DESAFIO – Classes Anônimas e Lambdas (Versão Python)

1. Classe Produto:

```
class Produto:

    def __init__(self, nome: str, preco: float, em_estoque: bool):
        self.nome = nome
        self.preco = preco
        self.em_estoque = em_estoque

    def __repr__(self):
        return f"Produto(nome='{self.nome}', preco={self.preco},
em_estoque={self.em_estoque})"
```

2. Interface Funcional:

```
from typing import Callable, TypeVar

T = TypeVar('T')
FiltroProduto = Callable[[T], bool]  # equivalente à interface funcional
```

3. Classe de Serviço – ServicoFiltro:

```
class ServicoFiltro:

    @staticmethod
    def filtrar(lista, filtro: FiltroProduto):
        resultados = []
        for item in lista:
            if filtro(item):
                resultados.append(item)
        return resultados
```

4. Implementação e Testes:

```
def main():

    # A. Criando e populando a lista de produtos
```

```
    produtos = [
        Produto("Notebook", 3500.0, True),
        Produto("Mouse", 49.90, True),
        Produto("Teclado", 120.00, False),
        Produto("Monitor", 899.00, True),
        Produto("Smartphone", 2500.00, False),
    ]

    # B. Classe Anônima (simulada com função interna)

def filtro_em_estoque(produto):
        return produto.em_estoque

    em_estoque = ServicoFiltro.filtrar(produtos, filtro_em_estoque)

    print("=== Produtos em estoque (Classe Anônima) ===")
    for p in em_estoque:
        print(p)

    # C. Expressão Lambda -> Preço acima de 1000

produtos_caros = ServicoFiltro.filtrar(produtos, lambda p: p.preco > 1000)

    print("\n=== Produtos com preço > R$ 1000 (Lambda) ===")
    for p in produtos_caros:
        print(p)

    # D. Bônus: fora de estoque OU preço abaixo de 50

condicao_extra = ServicoFiltro.filtrar(
        produtos,
        lambda p: not p.em_estoque or p.preco < 50
    )

    print("\n=== Bônus: fora de estoque OU preço < R$ 50 (Lambda) ===")
    for p in condicao_extra:
        print(p)


if __name__ == "__main__":
    main()
```

Leonardo Coelho Nunes