NOME: Matheus Dantas Carvalho RM: 558804

NOME: Rafael Panhoca Batista do Nascimento RM: 555014

NOME: Silas Alves dos Santos RM: 555020

NOME: Victor Hugo Xavier RM: 559094

NOME: Guilherme Oliveira Da Silva RM: 558797

1. **Qual a diferença entre exceções de regra de negócio e exceções técnicas? Onde devem ser tratadas?**

Exceções de regra de negócio: acontecem quando alguma regra definida pelo sistema é violada, como tentar solicitar um cartão com bandeira inválida, fazer uma compra acima do limite disponível ou cadastrar um cliente já existente. Essas exceções devem ser tratadas na camada de negócio (BO) e podem ser repassadas para a camada de apresentação, para que o usuário receba uma mensagem clara sobre o que aconteceu.

Exceções técnicas: acontecem por problemas de infraestrutura, como erro na conexão com o banco, falha ao carregar o driver JDBC ou erro em uma consulta SQL. Essas devem ser tratadas nas camadas mais baixas, como o DAO ou a Factory, e geralmente ficam registradas em logs para facilitar a manutenção.

2. **Explique as responsabilidades de Model, View e Controller no seu projeto. Dê um exemplo de fluxo.**

Model: É a camada que representa os dados e contém toda a lógica de negócio. No projeto, ela é composta por duas partes:

Classes de Entidade (pacote model): Cliente, Cartao e Transacao, que apenas armazenam os dados.

Business Objects (pacote bo): CartaoBO e TransacaoBO, que contêm as regras de negócio, como validar limites e aprovar cartões. A camada DAO, que acessa o banco, também faz parte do Model.

View: É a camada de apresentação, responsável por exibir a interface para o usuário e capturar suas interações.

Controller: É o intermediário que conecta a View e o Model. Ele recebe as solicitações da View, aciona a lógica de negócio no Model e devolve os resultados para a View exibir. As classes no pacote controller (ClienteController, CartaoController, etc.) cumprem esse papel.

Exemplo de Fluxo: "Registrar Compra"

View (App.java): O usuário escolhe a opção "3. Registrar compra", digita o número do cartão, o valor e a descrição. A View chama o método registrarCompra() no TransacaoController.

Controller (TransacaoController): O método registrarCompra() recebe os dados da View e, sem processá-los, os repassa para a camada de negócio, chamando o método registrarCompra() do TransacaoBO.

Model (TransacaoBO): Aqui, as regras de negócio são aplicadas. O BO usa o CartaoDAO para verificar se o cartão existe. Valida se o valor da compra é positivo e se o limite disponível é suficiente.

Se tudo estiver correto, ele cria um objeto Transacao e usa o TransacaoDAO para salvá-lo no banco e o CartaoDAO para atualizar o limite.

Retorno: A execução volta para o Controller e, em seguida, para a View, que exibe a mensagem "Compra registrada!" ao usuário.

3. **Mostre um exemplo de erro vindo do DAO até a View. O que acontece se não tratar corretamente?**

Um exemplo clássico de erro acontece quando o banco de dados está offline. Nesse caso, a camada DAO, ao tentar inserir um cliente, chama o método Conexao.getConexao(), que lança uma SQLException. Como essa exceção não é tratada no DAO, ela sobe para o Controller, que também não a trata e a repassa para a View. Na classe App.java, o bloco try-catch captura a exceção e exibe no console uma mensagem como “ERRO INESPERADO: Communications link failure”. Caso esse tratamento não existisse, a exceção chegaria até a JVM, resultando no encerramento abrupto da aplicação, com a impressão de um stack trace técnico difícil de entender para o usuário final. Isso não só interromperia o funcionamento do sistema, mas também prejudicaria a experiência do usuário, que não teria um retorno claro sobre o problema ocorrido.