Banking Management System Documentation

Table of Contents

Introduction

The Banking Management System is a comprehensive Java application designed to simulate banking operations. It provides functionalities for user registration and authentication, account management, transaction processing, loan management, and installment tracking. The system uses JDBC to interact with a MySQL database for persistent data storage.

System Architecture

The system follows a modular design with distinct classes responsible for specific functionalities:

- **Banking_System**: Entry point and main controller
- **Account_Holder**: User authentication and registration
- **Account_Repository**: Bank account creation and management
- **Transaction_Manager**: Financial transaction processing
- **Loan**: Data model for loan information
- **Loan_Manager**: Loan application and approval
- **Loan_Calculator**: Utility for EMI calculations
- **Installment_Manager**: Loan repayment tracking

Class Documentation

Banking_System

Description: Main class that serves as the entry point for the application. It initializes the database connection and manages the main menu and user interface flow.

Attributes:

- `url`: Database connection URL
- `username`: Database username
- `password`: Database password

Methods:

- `main(String[] args)`: Application entry point that establishes database connection and presents the main menu
- `handleLoanManagement(...)`: Sub-menu handler for loan-related operations

Features:

- User registration and login
- Account creation and management
- Money transactions (debit, credit, transfer)
- Balance inquiry
- Loan management
- Resource management with try-with-resources for proper connection handling

Account_Holder

Description: Manages user authentication, registration, and verification.

Attributes:

- `connection`: Database connection object
- `scanner`: Input scanner for user interaction

Methods:

- `register()`: Registers a new user with full name, email, and password
- `login()`: Authenticates users with email and password
- `user_exist(String email)`: Checks if a user with the given email already exists

Features:

- User registration with duplicate email checking
- Secure login process
- Database interaction for user persistence

Account_Repository

Description: Manages bank account creation and retrieval operations.

Attributes:

- `connection`: Database connection object
- `scanner`: Input scanner for user interaction

Methods:

- `open_account(String email)`: Creates a new bank account for a user
- `getAccount_number(String email)`: Retrieves account number associated with an email
- `generateAccountNumber()`: Generates a unique account number
- `account_exist(String email)`: Checks if a user already has a bank account

Features:

- Account creation with unique account number generation
- Initial deposit handling
- Security PIN setup
- Account verification

Transaction_Manager

Description: Handles financial transactions including deposits, withdrawals, and transfers between accounts.

Attributes:

- `connection`: Database connection object
- `scanner`: Input scanner for user interaction

Methods:

- `credit_money(long account_number)`: Deposits money into an account
- `debit_money(long account_number)`: Withdraws money from an account
- `transfer_money(long sender_account_number)`: Transfers money between accounts
- `getBalance(long account_number)`: Retrieves the current balance of an account

Features:

- Transaction security with PIN verification
- Transaction integrity with database transaction support (commit/rollback)
- Balance validation before withdrawals or transfers
- Proper error handling and user feedback

Loan

Description: Data model class that represents a loan entity.

Attributes:

- `loanId`: Unique identifier for the loan
- `accountNumber`: Associated bank account number
- `principalAmount`: Amount borrowed
- `interestRate`: Annual interest rate
- `loanTenure`: Duration of the loan in months
- `loanType`: Type of loan (e.g., Student, Property, Auto)
- `collateral`: Details of any collateral provided
- `status`: Current status of the loan (Pending, Approved, Rejected, Closed)

Methods:

- Getters and setters for all attributes
- `displayLoanDetails()`: Prints formatted loan information

Features:

- Complete representation of loan data
- Encapsulation of loan properties
- Formatted display of loan information

Loan_Manager

Description: Manages loan applications, approvals, and retrieval of loan details.

Attributes:

- `connection`: Database connection object
- `scanner`: Input scanner for user interaction

Methods:

- `applyForLoan(long accountNumber)`: Creates a new loan application
- `getInterestRateForLoanType(String loanType)`: Determines interest rate based on loan type
- `generateLoanId()`: Creates a unique loan identifier
- `calculateTotalInterest(Loan loan)`: Calculates the total interest payable
- `approveLoan(long loanId)`: Updates a loan's status to "Approved"
- `viewLoanDetails(long accountNumber)`: Displays all loans associated with an account

Features:

- Loan application processing
- Dynamic interest rate assignment based on loan type
- Loan approval workflow
- Comprehensive loan details retrieval

Loan_Calculator

Description: Utility class that provides financial calculations for loans.

Methods:

- `calculateEMI(double principal, double annualInterestRate, int tenureMonths)`: Calculates the Equated Monthly Installment
- `calculateTotalPayable(double principal, double annualInterestRate, int tenureMonths)`: Calculates the total amount payable over the loan term

Features:

- Industry-standard EMI calculation formula
- Total repayment calculation
- Static utility methods for easy access

Installment_Manager

Description: Manages loan repayments and tracks installment history.

Attributes:

- `connection`: Database connection object
- `scanner`: Input scanner for user interaction

Methods:

- `recordInstallmentPayment(long loanId)`: Records a payment toward a loan
- `viewInstallmentSummary(long loanId)`: Displays payment summary for a loan

Features:

- Installment payment recording
- Calculation of total amount repaid
- Remaining balance tracking
- Comprehensive payment summary

Database Schema

The system uses the following tables (implied from the code):

1. **User**:
   - Primary key: email
   - Fields: full_name, email, password
2. **Accounts**:
   - Primary key: account_number
   - Fields: account_number, full_name, email, balance, security_pin
3. **Loans**:
   - Primary key: loanId
   - Fields: loanId, accountNumber, principalAmount, interestRate, loanTenure, loanType, collateral, status
4. **LoanInstallments**:
   - Fields: loanId, installmentAmount (with timestamp)

User Workflows

User Registration and Account Creation

1. User registers with name, email, and password
2. User logs in with credentials
3. If no account exists, user creates one with initial deposit and PIN

Financial Transactions

1. User logs in and accesses main menu
2. User selects transaction type (debit, credit, transfer)
3. System verifies security PIN and balance (where applicable)

4. Transaction is processed with appropriate feedback

Loan Management

1. User applies for a loan, providing amount, tenure, type, and collateral
2. System assigns interest rate based on loan type
3. Admin approves the loan (status change)
4. User can view loan details and calculate EMI

Installment Payments

1. User makes installment payments toward their loan
2. System records the payment and updates the repayment summary
3. User can view payment history and remaining balance