This documentation has 3 parts:

1. Class design
2. Choice of starting design
3. Evaluation of starting design decision

Part 1 Class design

For each class, we listed 3 essential points of our design:

1. purpose of the class
2. benefit of this class to the existing design
3. benefit of this class, if any to a future bank

We highlighted changes made for security account and stock transactions.

Model

This package contains the date structure classes of the project.

Bank

1. This class defines a bank.
2. It contains the data of a bank.
3. In the future, it can be used by other bank systems.

Account

1. This class defines an account.
2. It records an account for a user and contains data of any type of account.
3. It is used to define a specific type of account. All accounts can be put together by polymorphism.

CheckingAccount extends Account

1. This class defines a checking account.
2. It contains all information of a checking account.
3. Provide code reusabililty: using this class, the bank system can create any number of checking accounts easily.

SavingAccount extends Account

1. This class defines a saving account.
2. It contains all information of a saving account.
3. Provide code reusabililty: using this class, the bank system can create any number of saving

accounts easily.

SecurityAccount extends Account

1. This class defines a saving account.
2. It contains all information of a security account.
3. Provide code reusabililty: using this class, the bank system can create any number of saving accounts easily.

Currency

1. This class defines a currency.
2. It contains data of a currency type.
3. In the future, it can be used anywhere there are currencies.

CurrencyConfig

1. This class defines the configurations of a currency.
2. The configuration can be changed by manager and helps the manager to collect money.
3. In the future, it can be used anywhere there are currencies.

DailyReport

1. This class defines the daily report.
2. This shows the daily report and helps the manager to manage the bank well.
3. In the future, it can be used anywhere there are transactions and wants to show the details of the report.

Date

1. This class defines the date.
2. It contains the elements date has.
3. In the future, it can be used anywhere date exists.

Loan

1. This class defines loan.
2. It contains the elements a loan has.
3. In the future, it can be used anywhere loans are needed.

Name

1. This class defines the name of a person.
2. It contains the elements a name has.
3. In the future, it can be used anywhere names are needed.

Person

1. This class defines a person.
2. It contains the elements a person has.
3. In the future, it can be used anywhere there is a person.

PhoneNumber

1. This class defines a phone number.
2. It contains the elements a phone number has.
3. In the future, it can be used anywhere phone numbers are needed.

Transaction

1. This class defines a transaction.
2. It contains the elements a transaction has.
3. In the future, it can be used anywhere there are transactions.

User extends Person

1. This class defines a user.
2. It contains the elements a user in bank has.
3. In the future, it can be used in other bank systems.

Controller

This package contains the back end classes of the project.

BankATM

1. This is the entrance of the project.
2. It helps set an entrance to start the project.
3. In the future, it can be changed the name and be the entrance of another project.

SystemInterface

1. This interface defines functions that a system should have. Here it means the functions that user and manager have in common.
2. It specifies the methods that the system must implement.
3. In the future, other system can be implemented which must register, login and logout. Some other methods can be added if need.

BankController

1. This is the controller for bank or manager. It contains the methods that bank system used.
2. It deals the operations that front end called in bank system.
3. This is designed for this bank ATM specifically. But some methods can be used in other manage systems.
4. Added methods that allows manager to add a stock and modify its prices.

UserController

1. This is the controller for user. It contains the methods that user system used.
2. It deals the operations that front end called in user system.
3. This is designed for this bank ATM specifically. But some methods can be used in other user

systems.
4. Added methods for purchasing and selling stocks.

Utils

This package contains some global functions and variables.

Config

1. This class defines some global variables, constants and default configurations.
2. It prevents specific number from appearing in the project and instead replaces them with variables for easily maintenance.
3. The variables and constants are defined for this project.

ErrCode

1. This class defines the error code for the project.
2. It can easily show the error message to users.
3. The variables and constants are defined for this project.

UtilFunction

1. This class defines some utility methods that used in the whole project.
2. The same as 1.
3. The methods are defined for this project. But some of the methods can be reused in other projects.

View

This package contains all the views for this project.

AccountDetail CheckBalance CheckCustomer ChooseIdentity CurrencyConfig GetDailyReport Login ManagerInterface Register SetConfig StockManagement StockMarket StockTransaction TakeLoan Transact TransactionDetail UserDetail UserInterface

Part 2 Choice of starting design

We chose to start from Qi Yin's FancyBank project as starting design.

Reasons and benefits:

1. This version has complete MVC structure, which is appropriate and neccessary for this project.
2. It implements all featured required without bugs or errors.
3. This version is more robust than others in terms of object design and GUI design. a. This version has an Account class, and thus we can create the security account by extending it. b. The GUI of

this version is more user-friendly than others.

Part 3 Evaluation of starting design decision

Our decision on the starting design worked out well in terms of features, object design and the actual implementation.

In our starting design, basic features (e.g. opening an account, recording transactions, rending user interface, etc.) can work correctly. This provides a solid fundamental for us to build new features upon it.

As for object design, the starting design provides classes and methods that can be reused easily.

The starting design uses appropriate data structures that facilitate database connection and data operations. It also provides input checking and considers corner cases.