Case Study : Custom Exception , Auto Closeable , OOPS , Try with Resources

Introduction

Exception handling is a critical aspect of enterprise applications, especially in domains like banking, where errors can lead to financial loss, fraud, or compliance violations.

This case study demonstrates:

- Designing a custom exception hierarchy.
- Implementing business processes with robust error handling.
- Using AutoCloseable with try-with-resources for safe cleanup.
- Applying object-oriented principles (has-a relationship).

Background

ABCBank, a mid-sized retail bank, recently faced challenges in its ATM and online transaction systems:

- Customers attempting withdrawals beyond available balance caused system crashes.
- Multiple cases of exceeding daily withdrawal limits went unnoticed.
- Blocked accounts were still being processed, leading to compliance issues.
- Potential fraudulent high-value transactions were not being detected early.

To address these, the bank decided to redesign the withdrawal module with:

1. A clear exception-handling strategy.
2. Strong separation of business rules from account state.
3. Reliable resource cleanup (audit logs, session closing , etc).

Requirement

ABCBank mandated that the new withdrawal process must:

- Reject transactions if balance is insufficient.
- Reject transactions if daily withdrawal limit is exceeded.
- Reject all operations if account is blocked.
- Detect and escalate fraudulent activity (unusually high withdrawals , wrong pin etc).
- Use a professional exception hierarchy for better debugging and reporting.

Proposed Solution

The solution team suggested the following entities and processes:

a) Entities

1. Account
    - Fields: accountNumber, balance, dailyLimit, blocked status.
    - Represents customer account state.
2. BankingProcess
    - Has-a relationship with Account.
    - Implements withdrawal process, business validation, and fraud checks.
    - Implements AutoCloseable to ensure safe cleanup (logs, sessions , other required things).

b) Exception Hierarchy

- BankingException → Base exception for all business errors.
    - InsufficientFundsException → Triggered when balance < withdrawal.
    - DailyLimitExceededException → Triggered when withdrawal > dailyLimit.
    - AccountBlockedException → Triggered when account is flagged.
- FraudDetectionException → Unchecked exception for suspected fraud

```java

class Account
{
    private int accountNumber;
    private int balance;
    private int dailyLimit;
    private boolean isBlocked;


}

class BankingProcess implements AutoCloseable
{
    private Account account;

    public void withdraw(int amount) {  }
    @Override
    public void close() {}


}
```