My Fancy Bank

By: Ting Hsu, Victoria-Rose Burke, Peter Gilbert

Features:

* Ability to add Savings and Checking accounts
  + Charge a fee whenever an account is opened or closed
  + Charge a fee whenever a checking account transaction or withdrawal is made
  + Give interest to savings accounts with high balances
* Maintain deposits in at least 3 different currencies
  + For when you deposit allow to input in 3 different currencies
* Take out loans (if they have collateral)
* Make a GUI for the ATM
* View Transactions
* View Current Balances of accounts
* Bank Manager should have the ability to check up on user accounts and get a daily report of transactions
* Stock trading
  + buy/sell
  + View report
  + If they have more than $5000 in their savings account -> transfer >= 1000 to a securities account
* Customer must maintain 2500 in their savings account
* Charge interest on all loans

Class hierarchy:

* BaseBank
  + Hold all the accounts in the bank
  + Hold the current currency exchange rate
  + Has a bank storage that store the user, loan and stock management systems
* User Management class
  + Can login/create users
* Loan Management class
  + For users to borrow/payback loans
  + Store the loan interest rate
  + Store all the loans that lend to the users
* Stock Management class
  + Add/edit stocks
  + Store all the stocks available
  + For users to buy/sell stock shares
* Transaction class
  + Hold the information of a single transaction
* TransactionType
  + Hold the types of transactions
* User class (information for an account object)
  + Holds username / password
* UserType
  + Hold the types of users
* NormalUser class (extends User)
  + Can borrow and payback loan
* BankManager class (extend User)
  + Can set loan interest rate
  + Can add/edit available stocks
  + Can get all the accounts created in the BaseBank
* Base Account object
  + Logic for depositing and withdrawing
  + Hold
    - User basic info
    - account balance
    - All transactions history
* AccountType
  + Hold types of accounts
* Savings Account object (extends base account)
* Checking Account object (extends base account)
* Investment Account object (extends base account)
  + Buy/sell stock
  + List realized/unrealized gains
* Loan receipt
  + Contains the borrower user account and the corresponding amount needed to be paid back
* Stock
  + Has name and price
  + Manager can add stock in StockManagement class
* Share
  + When user buy stock, they will get the corresponding shares of that stock
  + Has name, bought/sold price
* Storage - base storage class (can be expanded)
* BankStorage extends Storage
  + Writes to the JSON
  + Reads from the JSON
  + Holds information for the Bank with UserManagement, LoanManagment and StockManagement
* Login (GUI)
  + Logs in a user (Bank manager or User)
* BMinterface (GUI)
  + Window where the Bank manager can manage the bank
* UserBank (GUI)
  + Window where the user can make accounts, get stocks, loans, etc.
* Launcher
  + Class to launch the GUI and our Bank

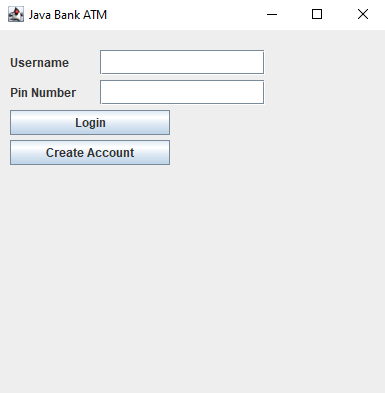
Data stored and hierarchy (in JSON) (stored in storage.json):

* Users
  + Username
  + password
  + UserType (Manager or Customer)
  + Accounts
    - User Account #
    - Account Type (Checking or Savings)
    - Money (balances)
    - Stock Information (for investment accounts)
  + Transactions
    - Type of Transaction
    - Money Exchanged
    - Origin/Destination
  + Loan information
* Current Stocks/Prices
* Exchange rates
* Loan interest rate

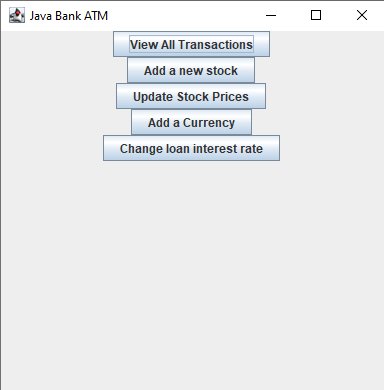
Design Choice:

We chose this design because we thought it represented a bank well. We modeled it after a banking app (ie Chase Bank). Additionally, we separated the management of the bank with our BankManager from regular Users and have separate actions for them depending on the type of account. Each user is able to make different types of accounts and deposit to them respectively. They can purchase stocks and sell them by the prices that are created by the bankmanager.

Login Page:



BankManager Page:



UserBank Page:

