

Aula 8

Docupedia Export

Author:Sobolevski Nycollas (CtP/ETS)

Date:06-Mar-2024 14:42

Table of Contents

1	Programação Orientada a Objetos (POO)	3
2	Problemas em usar essa metodologia	5
3	Conceitos	6
4	Métodos e Atributos	7
5	Construindo e destruindo objetos	8
6	Métodos estáticos	9
7	Atributos	10
8	Existem 4 pilares da Programação Orientada a Objetos	12
9	Exercício 2	14
10	Exercício 3	15

1

Programação Orientada a Objetos (POO)

Programação Estruturada

Assim como visto nas aulas anteriores, a programação estruturada é o algoritmo que realiza as operações em forma **sequencial** com decisões e repetições.

Problemas quando possui o mesmo tipo de dados na programação estruturada.

Ex: Definir dados das casas de uma cidade.

- **João** tem uma casa de 240 m², azul, na rua Vicente machado;
- **Maria** tem uma casa de 70 m², rosa, na rua Getúlio Vargas;
- **Alguém** tem uma casa verde de 160m² na rua Brigadeiro Franco;

```
casa_maria_area = 70  
casa_maria_rua = 'Getúlio Vargas'  
casa_maria_cor = 'Rosa'
```

```
casa_joao_area = 240  
casa_joao_rua = 'Vicente Machado'  
casa_joao_cor = 'Azul'
```

```
casa_verde_area = '160'  
casa_verde_rua = 'Brigadeiro Franco'  
casa_verde_cor = 'Verde'
```

Fazer um script para mostrar o seguinte texto:

- João tem uma casa Azul de 240 m² na rua Vicente Machado;
- Maria tem uma casa Rosa de 70 m² na rua Getúlio Vargas;
- Alguém tem uma casa Verde de 160m² na rua Brigadeiro Franco;

```
print('João tem uma casa', casa_joao_cor, 'de', casa_joao_area, 'm²', 'na rua', casa_joao_rua)  
print('Maria tem uma casa', casa_maria_cor, 'de', casa_maria_area, 'm²', 'na rua', casa_maria_rua)  
print('Alguém tem uma casa', casa_verde_cor, 'de', casa_verde_area, 'm²', 'na rua', casa_joao_rua)
```

Ao replicar (copiar e colar) a linha de código, a chance de ocorrer algum erro de digitação é considerável!

2 Problemas em usar essa metodologia

1. **Tamanho do código** para todas as casas de uma cidade (Curitiba...);
2. **Adicionar outras informações** como: ano de construção, R\$, etc. (imagine 100+);
3. Nesse formato, **número** são '**aceitos**' **como texto**, por exemplo (basta cometer um erro...);
4. Indexar os métodos (ex: mostrar, ligar luzes, cortar energia, etc.) **para cada caso declarado**.

Para cada uma das casas, existe duas novas variáveis:

- Uma delas indica se as luzes estão acesas ou não;
- A outra indica se a energia elétrica está ligada;

A energia elétrica (nesse exemplo) só pode ser desligada se não houver fornecimento de energia na cidade e, portanto, a energia de todas as casas deve ser desligada simultaneamente.

Suponha que o objetivo é fazer um método que desliga as luzes de cada casa individualmente, porém que liga/desliga a energia elétrica de todas as casas simultaneamente.

Qual é a melhor abordagem para solucionar esse problema?

Programação Orientada a Objetos (POO)

3 Conceitos

Classe:

- Estrutura que abstrai características e comportamento de determinado objeto físico ou abstrato.

Instânciar:

- Criar um objeto a partir da classe.



```
class Casa:  
    pass  
  
casa_joao = Casa()  
casa_maria = Casa()  
casa_verde = Casa()
```

4 Métodos e Atributos

Métodos:

- Funções que criam, descrevem ou emulam comportamentos.
- Ex: Ligar ou desligar as luzes, Pintar de outra cor, trancar ou destrancar as portas.

Atributos:

- Características que descrevem um objeto.
- Ex: Cor, Área, Local, Preço.

```
class Casa:
    def ligar_luzes(self):
        print("Luzes ligadas")
    def desligar_luzes(self):
        print("Luzes desligadas")
    def trancar_portas(self):
        print("Portas trancadas")
    def destrancar_portas(self):
        print("Portas destrancadas")

casa_joao = Casa()
casa_maria = Casa()
casa_verde = Casa()

casa_joao.destrancar_portas()
casa_joao.ligar_luzes()
casa_joao.trancar_portas()
```

```
class Casa():
    def __init__(self, area, rua, cor):
        self._area = area
        self._rua = rua
        self._cor = cor
    def __del__(self):
        pass

casa_joao = Casa(240,"Vicente Machado","Azul")
casa_maria = Casa(70,"Getúlio Vargas","Rosa")
casa_verde = Casa('160',"Brigadeiro Franco","Verde")
```

5 Construindo e destruindo objetos

Construtor:

- Função executada ao criar uma instância de determinada classe.

Destruitor:

- Função executada ao destruir um objeto de uma classe.

6 Métodos estáticos

Métodos e atributos estáticos

- São métodos e atributos compartilhados por todos objetos.

```
class Casa():
    def __init__(self, area, rua, cor):
        self._area = area
        self._rua = rua
        self._cor = cor
        self._luzes_ligadas = False
    def ligar_luzes(self):
        self._luzes_ligadas = True

    def desligar_luzes(self):
        self._luzes_ligadas = False
    @staticmethod
    def cortar_energia():
        Casa.energia = False

    @staticmethod
    def ligar_energia():
        Casa.energia = True

Casa.ligar_energia()
casa_joao = Casa(240, "Vicente Machado", "Azul")
casa_maria = Casa(70, "Getúlio Vargas", "Rosa")
casa_verde = Casa(160, "Brigadeiro Franco", "Verde")
casa_joao.ligar_luzes()
```

Método estático

Atributo estático

7

Atributos

```
class Casa():
    def __init__(self, area: int, rua, cor: str, prop='Dom'):
        self.area = area
        self.rua = rua
        self.cor = cor
        self.__prop = prop

    def alterarDono(self, prop):
        self.__prop = prop

casa1 = Casa(120, 'Quinta Avenida', 'Branco')
casa1.alterarDono('Queila')
```

Exercício 1

Objetivo:

- Crie uma classe 'Casa' parecida com os exemplos acima;
- Com atributos de Área, Rua, Cor e Nome;
- Sendo nome opcional na hora de instanciar;
- Possua um método para mostrar todos os atributos;

Tempo estimado: 20 minutos

Exercício 1

```
class Casa():
    def __init__(self, area, rua, cor, nome=None):
        self._area = area
        self._rua = rua
        self._cor = cor
        self._nome = nome

    def mostrar(self):
        if self._nome is None:
            nome = 'Alguém'
        else:
            nome = self._nome
        print("{} tem uma casa {} de {}m² na rua {}".format(nome, self._cor, self._area, self._rua))

casa_joao = Casa(240, "Vicente Machado", "Azul", "João")
casa_maria = Casa(70, "Getúlio Vargas", "Rosa", "Maria")
casa_verde = Casa(160, "Brigadeiro Franco", "Verde")

casa_joao.mostrar()
casa_maria.mostrar()
casa_verde.mostrar()
```

8 Existem 4 pilares da Programação Orientada a Objetos

1 - Abstração: Abstrai características e comportamento de determinado objeto físico ou abstrato.

2 - Encapsulamento: Restringir o acesso de outras classes a determinada informação. Deixando os atributos Públicos ou Privados.

```
class Casa():
    def __init__(self, area: int, rua, cor: str, prop='Dom'):
        self.area = area
        self.rua = rua
        self.cor = cor
        self.__prop = prop

    def alterarDono(self, prop):
        self.__prop = prop

casa1 = Casa(120, 'Quinta Avenida', 'Branco')
casa1.alterarDono('Queila')
```

3 - Herança:

- A herança ocorre quando uma classe criada herda funcionalidades de uma classe base.



```
class InstrumentoEscrita:
    def __init__(self, material):
        self._material = material

    def escrever(self):
        print("Escrevendo...")

    def desenhar(self):
        print("Desenhando...")

    def pintar(self):
        print("Pintando...")

class Lapis(InstrumentoEscrita):
    def __init__(self, grafite_n=0.7):
        super().__init__("grafite")
        self._grafite = grafite_n

class Caneta(InstrumentoEscrita):
    def __init__(self, cor_tinta):
        super().__init__("tinta")
        self._cor = cor_tinta
```

4 - Polimorfismo: consiste em modificar um método herdado na nova classe.

- Isso não altera o método original.
- Ex: Abrir janela no carro é a partir do botão; Abrir janela na casa é manualmente.

Têm a mesma função, mas dependendo da classe o método é diferente.

9 Exercício 2

Objetivo:

- Criar uma classe chamada 'Calculadora';
- Nessa classe, deve possuir funções básicas em forma de métodos;
- Crie uma 'Calculadora científica';
- Essa classe deve herdar a primeira classe, adicionando métodos com cálculos mais avançados;



10 Exercício 3

Objetivo:

- Crie uma estrutura para abrir e controlar as contas de um banco;
- As contas corrente, poupança e salário, devem possuir uma classe 'Conta' como herança;
- Herdando informações e métodos na qual todos os tipos de conta possuam. Ex: Abrir conta;
- Utilizando do encapsulamento. Ex: Valor da conta.