CS611 - Assignment5 - Bank

Submit By

- Zhaozhan Huang | zhzh@bu.edu
- Tianling Gong | tlgong@bu.edu
- Tony Cen Cen | tcen@bu.edu

How to Run

In the root directory of the project:

```
$ javac -d ./bin ./src/**/*.java
$ java ./bin/Main
```

Notes

Architecture

1. We have a frontend, a backend, and a database.
2. The frontend is running in ./src/frontend.
3. The backend is running in the other parts
4. The data is stored as .txt files in ./storage

Design Patterns

1. We use the Observer Pattern to update user balance. See account/BalanceListener
2. We use the Singleton Pattern to make sure we only have 1 bank.

Extensibility and Scalability

1. The data is stored in well-structured .txt files. So we can seamlessly integrate it into online database.
2. We use abstract class and interfaces. The app can have more types of account.

Citations

https://docs.oracle.com/javase/tutorial/uiswing/ (https://docs.oracle.com/javase/tutorial/uiswing/)

https://www.javatpoint.com/java-swing (https://www.javatpoint.com/java-swing)

https://www.geeksforgeeks.org/introduction-to-java-swing/ (https://www.geeksforgeeks.org/introduction-to-java-swing/)

Files

./src/account/Account.java

This is the abstract class representing an account. CheckingAccount, ManageAccount, SavingAccount and StockAccount inherit from Account.

./src/account/BalanceListener

This is the interface for the Observer Pattern.

./src/account/CheckingAccount

This is the class representing a checking account owned by our user.

./src/account/EnumAccount

This is the enum class representing the type of an account. Currently we have Checking, Saving, Manage, and Stock.

./src/account/ManageAccount

This is the class representing a manage account owned by our manager.

./src/account/SavingAccount

This is the class representing a saving account owned by our user.

./src/account/StockAccount

This is the class representing a stock account owned by our user. It can be used in the stock market.

./src/bank/ATM

This is a class representing the functionality of an ATM. It interacts with our frontend

./src/bank/Bank

This is the entry of our backend bank system. A List of users, A List of transactions, and an ATM is stored in the bank class.

frontend

TODO

./src/role/AccountListener

This is the interface for the Observer Pattern.

./src/role/User

This is the abstract class representing a user of our app. We have 2 kinds of user: Customer and Manager.

./src/role/Manager

This is the class representing a manager of the bank.

./src/role/Customer

This is the class representing a customer of the bank.

./src/role/EnumRole

This is the enum class of User. We have 2 kinds of user: Customer and Manager.

./src/stock/Market

This is the class representing the stock market. It uses the Singleton Pattern.

./src/stock/Stock

This is the class representing a single stock in our market.

./src/stock/StockEntry

This is the entry of a stock.

./src/stock/StockListener

This is the interface to implement the Observer Pattern

./src/transaction/Transaction

This is the abstract class for transaction. Deposit, Loan, Stock_Transaction, Trade inherit from this abstract class.

./src/transaction/Deposit

This is the class representing the deposit transaction.

./src/transaction/Loan

This is the class representing the loan transaction.

./src/transaction/Stock_Transaction

This is the class representing the stock transaction.

./src/utility/ColorPrint

This is the class holding the utility functions to print colorful message.

./src/utility/Constants

This is the App-level constants

./src/utility/InputHandler

This is a singleton class holding the utility functions to get valid input from the user

./src/utility/Read

This is the class holding the utility functions to input data from our database.

./src/utility/Read

This is the class holding the utility functions to output data into our database.

./src/Main

This is the Main class.

./storage/

This the folder serving as a database. It contains all the txt files in our App.