPLO

- Quando eu implemento um método na classe base (não será mais uma classe abstrata) e o mesmo não é implementado na subclasse, será executado o método da classe base

```
class Animal(object):
    def emitir_som(self):
        print("Um som qualquer")
class Girafa(Animal):
    def comer(self):
        print("A girafa está comendo.")
```

```
def barulho(animal):
    animal.emitir_som()
girafinha = Girafa()
barulho(girafinha)
```

EXERCÍCIOS

1. **Classe Conta de Investimento:** Faça uma classe contaInvestimento que seja semelhante a classe contaBancaria, com a diferença de que se adicione um atributo taxaJuros. Forneça um construtor que configure tanto o saldo inicial como a taxa de juros. Forneça um método adicioneJuros (sem parâmetro explícito) que adicione juros à conta. Escreva um programa que construa uma poupança com um saldo inicial de R$1000,00 e uma taxa de juros de 10%. Depois aplique o método adicioneJuros() cinco vezes e imprime o saldo resultante.
2. **Classe Funcionário:** Implemente a classe Funcionário. Um empregado tem um nome (um string) e um salário(um double). Escreva um construtor com dois parâmetros (nome e salário) e métodos para devolver nome e salário. Escreva um pequeno programa que teste sua classe.
3. **Classe Bichinho Virtual:** Crie uma classe que modele um Tamagushi (Bichinho Eletrônico):
   a. Atributos: Nome, Fome, Saúde e Idade
   b. Métodos: Alterar Nome, Fome, Saúde e Idade; Retornar Nome, Fome, Saúde e Idade
   c. Obs: Existe mais uma informação que devemos levar em consideração, o Humor do nosso tamagushi, este humor é uma combinação entre os atributos Fome e Saúde, ou seja, um campo calculado, então não devemos criar um atributo para armazenar esta informação por que ela pode ser calculada a qualquer momento.