UPDATED DESIGN DOCUMENT
MY FANCY BANK ATM

The group number associated with this project is 20 (i.e, Group #20). The group members are mentioned as follows:-

- Alimkhanov, Yernur
- Gao, Tian
- Duan, Lian
- Das, Yashvardhan

This document contains a brief description of the classes used to construct the updated program specification of the required assignment. As per the new requirements, we have incorporated a new functionalities related to the working of an additional ***Security Account***.

The primary code that has been used as a base for the rest of the program is the one designed initially by Yernur Alimkhanov. The primary reasons for selecting his code as the base for the updated project are listed as follows :-

- Major reusability of code for future use.
- Proper macro structure of entities for probable modifications.
- Compatible GUI for database connectivity.

- Easy recognition of classes and respective functionalities.
- Optimized code for scalability and running time.
- Detailed usage of class composition and inheritance.
- Significant overall improvement of class structure as compared to the rest of the codes generated by other remaining team members.

The following mentioned are the *new classes* have been incorporated as part of the Entity Classes in the project :

I. SHARES CLASS

- This class is programmed so as to get the abstract design of the constituent properties of the shares available. Appropriate setter and getter methods have been coded to generate the required functionality. The instance variables include the following:
    - name of the company whose share it is,
    - the abbreviation of the name of the name of the company that is listed in the stock exchange,
    - the price of the share,
    - total number/amount of shares.

II. PRIVATESHARES CLASS

- The purpose of this class is to get the details of the shares bought by a particular user.
- This class functions as the sub-class of the *SHARES* class.
- Setter and getter methods have been programmed for the providing the details of the price at which the share was bought by the user.

III. SECURITYACCOUNT CLASS

- This class provides the primary functionalities of the updated specifications.
- Since, a security account is a type of account opened by the user, this class is a child class of the parent *ACCOUNT* class.
- Here, a condition is programmed accordingly to verify whether the user has enough balance or not to purchase a share.
- Like in the above-mentioned classes, appropriate setters and getters have been coded to display relevant details about information of the user, list of shares, etc.

IV. STOCKMARKET CLASS

- The functionality provided by this class is to display the available shares in the exchange market.
- The instance variable programmed in this class includes an array-list of the object of the *SHARES* class.
- The requisite setter and getter methods have been programmed to get the information about the shares available.
- This class provides a basic database-like structure for garnering information about the shares present in the market/exchange.

The following mentioned are *existing classes* that were incorporated in Yernur Alimkhanov's base code. This classes was created to save all required attributes of the bank. The main class of this group of classes is Bank. It consist of main attributes of bank. I made this class because during different operation in different pages, I need a place, where I will get all information about the bank. Classes from Entity group are all generic classes

ENTITY CLASSES:

- Account
- Bank
- BankUser
- CheckingAccount
- Client
- Loan
- Manager

- SavingAccount
- Transaction

This classes is used to create JTables in GUI. We don’t have built - in support of classes in JTable. We need to create additional classes which will represent a list of object of class in table mode. Several GUI pages uses JTable to represent information to the user. 5 main classes(CheckingAccount, Client, SavingAccount, Loan, Transaction) has 5 connected classes which represents list of object of given classes as a table. All these classes are extended from AbstractTableModel class. We have different types of fee for different operations. All these fees are saved as static values in Account class, when client do different action to account, it will charge these fees. The main problem for me was a fees for account closing and opening. This are one time fees, but when client opens his first account, I did not understand how should pay Account opening fee. Because there is no account with money in client profile. I chose to solve this problem by adding otherFee attribute in Client class. This attribute wi ll show amount of money which client should pay for account opening and closing operations. Client can do it in page “Other fees” in his main menu.

TABLEMODEL CLASSES:

- CheckingAccountTableModel
- ClientTableModel

- LoanTableModel
- SavingAccountTableModel
- TransactionTableModel

DATABASE CONNECTIVITY USING JPA

We have utilized the **Java Persistence API** for connecting the back-end database with the overall program and GUI. A persistence entity is a lightweight Java class whose state is typically persisted to a table in a relational database. Instances of such an entity correspond to individual rows in the table. Entities typically have relationships with other entities, and these relationships are expressed through object/relational metadata. Object/relational metadata can be specified directly in the entity class file by using annotations, or in a separate XML descriptor file distributed with the application.

The primary reasons for our choice to utilize JPA for the database connectivity are described as follows:-

- JPA is an entity-based database management platform. It presents us an advantageous opportunity to not make

significant changes to the existing the code for the addition of database functionalities.

- The only modifications that are to be done are alterations of java entities into JPA entities and establish suitable connections between the respective entities.
- The primary class for the database management functionality is provided through the *DatabaseManager.java* class. It includes all the primary provisions for the project service and JPA's connectivity.
- The provision of JPA allowed us to code and add the functionalities of savings and related data items to be modified and updated the *DatabaseManager.java* class. There was no need for other significant individual additions of entity objects.

EVALUATION OF STARTING DESIGN

The choice of the starting design was a successful one. In our opinion, the planning and execution of the subsequent programming design worked out pretty well. The reasons for validating this viewpoint are briefly described as follows :-

- Owing to the significant structured design of the code along with its prominent reusability, there were no drastic changes added to include the new functionality of the security account.

- A significant advantage in the new code was that there was no need of the enumerators (enums) that were previously programmed for the currency alteration features. Since, we incorporated a database using JPA, the data was fed through that instead of declaring new enums for the same purpose.

DESCRIPTION OF JAVA SWING CLASSES

The java swing classes can be classified into two categories, namely that of the Manager entity and the User entity. These are briefly described as follows :-

- **MANAGER ENTITY SWINGS**
  - **ManagerMenu**
    - This provides the main window/menu for the bank manager to view the respective account details.
    - The manager can add stock shares and alter currency exchange rate.
  - **ManagerClientAccountsPage**
    - This swing window gives the provision to view all the accounts of the respective clients who are selected.

        - Through this menu, the manager can change the savings account and alter the balance by its interest rate.

    - **ManagerClientLoanPage**
        - This window is utilized by the manager to view all the active loans of the client.
    - **TransactionsListPage**
        - This swing window provisions the display of the respective client’s undertaken transactions.
    - **ChangePricePage**
        - The functionality of this swing window is utilized by the manager alter the share price.
    - **CurrenceyExchangePage**
        - This window provisions the functionality to alter the exchange rate with respect to individual currencies.
    - **CreateSharesPage**
        - This window is utilized to create new shares of companies in the stock market.

- **USER ENTITY SWINGS**
    - **AccountListPage**

        - This window presents the list of savings and checkings accounts of the client/user.
    - **AccountManagementPage**
        - This swing page is utilized by the client to create new accounts and delete existing accounts.
    - **ClientConfirmationPage**
        - This window is displayed for showing the necessary information during the client's authorization process.
    - **ClientLoginPage**
        - This window provides the functionality for the client to create a new client profile.
    - **ClientPersonalPage**
        - This page displays all the possible actions of the ATM to the client.
    - **DepositPage**
        - Through the provisioning of this window, the user can deposit money into his/her account.
    - **LoanPage**
        - This swing page can be utilized by the client to display all his/her loans.
        - The user can also make the provision of acquiring a new loan through this window.

- **LoanPayingPage**
  - To pay back the loan amount, the user does so by paying monthly installments. This is provisioned by this swing window.

- **OpeningMenu**
  - This is the swing window where the user will choose what category/type of client will be used into consideration.
- **OtherFeePage**
  - This swing page is utilized by the user to pay other miscellaneous fees.
- **ReturningClientPage**
  - This window provides the login access for returning clients.
- **SecurityAccountListPage**
  - This window provisions the list of the respective security accounts.
- **SecurityAccountPage**
  - This is the main menu for displaying the details of the security account.
- **TransferPage**

- This swing page provisions the user to transfer money from one account to the other.

- **WithdrawPage**
  - This swing page is used to withdraw money from the respective account.