



A woman with long dark hair, wearing sunglasses, is shown from the chest up, looking down at her smartphone. The background is a dark, slightly blurred image of her hair and face.

IN

# INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

# PY - POO II

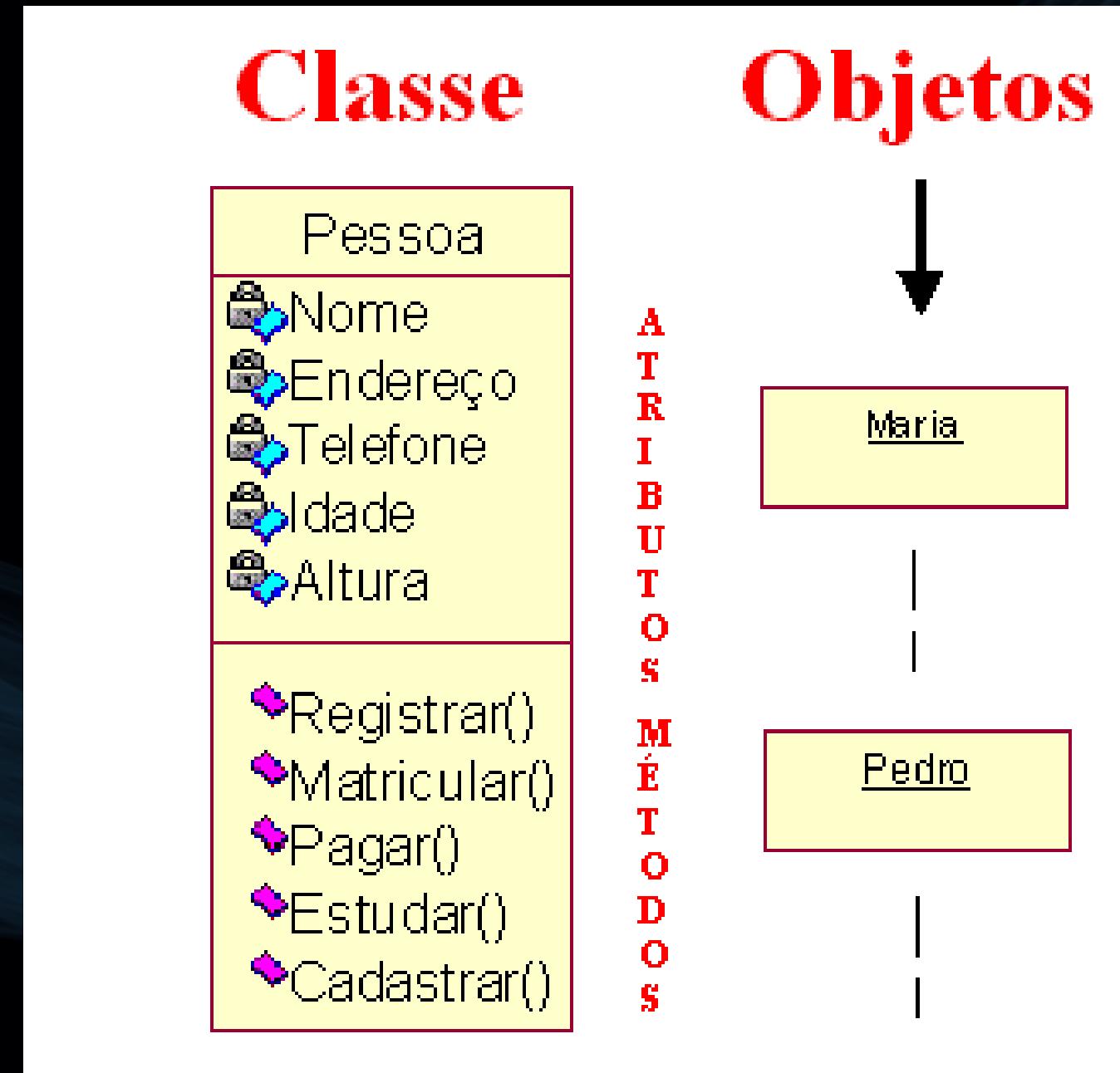
**01** Relembrando conceitos importantes

**02** Abstraindo mais as classes

**03** Herança

**04** Exercício





# PY - POO II

## Relembmando conceitos

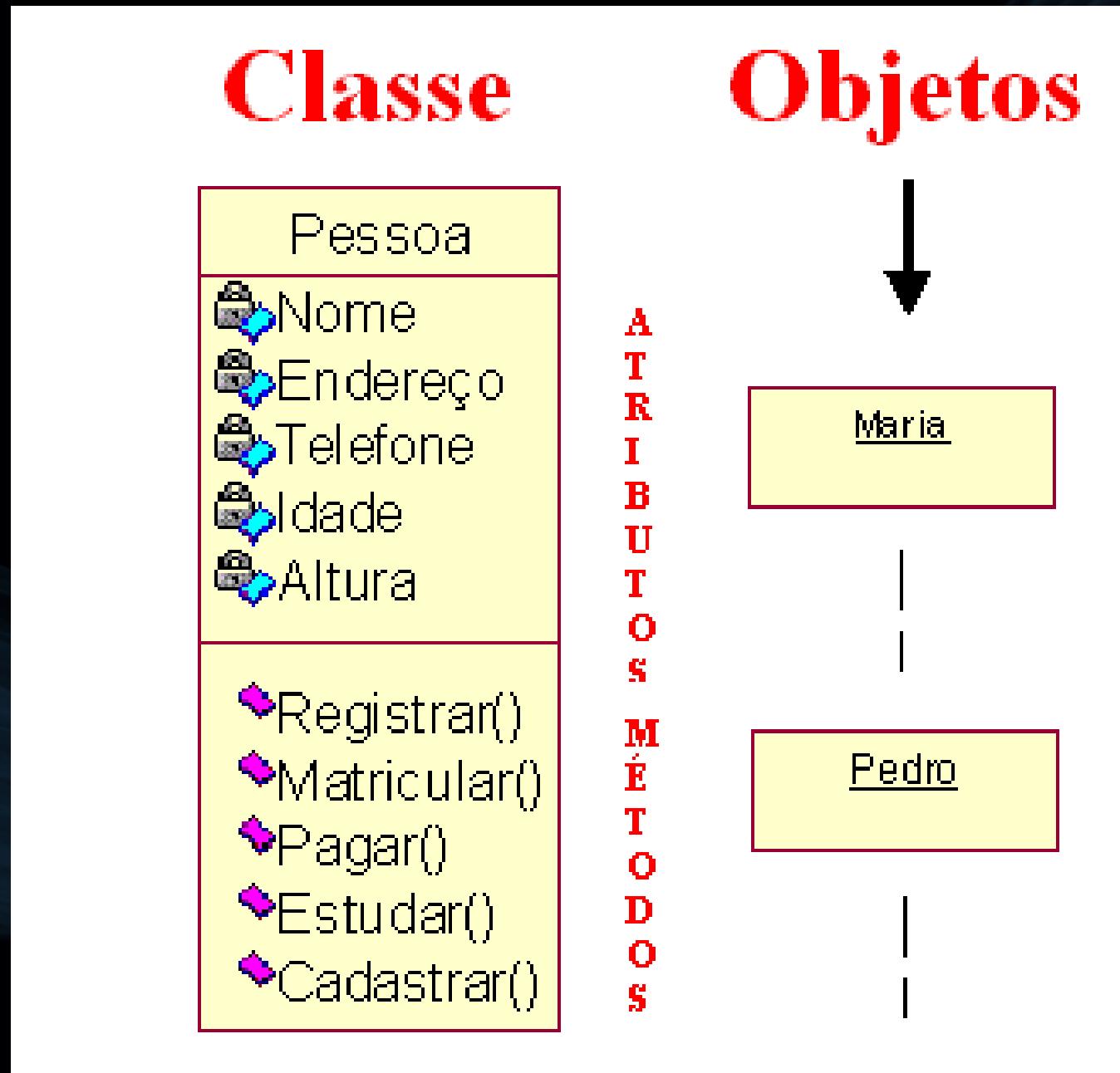
Como vimos na última aula, a programação orientada a objetos é uma programação "baseada em dados" e tem como foco os tipos abstratos de dados.

Conseguimos entender que um modelo orientado a objetos possui alguns elementos básicos: Objetos, métodos e classes.

Vimos que nossa classe é uma "*super variável*" que armazena atributos (variáveis) e métodos (funções), e que a partir dessas classes, instanciamos nossos objetos, que de fato é o que temos de concreto.

# PY - POO II

## Classe



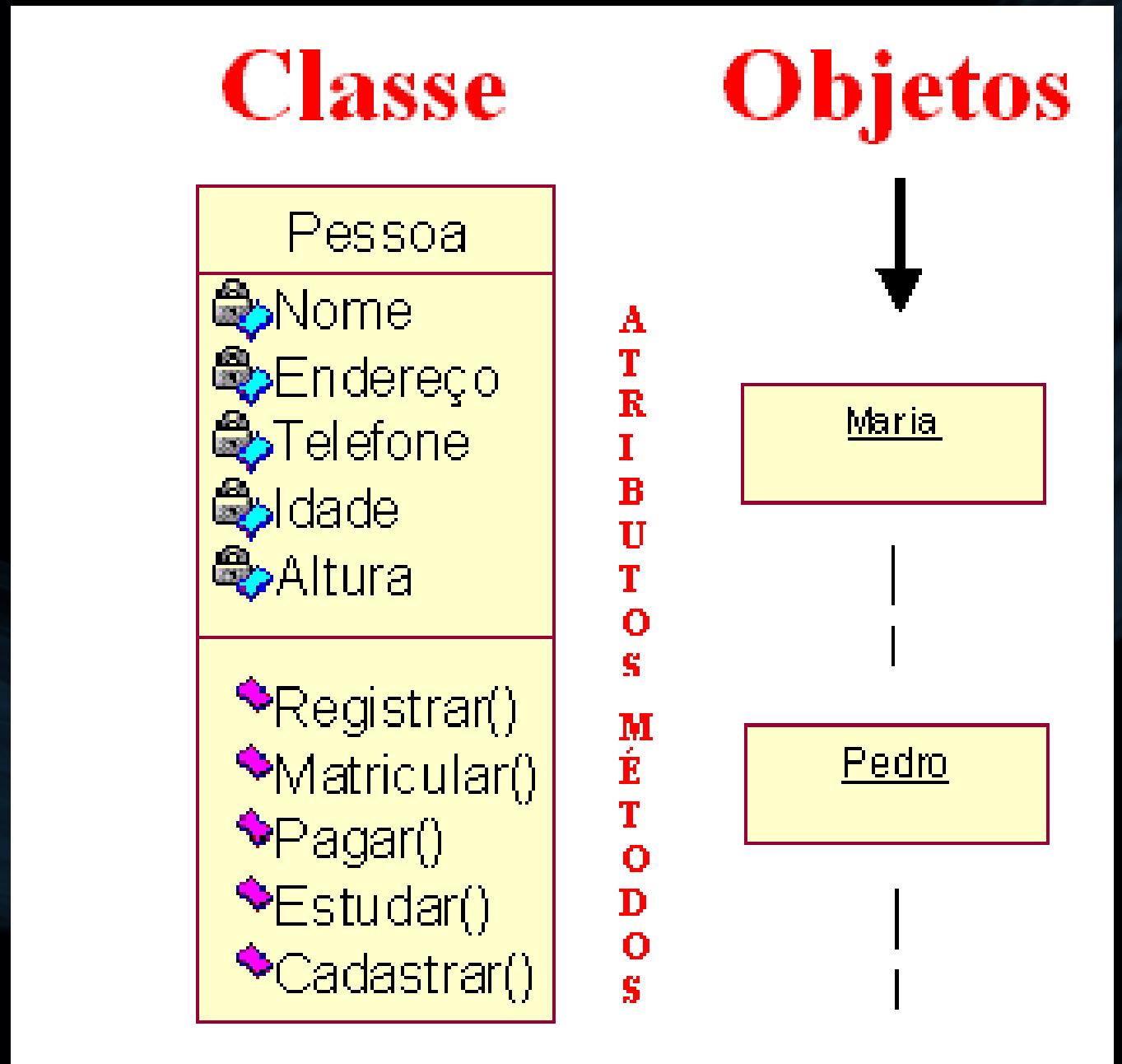
É a abstração que temos de algo do nosso mundo, é a estrutura que vamos implementar no sistema que representará algo do mundo real.

Ou seja, quando falamos de classe, falamos de abstração, de algo não concreto, que existe apenas em nossos sistemas e em nossas mentes.

A partir dela temos os objetos.

# PY - POO II

## Objeto

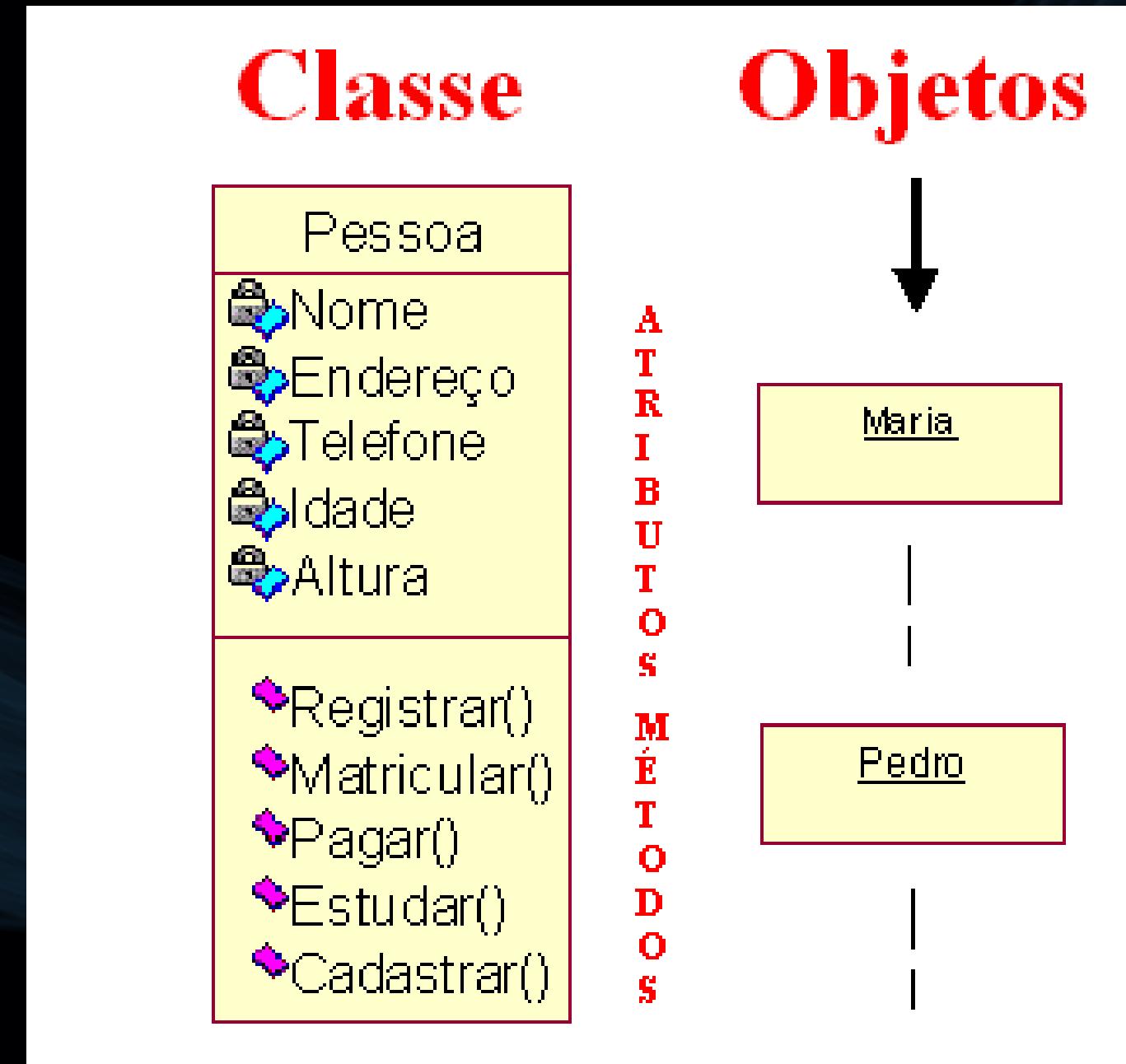


São instâncias das classes, a parte concreta, onde de fato os dados estarão armazenados.

Um objeto também pode, como parte da sua operação, enviar mensagens para outros objetos e para si mesmo. Para ilustrar, pode-se dizer que objetos representam uma coleção dados relacionados com um tema em comum.

# PY - POO II

## Atributo

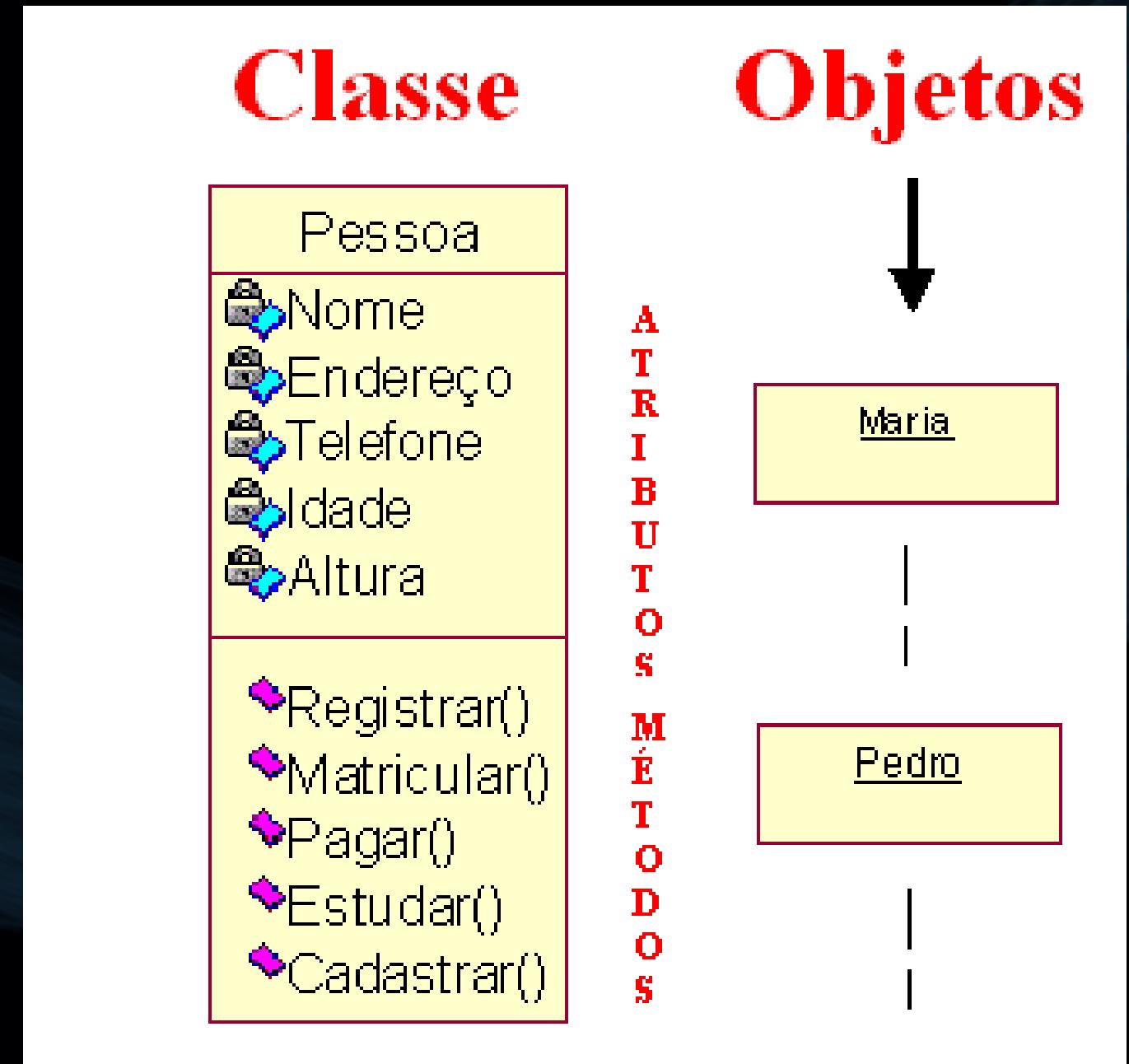


Relembrando o que é um objeto, vimos que ele possue características.

Podemos dizer que os atributos dos objetos são “variáveis” ou “campos” que armazenam os diferentes valores que as características dos objetos podem conter

# PY - POO II

## Método



Um método implementa algum aspecto do comportamento do objeto.

Comportamento é a forma como um objeto age e reage, em termos das suas trocas de estado e troca de mensagens.

Os métodos são as ações que o objeto pode realizar. Tudo que o objeto faz é realizado através de seus métodos

# PY - POO II

## Mão no código

Para fixar o que vimos na aula anterior, vamos realizar um pequeno exercício.

Crie uma classe chamada Cachorro que possua 3 atributos: nome, raça e peso. Implemente um método: comer ração, que retorna "croc croc croc".

Crie duas instâncias dessa classe, e imprima na tela: "O cachorro *nome\_do\_cachorro* é da raça *raça\_do\_cachorro*, pesa *peso\_do\_cachorro* quilos e come: *croc croc croc*".

Crie outra classe chamada Gato que possua os mesmos atributos e o possua um método chamado derrubar coisas que retorna "O gato *nome\_do\_gato* quebrou algo". Faça a mesma impressão na tela para a classe gato.



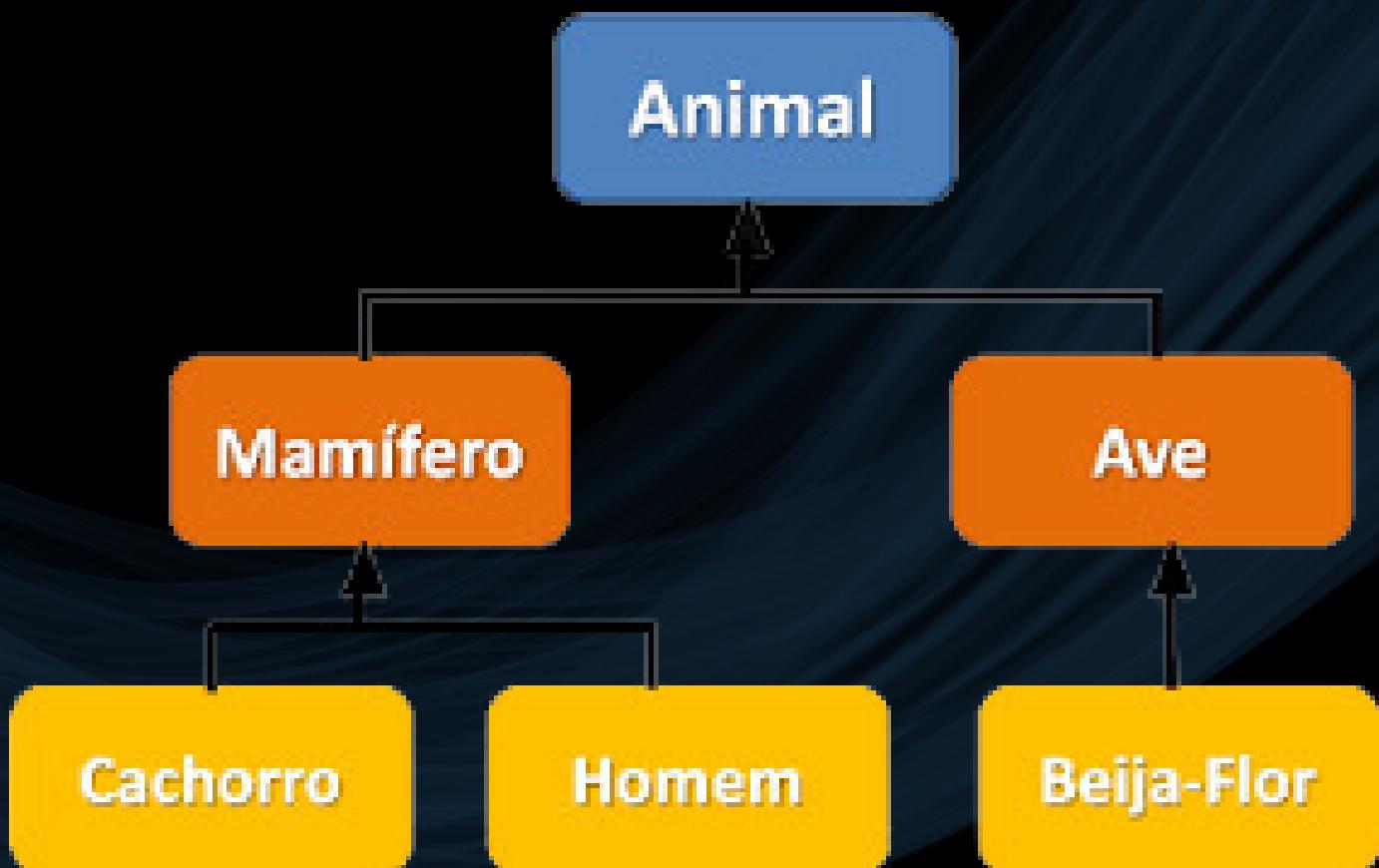
# PY - POO II

## Superclasses

Classes puras, Classes abstratas ou Superclasses, classes pai, são nomes distintos para representar a mesma ideia. São classes das quais os objetos **nunca** são instanciados diretamente, mas **sempre** por uma classe **descendente** dela, conhecidas como subclasses, classes filha, por exemplo.

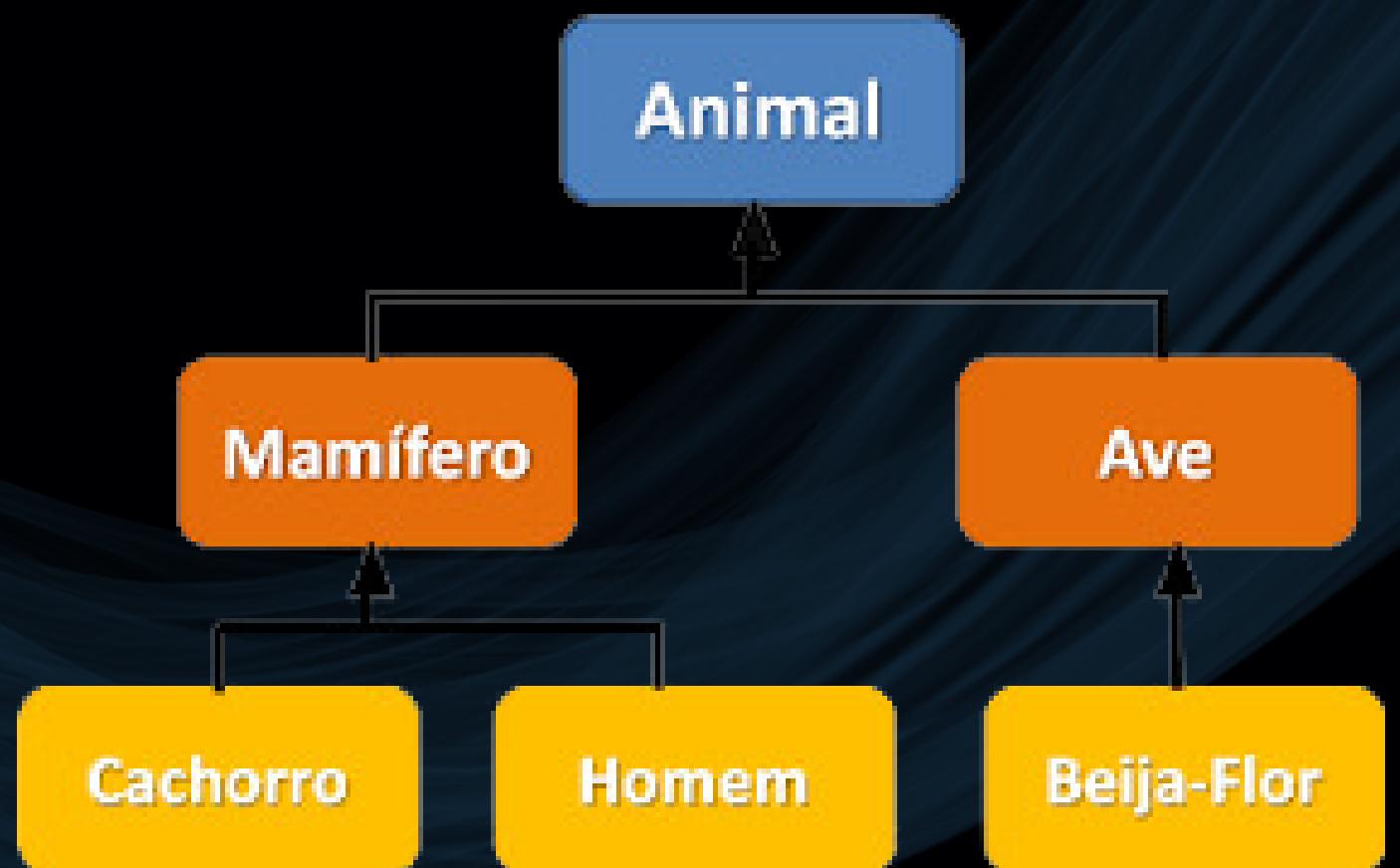
Essas classes são criadas para facilitar o processo de estruturação.

Um exemplo clássico é criar uma classe Pessoa, que contém os atributos (nome, endereço, telefone, etc.) e métodos (alteração de endereço, imprimir ficha, etc.) necessários para manusear dados de pessoas em um sistema de informação. A partir dessa classe genérica, criam-se classes descendentes específicas para manusear funcionário, gerente, etc.



# PY - POO II

## Superclasses



Neste exemplo, a classe Animal nunca terá uma instância de objeto, mas, terá duas classes filhos: Mamífero e Ave. E essas sim terão suas instâncias de objeto.

É importante destacar que uma classe filha **HERDA** todos os atributos e métodos da classe pai, ou seja, tanto a classe Mamífero quanto a classe Ave terão os atributos e métodos da classe Animal.

## PY - POO II



```
class Pai:  
    def __init__(self, peso, altura) -> None:  
        self.peso = peso  
        self.altura = altura  
  
class Filho(Pai):  
    def __init__(self, nome, idade) -> None:  
        self.nome = nome  
        self.idade = idade  
  
teste = Filho(nome = 'Teste', idade = 100, peso = 75, altura = 1.80)  
  
Traceback (most recent call last):  
  File "X:\SOnoixDEV\Ariel\tkinterII\a.py", line 11, in <module>  
    teste = Filho(nome = 'Teste', idade = 100, peso = 75, altura = 1.80)  
                                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  
  
TypeError: Filho.__init__() got an unexpected keyword argument 'peso'
```

## Superclasses em Python

É importante ter em mente que para que a classe filha herde os atributos da classe pai, utilizamos a função **super()**.

Se não a utilizarmos, o método construtor da classe filha irá sobrescrever o método construtor da classe pai, logo, não herdando seus atributos.

## PY - POO II



```
class Pai:  
    def __init__(self, peso, altura) -> None:  
        self.peso = peso  
        self.altura = altura  
  
class Filho(Pai):  
    def __init__(self, nome, idade, peso, altura) -> None:  
        super().__init__(peso, altura) # Função super  
        self.nome = nome  
        self.idade = idade  
  
teste = Filho(nome = 'Teste', idade = 100, peso = 75, altura = 1.80)
```

## Superclasses em Python

E para que a classe filha herde os atributos do pai e tenha seu método construtor próprio, utilizamos a função **super()** em conjunto com o método construtor, como se estivéssemos instanciando a classe pai dentro do método construtor da classe filha.

# PY - POO II

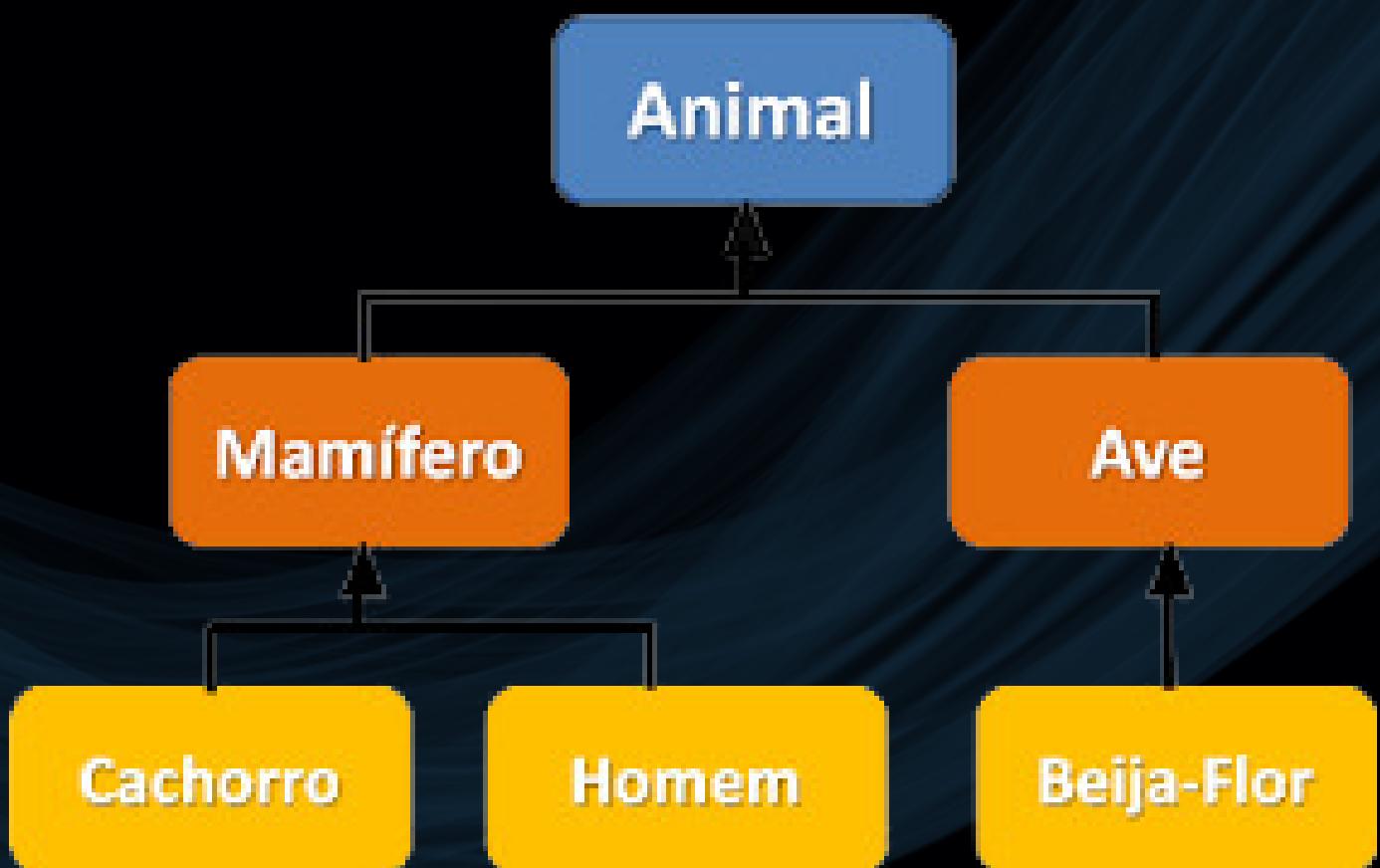
## Herança

Herança é um dos pontos chave de programação orientada a objetos (POO).

A ideia de herança é facilitar a programação. **Uma classe A deve herdar de uma classe B quando podemos dizer que A é um B.**

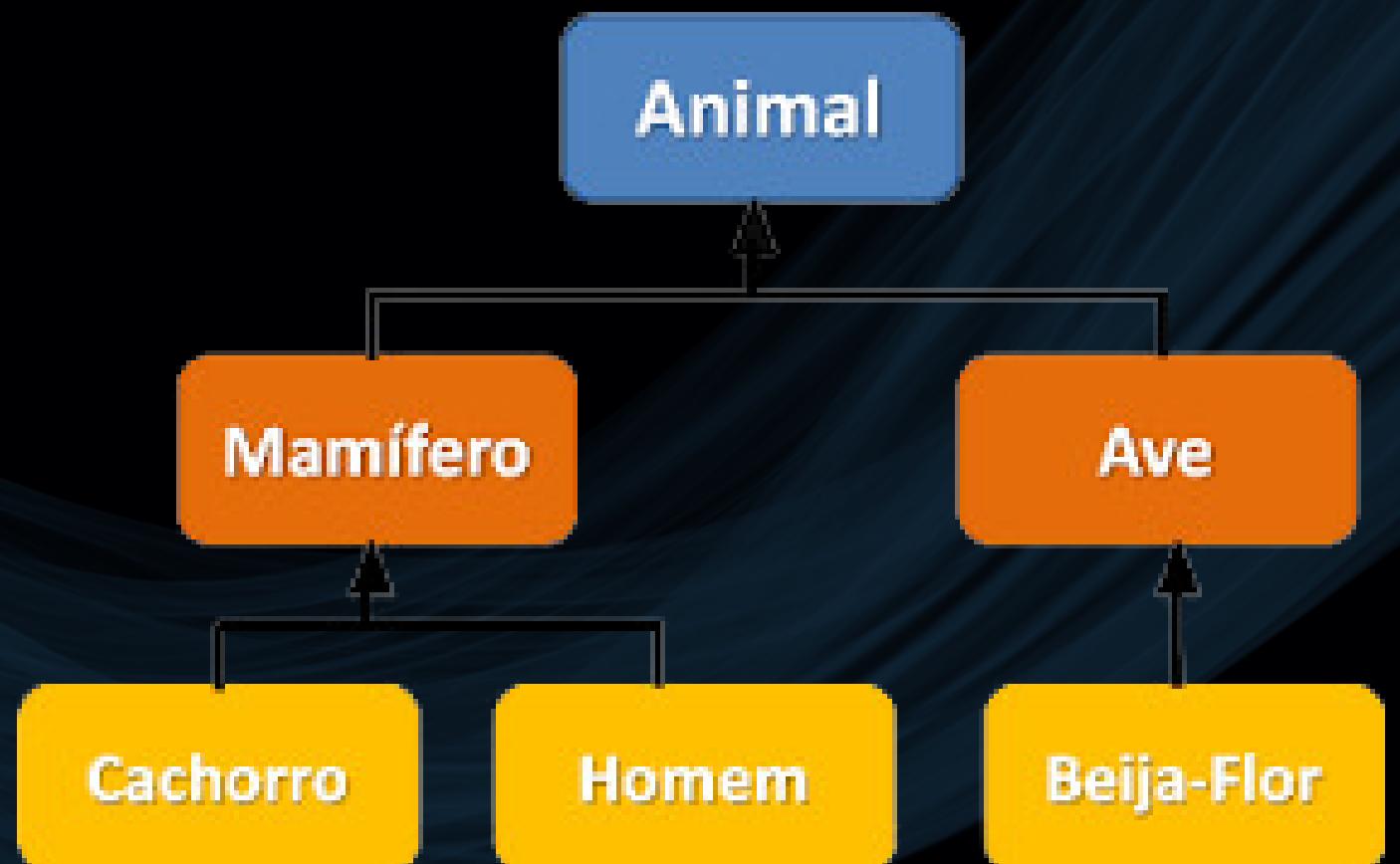
O mecanismo de herança permite a reutilização das propriedades de uma classe na definição de outra.

Ou seja, herança acontece quando duas classes são próximas, têm características mútuas mas não são iguais e existe uma especificação de uma delas.



## PY - POO II

## Herança



Portanto, em vez de escrever todo o código novamente é possível poupar algum tempo e dizer que uma classe herda da outra e depois basta escrever o código para a especificação dos pontos necessários da classe derivada (classe que herdou).

A herança é uma parte importante da orientação a objetos porque permite a reutilização de código existente e facilita o projeto, já que não temos que colocar todos os códigos dentro de um único arquivo.



```
1 class Animal(): # Classe pai
2     def __init__(self, nome, cor):
3         self.__nome = nome
4         self.__cor = cor
5
6     def comer(self):
7         print(f"O {self.__nome} está comendo")
8
9
10 class Cachorro(Animal): # Classe filha
11     def __init__(self, nome, cor):
12         super().__init__(nome, cor)
13
14 class Gato(Animal): # Classe filha
15     def __init__(self, nome, cor):
16         super().__init__(nome, cor)
17
18
19 gato = Gato("Bichano", "Branco")
20 cachorro = Cachorro("Totó", "Preto")
21
22 gato.comer() # Método declarado na classe pai
23 cachorro.comer() # Método declarado na classe pai
```

# PY - POO II

## Herança

Veja as classes Cachorro e Gato.

Por herdar da classe Animal, as classes Gato e Cachorro podem, sem nenhuma alteração, utilizar o método **comer()**, definido na classe Animal, pois elas herdam dessa classe, logo, elas possuem a permissão de invocar este método.

E elas também possuem os atributos nome e cor, herdados da superclasse Animal.

- 0 Bichano está comendo
- 0 Totó está comendo

# PY - POO II

## Mão no código

Refatore o código do exercício anterior utilizando a ideia de superclasse para as classes gato e cahorro, lembre-se de utilizar a função super.



# PY - POO II

## Mão no código

Pense em sistema bancário, nesse sistema há 2 contas: conta-corrente e conta poupança.

Ambas possuem um identificado numérico, nome do titular e saldo. Possuem também dois métodos: verificar informações, que retorna todas as informações daquela conta e, depósito, que adiciona determinado valor ao saldo daquela conta.

Apenas a conta-corrente possui também o método sacar que subtrai determinado valor da conta.

Apenas a conta poupança possui também um atributo chamado juros.

Quem utilizará esse sistema é um gerente, portanto, o sistema deve permitir que ele crie uma conta-corrente ou conta poupança nova e, que consiga utilizar todas as funções.





IN

# INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

IN PARABÉNS! VOCÊ TERMINOU A AULA 09 DO MÓDULO DE PYTHON