Perguntas para Reflexão:

1. O que acontece se você adicionar um novo método de pagamento sem modificar a função processar?

Adicionar um novo método de pagamento sem modificar a função processar, o sistema não saberá como lidar com esse novo método. Isso significa que, quando tentar usar o novo tipo de pagamento, ele não funcionará corretamente, porque a função processar não está preparada para reconhecê-lo. Para que tudo funcione, precisa atualizar a função processar para incluir a lógica do novo método de pagamento.

2. Como o polimorfismo ajuda a manter o código flexível e extensível?

O polimorfismo permite que diferentes classes implementem métodos com o mesmo nome de maneiras diferentes. Isso significa que você pode adicionar novos métodos de pagamento sem alterar o código existente, desde que cada nova classe de pagamento implemente o método processar_pagamento. Assim, o código se torna mais flexível e fácil de expandir.

3. Qual é a diferença entre a função processar e os métodos processar_pagamento nas subclasses?

A função processar na classe Pagamento é um método genérico que serve para verificar se o polimorfismo está funcionando corretamente . E o método processar_pagamento nas subclasses (PagamentoBoleto, PagamentoCartaoCredito, PagamentoPix) são implementações específicas que definem como cada tipo de pagamento deve ser processado.

4. Como você pode garantir que todos os métodos de pagamento implementem o método processar_pagamento corretamente?

Para garantir que todos os métodos de pagamento implementem o método processar_pagamento corretamente, você pode transformar a classe Pagamento em uma classe abstrata (usando o módulo abc do Python) e definir processar_pagamento como um método abstrato. Isso forçará todas as subclasses a implementar esse método, garantindo que cada tipo de pagamento tenha sua própria lógica de processamento.