Programação Orientada a Objetos

HERANÇA

No mundo real as pessoas herdam as caraterísticas dos seus pais que, por sua vez, as herdaram dos seus. Em POO é possível definir uma hierarquia de classe em que cada classe "herda" as características das suas superiores

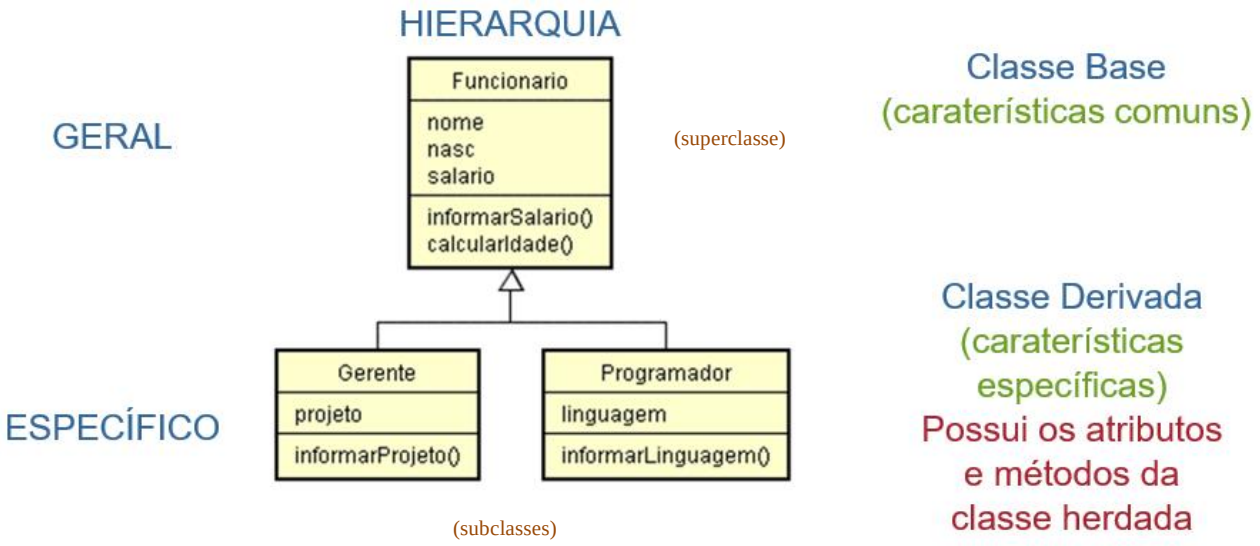


Fig. 2 – Estrutura hierárquica com a classe "Funcionário" como superclasse

Na estrutura hierárquica da figura 2, a superclasse "Funcionário" possui os atributos e métodos comuns a todos os objetos Funcionários. Cada uma das subclasses possui os atributos e métodos específicos de cada uma delas, herdando os atributos e métodos da superclasse.

Se uma classe for declarada como uma subclasse de outra classe a sintaxe será a seguinte:

```
class nome_da_ classe (Nome_da_superclasse) :
      def __init__ (self, p1, p2, …Pm, Pm+1…Pn):
            Nome_da_superclasse.__init__(self,P1, …Pm):
                  self.Pn=pm+1
                  …
                  self.Pn=Pn
            …
```

CONCEITO DE POLIMORFISMO

O termo `polimorfismo` o é originário do grego e significa "muitas formas". Aplicado a linguagem orientada por objetos, o polimorfismo é uma técnica de permite implementar código para um objeto em que são criadas várias versões desse objeto para ser utilizado em contextos diferentes, com um modo de funcionar adequado a cada contexto.
O polimorfismo aplica-se apenas aos métodos da classe e, por defeito, exige a utilização da herança. A comunicação é estabelecida apenas na classe mais alta da hierarquia, contudo, como as subclasses estão ligadas por herança, é possível essas classes "receberem" a comunicação.
Com o conceito de polimorfismo, é possível acrescentar novos métodos a classes já existentes, sem a necessidade de recompilar a aplicação.
Por vezes também se usa a palavra `sobrecarga` (em inglês, `overload` ou `override`) para referir esta caraterística.
Pode-se dizer que a execução das versões do método irá depender da instância da classe que for criada.

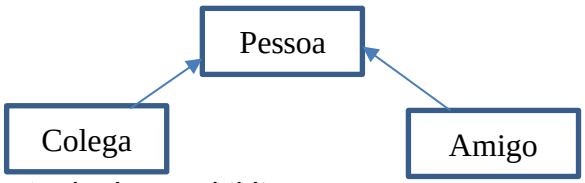
"**`Polimorfismo` é o princípio pelo qual `duas ou mais classes derivadas` de uma `mesma superclasse` podem invocar métodos que têm a `mesma identificação`, `assinatura` (`que é a sua lista de parâmetros`), mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse**. "

Exercícios (consulta o NB10 e NB11. 10794 POO em Python):

1. Para cada alínea grava um programa com o nome: **F3_ex1_a.py** , em que **a** é o nº da alínea)

`Hierarquia de classes`

1.1 Define a classe Pessoa e as subclasses Amigo e colega que derivam da classe Pessoa. Cada classe e os seus atributos e métodos deve ser guardada separadamente num programa, de nome F3Pessoa.py, F3HAmigo.py e F3HColega.py, respetivamente.



Nota: Procede à importação de classes e bibliotecas, sempre que necessário.
A classe Pessoa *encapsula*:

- Duas variáveis de instância, nome e telefone;
- Construtor;
- Acessos para ler os valores das variáveis de instância.

A subclasse Amigo *encapsula*:

- Duas variáveis de instância, local e ano de início da amizade;
- Construtor;
- Acessos para ler os valores das variáveis de instância.

A subclasse Colega *encapsula*:

- Duas variáveis de instância, local de trabalho e profissão;
- Construtor;
- Acessos para ler os valores das variáveis de instância.

No programa principal:

- Cria as instâncias para o:
    - Colega Pedro, telefone: 917675522, Profissão Engenheiro, local de trabalho: UBI
    - Amiga Rita, telefone: 915553344, local e ano de início de amizade: Lisboa, 2015
    - Imprime os dados do colega e da amiga, através dos métodos

Output esperado:

```
Pedro     917675522     Engenheiro      UBI
Rita      915553344     Lisboa          2015
```

Respeita os alinhamentos de saída.

Métodos de impressão para as 3 classes

1.2. Dada a hierarquia de classe do exercício anterior, define para a classe Pessoa um método de impressão das suas variáveis de instância. As subclasses Amigo e Colega deverão também poder, cada uma, invocar este método de Pessoa para imprimir as suas variáveis de instância (que são genéricas).
As classes deverão, agora, ter os nomes F3Pessoa2.py, F3HAmigo2.py e F3HColega2.py, respetivamente.

No programa principal:

- Cria as instâncias para o:
    - Colega Pedro, telefone: 917675522, Profissão Engenheiro, local de trabalho: UBI
    - Amiga Rita, telefone: 915553344, local e ano de início de amizade: Lisboa, 2015

- Imprime os dados do colega e da amiga (invocando os respetivos métodos de impressão).

1.3 Dada a hierarquia de classe do exercício anterior, define acessos às variáveis de instância usando propriedades. Os dados deverão ser fornecidos pelo utilizador.
As classes deverão, agora, ter os nomes F3Pessoa3.py, F3HAmigo3.py e F3HColega2.py, respetivamente.

Output de exemplo esperado:

```
Dados do/a colega:
Nome: Pedro
Telefone: 917675544
Profissão: Professor
Local de trabalho: AEFHP
- - - - - - - - - - - - - - - - - - - - - - -
Dados do/a amigo/a:
Nome: Alice
Telefone: 913434444
Início da amizade:
    Local: Covilhã
    Ano: 2010
Pedro     917675522     Engenheiro      UBI
Rita      915553344     Lisboa          2015
```

2. Equipas e Jogadores

Uma equipa de determinado jogo tem vários jogadores. Cada jogador é instanciado a partir da classe Jogador:

```
class Jogador:
    def __init__(self,nome, altura, peso):
        self.nome=nome
        self.altura=altura
        self.peso=peso
    def imprime(self):
        return self.nome + "---"+str(self.altura)+ "cm ---
"+str(self.peso)+"kg"
```

2.1. Completa os métodos da classe Equipa (grava o ficheiro com o nome **F3_2_1.py**):

- A classe equipa deve conter um método construtor para o atributo nome da equipa e uma variável de classe para criar a lista;
- Um método para adicionar os nomes dos jogadores efetivos numa lista;
- Um método para listar os jogadores efetivos.

2.2. Elabora um programa para adicionar jogadores à equipa e imprimir a lista dos jogadores.

Output:

```
In [9]: runfile('C:/DD/ESCOLA 22-23/PI6/PSD 11 - PI6/python_ ex_spyder/F3_2_2.py'
wdir='C:/DD/ESCOLA 22-23/PI6/PSD 11 - PI6/python_ ex_spyder')

nome do jogador (ou z para parar)-> Cristiano Ronaldo

Altura (em cm)->187

Peso (em Kg)->85

nome do jogador (ou z para parar)-> Rafael Leão

Altura (em cm)->188

Peso (em Kg)->81

nome do jogador (ou z para parar)-> Bruno Fernandes

Altura (em cm)->179

Peso (em Kg)->69

nome do jogador (ou z para parar)-> z
Cristiano Ronaldo---187cm ---85kg
Rafael Leão---188cm ---81kg
Bruno Fernandes---179cm ---69kg
```

Bibliografia:

https://www.usandoaccess.com.br/tutoriais/classe-no-access-orientacao-a-objetos.asp
Carvalho. Adelaide. (2021). Práticas de Python - Algoritmia e programação. Lisboa: FCA
Vasconcelos, J. (2015). Python - Algoritmia e Programação Web. Lisboa: FCA