Design Document

Date: 5/6/2020
Team #9

Elyse Kaczmarek
BU ID: U65134290

Eunice Choi
BU ID: U75265203

Ningxiao Tang
BU ID: U20262071

OOD files

We didn't have a UML, instead we shared a Google Doc file where we had all the classes organized in a way where we can see their relationship to one another (see below for our reference diagrams). We tried to make our app follow OOD as close as possible, and in particular ensure that we used inheritance for things that had multiple potential concrete types, and packages to contain each separate aspect of the project -- a bank package for the major OOD classes, Database package for the database interactor class, and a GUI package for the front-end classes.. The main superclasses in the OOD section were AccountType, User, Transaction. We considered having a superclass account connecting the AccountType and User classes (as they are both types of accounts), but ultimately decided against it, since they wouldn't really have any attributes or methods in common. Transaction was its separate class because just like in Quest where there was Item, here Transaction is the "Item" of Bank ATM. For the Currency class, instead of using separate classes for the different types of currencies, we decided to go with enum with the file CurrencyType. We thought that an Enumerable currency type was more applicable to the functionality we wanted to implement, since we would simply have to add a new CurrencyType to enumerate over in order to add additional types of currencies.

Database

We chose to use MySQL for the database, since we had many tables of persistent information that we needed to keep track of (accounts, bought_stocks, stocks, customers, loans, transactions). Keeping this information via simple File I/O would have gotten very complicated very quickly. In addition, the SQL call syntax was simpler to use in order to extract or insert information into the database.

We decided to create only one "database class" because, from a reusability standpoint, it would allow a different application to interact with the same database, without having to go through some concrete part of BankATM. For example, say you wanted to make a BankStatistics class. We wouldn't have to create an instance of BankATM (held in the Main file under Bank) or create any of its own new method in order to conveniently access all of the database information that is stored in the MySQL database.

GUI

Customer view consists of a left panel of option buttons, and a right panel where details of their accounts are displayed. For deposit, withdraw, and loan action, the customer first selects an account from the account table and chooses one of the actions, then a dialog window is created to handle the event. For transfer, buy stock, sell stock, and history options, a new window is created to handle the event. For stock activity, a separate stock list view is created to display stock information from the database, so it is reusable for both customers' and manager's stock view. Similarly, transaction view would retrieve transaction history from database and could be reused for manager's daily report view and customer's transaction history view. Due to time constraints, the transaction and stock list view are not fully developed.

Room For Improvement / Future Development

Were we to continue this project, we would integrate the backend with the front end further to complete the functionality of the application as a whole. We think that the overall design was well-thought-out, but unfortunately we weren't able to integrate all the features we had methods for into the final project. We also could have abstracted out some

of the GUI methods a bit more, so that we could abstract out views that were overall quite similar to one another (such as the DepositView and the WithdrawView classes), in addition to adding improved aesthetic style.

In order to integrate the features, we would simply have to call the appropriate pre-created methods from the OOD (bank package) files and the BankData (Database package) file in the correct spot of the GUI views. All of the areas we intended on implementing are there and the core structure is intact, but the integration isn't present.

Reference Diagrams

NOTE: These diagrams do not reflect the exact structure of our current code.
Rather, these are the reference diagrams we agreed upon before doing any kind of coding whatsoever. We created these so that all group-mates were aware of the design structure before implementing it, as we learned in class that this was the most efficient and optimal way of organizing a OOD project such as this.

```
########################
###    CLASS HIERARCHY    ###
########################
```

Main // Driver class that starts the program
AccountType // any type of account that can hold money for a customer
    Checking //
    Savings // has interest
    Securities // can only have when specific conditions met
User // types of users that can be made within the bank
    Manager // oversees customers, sets stocks, approves loans, etc
    Customer // can have different types of accounts
Currency // uses enumeration with CurrencyType
CurrencyType
BankData // holds persistent data -> MySQL integration
    CustomerInfo // maintain transaction history, bank balance, etc.
    StockOptions // maintains stock options
    Login // maintains & 'authenticates' usernames and passwords
Transaction
    Deposit
    Withdraw

Transfer
Stock
Loan

######################
CLASS CONTENT
######################

AccountType // abstract
+ private Currency balance
+ public static Currency accountOpenFee // set by bank manager
+ public static Currency accountCloseFee // set by bank manager
+ public static Currency withdrawlFee // set by bank manager
+ public void deposit()
+ public void withdraw()

CheckingAccount
+ public static transactionFee // set by bank manager

SavingsAccount
+ public static float percentInterest // set by bank manager
+ public boolean qualifiesForSecurities()
+ createSecuritiesAccount() // transfer any amount over $1000 to new sec account (in diff class?)

SecuritiesAccount
+ Currency realizedProfits
+ Currency unrealizedProfits
+ Stock[] boughtStocks
+ viewStockOptions()
+ buyStock()
+ sellStock()

User
+ Name name
+ String email
+ String username
+ String password
+ login() // uses txt file to approve login to account
+ logout()

Manager
+ Customer[] customers
+ Stock[] stocks
+ dailyReport()
+ getCustomerInfo()
+ changeStockValue() // also change in database

Customer
+ Checking[] checkingAccts
+ Savings[] savingsAccts
+ Securities[] securitiesAccts
+ requestLoan()

Currency
+ String type // use enumeration
+ double value
+ abstract convert() // convert between this and some other specified currency
+ add() // adds a specified amt to the current value
// ex: you can have $3.14 in USD, but when converted to Euro or RMB, it will have a different value
+ subtract() // subtracts a specified amt from the current value

BankData

CustomerInfo
+ showInfo(Customer customer)
+ transactionHistory(Customer customer)

StockOptions
+ addStock() // only available to Manager user
+ removeStock() // only available to Manager user
+ viewStocks()

Login
+ public boolean login(String username, String password)
+ logout()

Transaction
+ String affectedAccount
+ Currency amt // + is deposit, - is withdraw

Stock

+ String stockName
+ Currency stockValue
+ public void changeValue()

GUI

+ BuyStockView: The view for customer to buy stocks
+ CreateAccountMenu: The view for customer to create an account
+ CustomerInfoView: The view for manager to check all customers
+ CustomerView: The view of logged in customers
+ DepositView: The view for customer to make a deposit to the selected account
+ LoanView: The view for customer to make a loan
+ LoginMenu: Main menu of BankATM app
+ ManagerStockView: The view for manager to edit stock attributes
+ ManagerView: The view of logged in bank manager
+ RegisterView: View for customer to register in the Bank
+ ReportView: View for bank manager to check daily report
+ SellStock: View for customer to sell their stocks
+ SignInview: The view for users to sign in to the BankATM
+ StockListView: Frame for all stocks, managers can view all stocks, customers can view their bought stocks
+ TransactionsView: View of customer's transactions
+ TransferView: View for customers to make a transfer from selected account to another account in BankATM system.
+ WithdrawView: The view for customer to withdraw from the selected account