Sistema de Processamento de Pagamentos

Imagine que você está desenvolvendo o sistema de processamento de pagamentos para uma loja online. A loja permite que os clientes escolham entre várias formas de pagamento, incluindo **cartão de crédito**, **PayPal** e **transferência bancária**. Cada forma de pagamento tem suas próprias regras de processamento e taxas associadas.

Seu objetivo é projetar uma solução flexível que permita adicionar novas formas de pagamento no futuro sem alterar o código existente. O padrão de design **Strategy** será usado para lidar com as diferentes estratégias de processamento de pagamento.

Instruções para o Exercício:
1. Crie uma interface (ou classe abstrata) chamada **FormaPagamentoStrategy** que declare um método chamado **processar_pagamento**. Este método aceita o valor do pedido como argumento e retorna uma mensagem de confirmação do pagamento.

2. Implemente três classes que representam estratégias de processamento de pagamento: **CartaoCreditoStrategy**, **PayPalStrategy** e **TransferenciaBancariaStrategy**. Cada uma dessas classes deve implementar a interface **FormaPagamentoStrategy** e fornecer sua própria lógica de processamento de pagamento.

Instruções para o Exercício:

1. Crie uma interface (ou classe abstrata) chamada **FormaPagamentoStrategy** que declare um método chamado **processar_pagamento**. Este método aceita o valor do pedido como argumento e retorna uma mensagem de confirmação do pagamento.

2. Implemente três classes que representam estratégias de processamento de pagamento: **CartaoCreditoStrategy**, **PayPalStrategy** e **TransferenciaBancariaStrategy**. Cada uma dessas classes deve implementar a interface **FormaPagamentoStrategy** e fornecer sua própria lógica de processamento de pagamento.