■ Bank Management System - Project Guide

Objective: Build a console-based Bank Management System using Core Java, OOP, and Data Structures.

Goal: To apply Encapsulation, Inheritance, Polymorphism, Collections, and DSA in a real-world mini project.

1. Project Architecture

Recommended folder structure:

bank/ ■■■ models/ ■ ■■■ Account.java ■ ■■■ SavingsAccount.java ■ ■■■ CurrentAccount.java ■ ■■■ Customer.java ■ ■■■ Transaction.java ■■■ services/ ■ ■■■ BankService.java ■ ■■■ AccountService.java ■ ■■■ TransactionService.java ■■■ utils/ ■ ■■■ ValidationUtils.java ■■■ main/ ■■■ BankApp.java

2. Core Classes and Responsibilities

1. **Customer:** Holds customer info (id, name, email, list of accounts).
2. **Account (abstract):** Base for Savings and Current accounts; contains balance, owner, and transactions.
3. **SavingsAccount:** Extends Account; adds interest feature.
4. **CurrentAccount:** Extends Account; supports overdraft.
5. **Transaction:** Records transaction type, amount, and date.

3. Data Structures Used

1. HashMap – Store and access customers efficiently.
2. HashMap – Manage accounts by account number.
3. LinkedList – Maintain ordered transaction history.
4. PriorityQueue – Rank customers by balance for reports.
5. Queue – Track inactive/deactivated accounts.

4. Core Features to Implement

1. Customer Management: Add, view, search, and delete customers.
2. Account Operations: Create account, deposit, withdraw, transfer funds.
3. Transaction Management: Maintain history, filter by type/date.
4. Reports: Show top customers, total balance, and transaction count.

5. Java Concepts Practiced

1. Encapsulation – Private fields with getters/setters.
2. Inheritance – SavingsAccount and CurrentAccount inherit from Account.

3. Polymorphism – Different implementations of deposit/withdraw.
4. Collections – Use of HashMap, ArrayList, LinkedList.
5. Exception Handling – Custom exceptions for invalid transactions.
6. File I/O – Save and load account data (optional).
7. Comparator/Comparable – Sorting customers by balance.

6. Console Menu Example

=== BANK MANAGEMENT SYSTEM === 1. Add Customer 2. Open New Account 3. Deposit Money 4. Withdraw Money 5. Transfer Money 6. View Account Details 7. View Transaction History 8. Show Bank Reports 9. Exit Enter choice: _

7. Step-by-Step Implementation Roadmap

1. **Day 1:** Create models (Customer, Account, Transaction). Implement encapsulation and constructors.
2. **Day 2:** Implement SavingsAccount, CurrentAccount, and BankService with HashMaps.
3. **Day 3:** Add deposit, withdraw, and transaction logic. Test using sample customers.
4. **Day 4:** Add transfer, interest, and reporting features.
5. **Day 5:** Build console menu and integrate services.
6. **Bonus:** Add custom exceptions, file storage, and multithreading.

8. Optional Enhancements

1. File I/O to save and load bank data.
2. Multithreading to simulate concurrent transactions.
3. Validation utilities for input handling.
4. Custom exceptions like InsufficientFundsException.
5. Unit testing using JUnit.

■ **End Goal:** A fully functional console-based banking app demonstrating mastery of Java OOP and DSA.