PROJECT 1: BANKING SYSTEM

Advanced Requirements

Endino, Michaela RJ
Sanchez, Jan Eduard
Tabianan, Ken
Velasco, Shauntie

March 2025

Account Class

An abstract account class that has comparators to compare itself with different account objects.

Attributes

Data Type and Name	Description
String bankID	The unique identifier of the bank associated with this account.
String accountNumber	The account number of this account object. Cannot be modified once set.
String ownerFName	First name of the account owner.
String ownerLName	Last name of the account owner.
String ownerEmail	Email address of the account owner.
String pin	The PIN used for account security.
HashMap<String, Transaction.Transactions> Transactions	A map storing the transactions associated with this account.

Methods

Method Name	Description
String getAccountNumber()	Returns the account number of this account.
String getOwnerFName()	Returns the first name of the account owner.
String getOwnerLName()	Returns the last name of the account owner.
String getOwnerEmail()	Returns the email address of the account owner.
String getOwnerFullName()	Returns the full name of the account owner.
String getBankID()	Returns the bank ID associated with this account.
String getPin()	Returns the PIN of the account.
String toString()	Returns a formatted string representation of the account (e.g., accountNumber - full name).
String csvString()	Returns a CSV-formatted string representation of the account.

Credit Account

Attributes

Data Type and Name	Description
double loan	The loan amount associated with this credit account.

Methods

Method Name	Description
void setLoan(double amount)	Sets the loan amount for this credit account.
double getLoan()	Returns the loan amount associated with this credit account.
void init()	Initializes the credit account by calling CreditAccountLauncher.creditAccountInit().

Savings Account

Attributes

Data Type and Name	Description
double balance	The balance amount associated with this savings account.

Methods

Method Name	Description
void setBalance(double amount)	Sets the balance for this savings account.
double getBalance()	Returns the current balance of the savings account.
void init()	Initializes the savings account (currently commented out).

Bank

Attributes

String ID	Unique identifier for the bank.
String name	Name of the bank.
String passcode	Security passcode for the bank.
double depositLimit	Maximum amount a savings account can deposit per transaction (default: 50,000.0).
double withdrawLimit	Maximum amount withdrawable or transferable per transaction (default: 50,000.0).
double creditLimit	Maximum total loan that all credit accounts in the bank can handle (default: 100,000.0).
double processingFee	Additional fee applied to inter-bank transactions (default: 10.00).
HashMap<String, Account> BANKACCOUNTS	Collection of accounts registered to this bank.
String ID	Unique identifier for the bank.

Methods

Method Name	**Description**
String getName()	Returns the name of the bank.
String getID()	Returns the bank ID.
String getPasscode()	Returns the bank passcode.
double getDepositLimit()	Returns the deposit limit for savings accounts.
double getWithdrawLimit()	Returns the withdrawal/transfer limit for savings accounts.
double getCreditLimit()	Returns the total credit limit for the bank's credit accounts.
double getProcessingFee()	Returns the inter-bank transaction fee.
String toString()	Returns a formatted string representing the bank (ID - Name).
String csvString()	Returns a CSV-compatible string representation of the bank.

Account Launcher

A class primarily used for interacting with the account module.

Attributes

Data Type and Name	Description
protected static AccountLoginService logSession	Manages account login sessions.

Methods

Method Name	Description
public static AccountLoginService getLogSession()	Returns the current login session instance.
public static void accountInit()	Initializes an account if a user is logged in; otherwise, notifies that no account is logged in.
public static void accountLogin()	Handles the account login process, requiring a user to first select a bank and then enter login credentials.

Bank Launcher

A class primarily used for interacting with the bank module.

Attributes

Data Type and Name	Description
private static BankLoginService logSession	Manages the bank login session. Ensures only logged-in users can interact with the bank.

Methods

Method Name	Description
public static BankLoginService getLogSession()	Retrieves the current bank login session.
public static void bankInit()	Initializes the bank interaction interface.
Functionality:	

CreditAccountLauncher Class

Attributes

Data Type and Name	***Description***
public static CreditAccount getLoggedAccount()	Retrieves the currently logged-in credit account. Returns null if no credit account is logged in.

Methods

Method Name	**Description**
public static void creditAccountInit()	Initializes the **Credit Account Menu** interface. **Functionality:** - Checks if a **credit account** is logged in. - Displays the **Credit Account Menu** with the following options: 1. View loan balance. 2. Apply for a loan. 3. Make a recompense payment. 4. View transaction logs. 5. Logout.
public static void loan(CreditAccount account)	Handles loan payment transactions. **Functionality:** - Prompts the user to enter a **loan amount** and **recipient's account number**. - Validates if the recipient is a **BalanceHolder** (must be a savings account). - Processes the payment using PaymentService.
public static void recompense(CreditAccount account)	Processes recompense transactions. **Functionality:** - Prompts the user to enter an **amount** to recompense. - Calls RecompenseService to process the recompense. - Displays a confirmation message upon success.

Savings Account Launcher

Attributes

Data Type and Name	**Description**
public static SavingsAccount getLoggedAccount()	Retrieves the currently logged-in savings account. Returns null if no savings account is logged in.

Methods

Method Name	Description
public static void savingsAccountInit()	Initializes the **Savings Account Menu** interface. **Functionality:** - Checks if a **savings account** is logged in. - Displays the **Savings Account Menu** with the following options: 1. View balance. 2. Deposit money. 3. Withdraw money. 4. Transfer funds. 5. View transaction logs. 6. Logout.
public static void deposit(SavingsAccount account)	Handles **deposit transactions**. **Functionality:** - Prompts the user to enter a deposit **amount**. - Calls DepositService to process the deposit. - Displays a confirmation message.
public static void withdraw(SavingsAccount account)	Handles **withdrawal transactions**. **Functionality:** - Prompts the user to enter a withdrawal **amount**. - Calls WithdrawService to process the withdrawal. - Displays a confirmation message.
public static void transfer(SavingsAccount account)	Handles **fund transfers**. **Functionality:** - Displays a list of available banks. - Prompts the user to select a **Bank ID**. - If the recipient is in the same bank, prompts for a **transfer amount** and **recipient's account ID**. - Calls TransferService to complete the transaction. - If transferring to another bank, verifies the bank and completes the transfer accordingly.

AccountCreationService

Attribute

Data Type and Name	Description
ArrayList<Field<String, ?>> fields	Stores user-provided basic information for account creation.

Methods

Method Name	Description
private static ArrayList<Field<String, ?>> createNewAccount()	Gathers basic account information (first name, last name, email, and PIN) and returns an array list of field objects.
public static void createNewCreditAccount(Bank bank)	Creates a new CreditAccount by collecting user details and generating a unique account number.
public static void createNewSavingsAccount(Bank bank)	Creates a new SavingsAccount by collecting user details and an initial balance, then generating a unique account number.
private static void addNewAccount(Bank bank, Account account)	Adds the created account to the bank if it does not already exist.

Account Login Service

Attributes

Data Type and Name	Description
private Account loggedAccount	Stores the currently logged-in account.
private Bank assocBank	Stores the bank associated with the logged-in account.

Methods

Method Name	Description
public Account getLoggedAccount()	Returns the currently logged-in account.
public Bank getAssocBank()	Returns the bank associated with the logged-in account.
public boolean isLoggedIn()	Checks if a user is currently logged in.
public static Account checkCredentials(String accountNum, String passcode)	Verifies the inputted credentials and returns an Account object if valid; otherwise, returns null.
public void setLogSession(Account account)	Establishes a login session for the given account by setting the logged account and associated bank.

public void destroyLogSession()	Ends the current login session by resetting the logged account and associated bank.

BankCreationService

Attributes

Data Type and Name	Description
Field<String, String> bankName	Stores the name of the bank entered by the user.
Field<String, Integer> bankPasscode	Stores the 8-digit passcode for the bank.
String bankID	A randomly generated unique ID for the new bank.
Field<Double, Double> depositLimit	Deposit limit for the bank (user-defined).
Field<Double, Double> withdrawLimit	Withdrawal limit for the bank (user-defined).
Field<Double, Double> creditLimit	Credit limit for the bank (user-defined).
Field<Double, Double> processingFee	Processing fee for bank transactions (user-defined).

Methods

Method Name	Description
createNewBank()	Creates a new bank record. **Steps:** 1. Prompts the user to enter a bank name. 2. Generates a unique bank ID. 3. Requests an 8-digit bank passcode. 4. Asks the user if they want to set custom deposit, withdrawal, credit limits, and processing fees. 5. If yes, prompts the user for limits and fees before saving the bank record. 6. If no, saves the bank record with default limits. **Throws:** NumberFormatException – If an invalid number format is entered for deposit limit, withdraw limit, credit limit, or processing fee.

BankDisplayerService

Attributes

Data Type and Name	Description
BankDBManager bankDBManager	Manages database operations related to banks and accounts.
Main main	Handles user interactions and menu displays.
AccountCreationService accountCreationService	Manages the creation of new accounts.

Methods

Method Name	Description
showBanks()	Displays a menu of all registered banks in the system. **Functionality:** - Retrieves all banks from the database. - Displays the bank ID and name. - Shows a message if no banks are registered.
showAccounts(String bankID)	Displays accounts registered under a specific bank. **Functionality:** - Prompts the user to select the type of accounts to display: (1) Credit Accounts, (2) Savings Accounts, or (3) All Accounts. - Retrieves and displays the selected account type from the database. - Shows a message if no accounts are registered.
createAccounts(Bank bank)	Facilitates the creation of new accounts for the currently logged-in bank. **Functionality:** - Prompts the user to choose between creating a Credit or Savings account. - Calls AccountCreationService to create the selected account type. - Displays a confirmation message upon successful creation.

Bank Login Service

Attributes

Data Type and Name	Description
Bank loggedBank	Stores the currently logged-in bank instance. null if no bank is logged in.

Methods

Method Name	Description
getLoggedBank()	Retrieves the currently logged-in bank.
isLogged()	Checks if a bank account is currently logged in. **Returns:** true if a bank is logged in, otherwise false.
setLogSession(Bank bank)	Creates a new login session for a bank. **Functionality:** - Assigns the given bank instance to

	loggedBank if no bank is currently logged in. - Displays a success message upon login.
logout()	Destroys the current login session. **Functionality:** - Prints a logout message with the bank's name. - Sets loggedBank to null.

Balance Manager

Methods

Method Name	Description
hasEnoughBalance(BalanceHolder account, double amount)	Checks if the account has enough balance to proceed with a transaction. **Returns:** true if the account balance is sufficient, otherwise false.
insufficientBalance()	Displays a warning message when an account has insufficient funds for a transaction.
adjustAccountBalance(BalanceHolder account, double amount)	Adjusts the account balance by the specified amount. **Functionality:** - Adds or subtracts the given amount from the account balance. - Ensures that the balance does not drop below 0.0.

IDGenerator

Methods

Method Name	Description
bankIDGenerator(String name)	Generates a unique Bank ID using the format **YYYYMMX####**, where: - **YYYY** = Current year - **MM** = Current month - **X** = First letter of the bank name - **####** = Random four-digit number **Ensures uniqueness** by checking against existing IDs in BankDBManager.
accountIDGenerator(Bank bank)	Generates a unique Account ID by appending a random 6-digit number to the last 4 characters of the Bank ID. **Ensures uniqueness** by checking against AccountDBManager.
transactionIDGenerator(String accountNumber)	Generates a unique Transaction ID using the last 6 digits of the account number and a random 6-digit number.

Loan Manager

Method

Method Name	Description
adjustLoanAmount(LoanHolder account, double amountAdjustment)	Adjusts the loan amount for a given account. - Prevents the loan amount from becoming negative. - If the adjustment would result in a negative loan, prints a warning message.

AccountDBManager

Method

Method Name	Description
Table Creation	
createTable()	Creates the accounts, savings_accounts, and credit_accounts tables if they do not exist.
createSavingsAccTable()	Creates the savings_accounts table if it does not exist.
createCreditAccTable()	Creates the credit_accounts table if it does not exist.
Account Management	
addAccount(Account account)	Inserts an account into the accounts table and adds it to the corresponding savings or credit account table.
addSavingsAccount(SavingsAccount account)	Inserts a savings account into the savings_accounts table.
addCreditAccount(CreditAccount account)	Inserts a credit account into the credit_accounts table.
fetchAccount(String accountID): Account	Retrieves an account by ID and returns an Account object.
fetchSavings(String accountID): Account	Retrieves a savings account and returns it as a SavingsAccount object.
fetchCredit(String accountID): Account	Retrieves a credit account and returns it as a CreditAccount object.
fetchType(String accountNumber): String	Determines the type of an account (1 = Savings, 2 = Credit).
Existence Checks	
accountExists(String accountID): boolean	Checks if an account exists in the accounts table.
existsInSavings(String accountID): boolean	Checks if an account exists in the savings_accounts table.
existsInCredit(String accountID): boolean	Checks if an account exists in the credit_accounts table.
existsInBank(String bankID, String accountID): boolean	Checks if an account exists within a specific bank.

Account Updates	
updateAccountBalance(String accountNumber, double amount)	Updates the balance of a savings account by adding or subtracting the specified amount.
updateAccountLoan(String accountNumber, double amount)	Updates the loan amount of a credit account by adding or subtracting the specified amount.

BankDBManager

Methods

Method Name	**Description**
createTable()	Creates the banks table in the database if it does not already exist.
addBank(Bank bank)	Adds a new bank to the database if it does not already exist.
fetchBank(String bankID)	Retrieves a Bank object from the database based on its ID. Returns null if not found.
getBanks()	Retrieves all banks in the database as a HashMap<String, String>. Returns null if no banks exist.
bankExists(String bankID)	Checks if a bank with the given bank_id exists. Returns true if found, otherwise false.
getAllAccounts(String bankID)	Retrieves all accounts associated with a given bank as a HashMap<String, String>.
getCreditAccounts(String bankID)	Retrieves all credit accounts under a given bank. Uses an INNER JOIN with credit_accounts.
getSavingsAccounts(String bankID)	Retrieves all savings accounts under a given bank. Uses an INNER JOIN with savings_accounts.
getBankProcessingFee(String bankID)	Retrieves the processing fee of a given bank.
getBankWithdrawLimit(String bankID)	Retrieves the withdraw limit of a given bank.
getBankDepositLimit(String bankID)	Retrieves the deposit limit of a given bank.
getBankCreditLimit(String bankID)	Retrieves the credit limit of a given bank.
getBankDoubleField(String bankID, BankFields field)	Retrieves a numeric field (double) from the database using an enum for flexibility. Returns 0.0 if an error occurs.
isEmpty(ResultSet rs)	Checks if a ResultSet contains any records. Returns true if empty, false otherwise.
extractAccountDetails(ResultSet rs)	Extracts account details from a ResultSet and returns them as a HashMap<String, String>.

TransactionManager

Methods

Method Name	Description
static void createTable()	Creates the transactions table if it does not exist. The table stores transaction logs associated with an account, including transaction type and description.
static void addTransaction(String accountID, Transaction.Transactions type, String description)	Inserts a new transaction record into the transactions table with the given accountID, type, and description.
static ArrayList<String> fetchTransactions(String accountID)	Retrieves all transactions associated with a given accountID and returns them as an ArrayList<String>. Each transaction is formatted as [Acc. No. <account_id>] - [<type>] -- <description>. Returns null if no transactions are found.

DBConnection

Attributes

Data Type and Name	Description
static Connection sqliteConnection	A static connection object that manages the SQLite database connection.

Methods

Method Name	Description
DBConnection()	Creates a new database connection instance. Initializes sqliteConnection and calls the createTable() methods for BankDBManager, AccountDBManager, and TransactionDBManager. Throws SQLException or ClassNotFoundException if an error occurs.
static ResultSet runQuery(String query, Object... params)	Executes a given SQL query. If the query is a SELECT statement, it returns a ResultSet. Otherwise, it performs an INSERT, UPDATE, or DELETE operation.
static void closeConnection()	Closes the current SQLite database connection and sets sqliteConnection to null.

SQLinteraction

Attributes

Data Type and Name	Description
private static boolean verbose	A flag to enable detailed error logging.
private static Connection c	Stores the connection to the database.
private static Statement stmt	Executes SQL commands.
private static boolean initialized	Tracks whether the database is initialized.

Methods

Method Name	Description
static void start(String dbname)	Initializes a database connection using the given dbname. If already initialized, it prevents reinitialization.
static void open(String dbname)	Alias of start(String).
static void start()	Starts the database with the default name "master.db".
static void open()	Alias of start().
static void stop()	Closes the database connection. If already closed, it prevents errors.
static void close()	Alias of stop().
private static boolean executeUpdate(String sqlcommand)	Executes SQL commands that modify the database (e.g., INSERT, UPDATE, DELETE). Returns true if successful.
private static ResultSet executeQuery(String sqlquery)	Executes SELECT queries and returns a ResultSet. Returns null on errors.
static boolean createTable(String name, String sqlparameters)	Creates a new table if it does not exist.
static boolean insert(String table, String sqlparameters, String sqlvalues)	Inserts a new row into a table.
static boolean update(String table, String sqlkv, String sqlcondition)	Updates existing rows in a table based on a condition.
static ResultSet select(String column, String table, String rowfilter)	Selects specific columns from a table with an optional condition.