Exercício 8

Questão 1) 1 - Blocos try-catch-finally: blocos try, catch e finally para capturar exceções (erros) que podem ocorrer durante a execução do código. Isso é útil para envolver o código suscetível a erros em um bloco try e capturar e tratar esses erros no bloco catch.

2 - Declaração throw e tipos de erro personalizados**:** erro personalizados estendendo a classe Error ou utilizando tipos de erro existentes. Você pode lançar esses erros usando a declaração throw para indicar condições de erro específicas e capturá-los usando blocos try-catch.

3 - Uso de Promises com .catch(): Ao lidar com código assíncrono em TypeScript usando Promises, é comum utilizar o método .catch() para lidar com erros que podem ocorrer durante a execução de operações assíncronas.

Obs: exemplos nos códigos.

Questão 2) 1) Limitações do try-catch: O uso excessivo de blocos try-catch pode afetar o desempenho do código, pois a execução dentro do bloco try é mais lenta do que o código normal. Os blocos try-catch são úteis para lidar com erros síncronos, mas não podem tratar diretamente erros em código assíncrono.

2) Limitações de erro personalizados: Criar e gerenciar tipos de erro personalizados pode adicionar complexidade ao código, especialmente em pequenos projetos ou situações onde o controle preciso de tipos de erro não é crucial. Falta de documentação ou consistência pode dificultar a manutenção do código.

3)Limitações do Promises: O método .catch() em Promises lida apenas com erros que ocorrem durante a execução da própria Promise. Erros lançados fora do escopo da Promise (por exemplo, erros de sintaxe ou de execução fora de Promises) não são capturados por este método.

Questão 3) Nos códigos.

Questão 4) O método transferir na classe Conta tenta sacar o valor da conta de origem e depositá-lo na conta de destino. Se a conta de origem não tiver saldo suficiente, ele lançará um SaldoNegativoError. No teste, a contaOrigem possui um saldo de R$100 e tentamos transferir R$200 para a contaDestino. Isso resultará em um saldo insuficiente na conta de origem, desencadeando o erro de SaldoNegativoError no bloco catch. No console, a mensagem de erro será exibida indicando que houve um erro ao transferir devido ao saldo insuficiente na conta de origem. O saldo das contas antes e após a tentativa de transferência será exibido para verificar que a transferência não ocorreu devido ao erro.

Questão 5) A exceção lançada no método Conta.sacar() durante a transferência foi capturada dentro do try-catch no método Banco.transferir(). Essa exceção não foi propagada para os métodos conta.transferir() ou banco.transferir() diretamente. A propagação de exceções refere-se à passagem de uma exceção através de várias camadas de chamadas de métodos sem ser capturada, até que seja tratada ou capturada em algum nível mais alto. A confiabilidade dessa implementação depende do contexto e dos requisitos do sistema. A captura da exceção dentro do **Banco.transferir()** oferece um controle centralizado sobre o tratamento de exceções relacionadas às operações de transferência entre contas. Centralização do tratamento de exceções: Isso pode facilitar a manutenção e o gerenciamento de exceções relacionadas a transferências bancárias, tornando mais fácil implementar e alterar a lógica de tratamento de exceções. Ausência de propagação: A não propagação da exceção para níveis mais altos pode

limitar a capacidade de lidar com a exceção em camadas superiores da aplicação, caso seja necessário um tratamento ou registro mais específico.

Obs: resto das questões nos códigos.