Bank Account Command Pattern

Business Scenario

You are developing a banking application that needs to: 1. Create new accounts with initial deposits 2. Process deposits and withdrawals 3. Provide current account balances 4. Display transaction history

Your task is to implement this system using the Command pattern to separate read and write operations.

Level 1: Basic Implementation

Requirements

Create a simple console application that: - Implements a `BankAccount` domain model with basic operations - Uses separate command objects for account creation, deposits, and withdrawals - Implements query methods for retrieving account details and transaction history - Uses in-memory storage (dictionaries or lists) for both the write and read models - Provides a basic user interface to demonstrate the functionality

Level 2: Intermediate Implementation

Requirements

Extend the basic implementation to: - Implement a proper event system that publishes events when commands are processed - Create separate read models that are updated based on events - Add more sophisticated validation logic for commands - Implement proper repositories for both write and read models - Add support for additional operations (transfer between accounts, interest calculation)

Level 3: Advanced Implementation

Requirements

Transform the application into a complete CQRS system with: - Event sourcing for the write model - Asynchronous event handling with message queues - Multiple specialized read models for different query scenarios - Eventual consistency between write and read models - Unit and integration tests for all components - Proper dependency injection - Exception handling and error reporting