Exercício Prático: Simulação de um Sistema de E-commerce

Objetivo: Projetar e implementar um sistema simplificado de e-commerce, aplicando os conceitos de classes, objetos, atributos e métodos. Você criará três classes que interagem entre si: Produto, CarrinhoDeCompras e Cliente.

Contexto: O Produto é o item individual à venda. O CarrinhoDeCompras agrupa vários produtos que um cliente deseja comprar. O Cliente é quem possui o carrinho e realiza a compra.

Classe 1: Produto

Esta classe representa um item único no catálogo da loja.

- **Descrição:** Deve conter informações sobre o nome, preço, categoria e a quantidade disponível em estoque.
- **Atributos de Instância:**
    - nome (str)
    - preco (float)
    - categoria (str)
    - estoque (int)
- **Atributo de Classe:**
    - total_produtos_criados (int): Um contador que incrementa a cada novo produto instanciado, similar ao total_musicas da aula.
- **Métodos:**
    - __init__(self, nome, preco, categoria, estoque): Construtor para inicializar o produto.
    - verificar_estoque(self, quantidade_desejada): Retorna True se a quantidade_desejada for menor ou igual ao estoque disponível, e False caso contrário.
    - atualizar_estoque(self, quantidade_vendida): Diminui o valor do estoque com base na quantidade vendida.

Classe 2: CarrinhoDeCompras

Esta classe gerencia uma coleção de produtos que um cliente pretende comprar.

- **Descrição:** Deve permitir adicionar, remover e listar produtos, além de calcular o valor total da compra, assim como a Playlist gerenciava as músicas.
- **Atributos de Instância:**
    - itens (dict): Um dicionário para armazenar os produtos e suas respectivas quantidades. Ex: {<objeto_produto_1>: 2, <objeto_produto_2>: 1}.
    - valor_total (float): O valor total dos itens no carrinho.

- **Métodos:**
    - __init__(self): Inicia um carrinho vazio.
    - adicionar_item(self, produto, quantidade): Adiciona um produto e a quantidade ao carrinho. Antes de adicionar, deve usar o método produto.verificar_estoque() para garantir que a quantidade é válida. Se for, adiciona ao dicionário itens e atualiza o valor_total.
    - remover_item(self, produto): Remove um produto por completo do carrinho e recalcula o valor_total.
    - listar_itens(self): Exibe de forma organizada todos os produtos no carrinho, suas quantidades e o subtotal por item, similar ao método listar_musicas.
    - esvaziar_carrinho(self): Limpa todos os itens do carrinho após uma compra.

Classe 3: Cliente

Esta classe representa o usuário que interage com a loja.

- **Descrição:** Cada cliente terá seus próprios dados e possuirá uma instância única de CarrinhoDeCompras.
- **Atributos de Instância:**
    - nome (str)
    - email (str)
    - carrinho (CarrinhoDeCompras): O carrinho de compras associado ao cliente.
- **Métodos:**
    - __init__(self, nome, email): Ao criar um novo cliente, este método deve também criar uma nova instância de CarrinhoDeCompras e atribuí-la ao self.carrinho.
    - adicionar_produto_ao_carrinho(self, produto, quantidade): Este método não implementa a lógica de adição, mas sim **delega** a chamada para o método correspondente no seu próprio carrinho: self.carrinho.adicionar_item(produto, quantidade).
    - ver_carrinho(self): De forma similar, delega a chamada para self.carrinho.listar_itens().
    - finalizar_compra(self):
        1. Verifica se o carrinho não está vazio.
        2. Exibe o valor total da compra.
        3. Para cada item no carrinho, chama o método produto.atualizar_estoque() para dar baixa no sistema.
        4. Chama o método self.carrinho.esvaziar_carrinho().

5. Exibe uma mensagem de "Compra realizada com sucesso!".

Exemplo de Uso (Cenário de Teste)

Python

```
# 1. Criar alguns produtos
p1 = Produto("Notebook Gamer", 5500.00, "Eletrônicos", 10)
p2 = Produto("Mouse Sem Fio", 120.50, "Acessórios", 30)
p3 = Produto("Teclado Mecânico", 350.00, "Acessórios", 15)

# 2. Criar um cliente (seu carrinho é criado automaticamente)
cliente1 = Cliente("Mariana", "mariana@email.com")

# 3. Cliente adiciona produtos ao seu carrinho
print(f"Estoque inicial de '{p1.nome}': {p1.estoque}")
cliente1.adicionar_produto_ao_carrinho(p1, 1)
cliente1.adicionar_produto_ao_carrinho(p2, 2)

# 4. Cliente visualiza seu carrinho
print("\\n--- Carrinho da Mariana ---")
cliente1.ver_carrinho()

# 5. Cliente finaliza a compra
print("\\n--- Finalizando a Compra ---")
cliente1.finalizar_compra()

# 6. Verificar se o estoque do produto foi atualizado
print(f"\\nEstoque final de '{p1.nome}': {p1.estoque}")

# 7. Verificar se o carrinho está vazio
print("\\n--- Carrinho da Mariana após a compra ---")
cliente1.ver_carrinho()
```