**UPDATED DESIGN DOCUMENT**

***MY FANCY BANK ATM***

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 entity classes have been incorporated as part of the Entity Classes in the project :

1. **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.
2. **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.
3. **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.
4. **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.

**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.