Desenvolvimento e Testes de um Sistema de Gestão de Pedidos Orientado a Objetos

Objetivo

Os alunos devem **desenvolver um sistema orientado a objetos** para gerenciar pedidos de um e-commerce e, em seguida, criar **testes automatizados** para validar seu funcionamento. O objetivo é garantir que a lógica do sistema esteja correta, utilizando **testes de unidade, integração e tratamento de erros**.

Tanto o sistema quanto os testes devem ser feitos preferencialmente em Java ou Python, desde que utilizem **testes automatizados e boas práticas de POO**.

OBS: Se quiser utilizar outra linguagem, justificar bem o motivo
(por exemplo: afinidade e/ou experiência)

Contexto da Atividade

Você foi contratado para desenvolver um **sistema de gestão de pedidos para um e-commerce**. O sistema deve permitir que **clientes realizem pedidos, escolham produtos, apliquem descontos, realizem pagamentos e acompanhem o status da entrega**.

Além disso, o sistema precisa suportar **diferentes métodos de pagamento**, como **cartão de crédito, boleto bancário e Pix**, e deve garantir que **os pedidos sejam processados corretamente**.

Requisitos do Sistema

1. Criar as Classes do Sistema

O sistema deve conter **cinco classes principais**:

- `Produto`: Representa um item disponível para compra.
- `Cliente`: Representa um cliente do e-commerce.
- `Pedido`: Armazena informações sobre pedidos e permite calcular valores.
- `Pagamento`: Processa pagamentos usando diferentes métodos.
- `Entrega`: Gerencia o status de envio dos pedidos.

---

2. Implementar as Funcionalidades

Cada classe deve conter **os seguintes atributos e métodos**:

Classe `Produto`

- `id`: Identificador único do produto.
- `nome`: Nome do produto.
- `preco`: Preço unitário do produto.

Classe `Cliente`

- `id`: Identificador único do cliente.
- `nome`: Nome do cliente.
- `endereco`: Endereço de entrega.
- `historico_pedidos`: Lista de pedidos anteriores.

Métodos:

- `adicionar_pedido(pedido)`: Associa um pedido ao cliente.

Classe `Pedido`

- `id`: Identificador único do pedido.
- `cliente`: Cliente que fez o pedido.
- `itens`: Lista de produtos comprados.
- `status`: Pode ser **"Aguardando Pagamento", "Pago", "Enviado" ou "Entregue"**.
- `valor_total`: Soma dos preços dos produtos.

Métodos:

- `adicionar_produto(produto, quantidade)`: Adiciona itens ao pedido.
- `calcular_total()`: Retorna o valor total do pedido.
- `atualizar_status(novo_status)`: Atualiza o status do pedido.

Classe `Pagamento`

- `pedido`: Pedido associado ao pagamento.
- `metodo`: Pode ser **"Cartão", "Boleto" ou "Pix"**.
- `status_pagamento`: Pode ser **"Aguardando", "Aprovado" ou "Recusado"**.

Métodos:

- `processar_pagamento()`: Simula a aprovação ou rejeição do pagamento.

Classe `Entrega`

- `pedido`: Pedido sendo entregue.
- `status_entrega`: Pode ser **"Aguardando Envio", "Em Transporte" ou "Entregue"**.
- `codigo_rastreamento`: Código único para rastreamento.

Métodos:

- `iniciar_entrega()`: Define o status como **"Em Transporte"**.
- `finalizar_entrega()`: Define o status como **"Entregue"**.

---

3. Criar os Testes Automatizados

Os alunos devem garantir, no mínimo, que os seguintes **cenários sejam testados**:
OBS: Testes adicionais implementados contaram como pontos extras durante a avaliação, aproveitem!

Cenário	Descrição
Criar um cliente e um pedido	O cliente deve ser corretamente vinculado ao pedido.
Adicionar produtos ao pedido	O pedido deve armazenar os produtos corretamente.
Calcular o valor total do pedido	O total deve refletir a soma dos produtos.
Aplicar um pagamento e validar o status	O pedido só pode ser marcado como "Pago" se o pagamento for aprovado.
Simular falha no pagamento	O pedido deve permanecer "Aguardando Pagamento" se o pagamento for recusado.
Iniciar e finalizar entrega	O status de entrega deve ser atualizado corretamente.

Os testes devem utilizar:

- **Testes de unidade** (para métodos individuais).
- **Testes de integração** (para validar comunicação entre classes).

Resumindo:

Criar as Classes: Implementar `Produto`, `Cliente`, `Pedido`, `Pagamento` e `Entrega`.
Desenvolver os Métodos: Criar funcionalidades como adicionar produtos, processar pagamentos e gerenciar entregas.
Criar os Testes Automatizados: Aplicar **testes de unidade e integração**.
Rodar os Testes: Executar os testes e verificar a cobertura do código.
Refatorar (se necessário): Ajustar erros encontrados durante os testes.

Pontos extras: Sejam criativos, implementem mais funções e testes além do orientado neste documento para garantir uma maior nota. Boa sorte!