UNIVERSIDADE DE FORTALEZA - UNIFOR
Disciplina: [T200] Projeto Arquitetura de Sistemas
Prof. Doutorando Nathalino Pachêco

Princípio da Substituição de Liskov (LSP)

Contexto

Você foi contratado em um estágio e a primeira atividade é desenvolver um sistema de transações bancárias que deve processar pagamentos para diferentes tipos de contas: "*ContaCorrente*" e "*ContaPoupanca*". O sistema deve garantir que todas as contas possam processar pagamentos de maneira segura e correta.

Objetivo

Neste exercício, você aplicará o Princípio da Substituição de Liskov (LSP) para garantir que a classe base Conta e suas subclasses "*ContaCorrente*" e "*ContaPoupanca*" possam ser usadas de forma intercambiável sem causar comportamentos incorretos.

Instruções

1. Definir a Classe Base **`Conta`**:
    a. Crie uma classe base (Super (Pai) Classe) `Conta` que contenha um método abstrato: `processarPagamento(double valor)`.
2. Criar Subclasses **`ContaCorrente`** e **`ContaPoupanca`**, regra de negócio:
    a. **`ContaCorrente`** deve permitir qualquer valor de pagamento; e
    b. **`ContaPoupanca`** deve lançar uma exceção se o valor do pagamento for superior a R$ 1000,00.
3. Definir a Interface **`ProcessadorPagamento`**:
    a. Que contenha o método `processarPagamento(double valor)`. Isso garantirá que todas as classes que implementam essa interface possam processar pagamentos, respeitando suas próprias regras de negócio.
    b.
4. Criar uma Classe de Serviço **`ProcessadorDePagamentos`**:

    a. Crie uma classe `ProcessadorDePagamentos` que tenha um método `processar` que aceite um `ProcessadorPagamento` e um valor de pagamento.
5. Testar o Sistema:
    a. Implemente um método `main` que teste o processamento de pagamentos para instâncias de `ContaCorrente` e `ContaPoupanca`, demonstrando a conformidade com o LSP.

Observações/Avisos:

Certifique-se de compreender claramente o Princípio da Substituição de Liskov (LSP) antes de iniciar a atividade, pois ele é essencial para o sucesso do exercício.

Além disso, sinta-se à vontade para modificar as instruções, desde que o projeto mantenha uma alta coesão.