Note - Updated after Presentation:

- *Error Message now thrown when negative input is given for Deposit/Withdrawal/Transfer*
- *Creating a new account always uses updated accountIDs.*

Documentation of the classes you implemented.

Implemented Classes:

- GUI:
    - GUIAccount: Creates the GUI that shows account details.
    - GUIAccountManager: Creates the GUI that shows account details to the manager
    - GUIAccountsHome: Creates the GUI that shows the account tab.
    - GUIAccountsOverview: Creates the GUI that shows all the user's accounts.
    - GUIBank: Creates GUI for the online Bank
    - GUIDailyTransactions: Creates the GUI that shows daily transactions to the manager.
    - GUIDeposit: Creates the GUI that allows users to deposit money.
    - GUIHome: Creates the GUI that allows users to view accounts, take out loans, invest, and manage accounts (if user is a manager).
    - GUIInvestmentHome: Creates the GUI that allows users to play in the stock market.
    - GUILoan: Creates the GUI that allows users to manage their loans. Our implementation only allows users to take out one loan at a time.
    - GUILogin: Creates the GUI that allows users to login.
    - GUINewAccount: Creates the GUI that allows users to create new accounts.
    - GUIPosition: Creates the GUI that allows users to see their investment positions.
    - GUIRegister: Creates the GUI that allows new users to register.
    - GUISettings: Creates the GUI that allows users to manage their username and password.
    - GUIStockOrder: Creates the GUI that allows users to view a single stock order.
    - GUIStockOrderHistory: Creates the GUI that allows users to see their entire stock order history.
    - GUIStockPositions: Creates the GUI that allows users to see all their investment positions.

    - GUIStockTrade: Creates the GUI that allows users to trade stocks.
    - GUITransaction: Creates the GUI that allows users to see a single transaction.
    - GUITransfer: Creates the GUI that allows users to transfer money across accounts or to different users.
    - GUIUser: Creates the GUI that allows managers to see a single user's information.
    - GUIUsersOverview: Creates the GUI that allows managers to see all users' information.
    - GUIWithdraw: Creates the GUI that allows users to withdraw money.

- Interfaces:
    - TransactionInterface: Ensures that objects implementing this interface have the transaction functionality.

- Objects:
    - Account: Implements the TransactionInterface and provides a base for all types of accounts.
    - Checking: Extends Account to represent a Checking account.
    - Loan: Represents a loan that the user could have.
    - Manager: Extends Person to represent a Manager of the bank.
    - Person: Represents a person that can interact at the Bank.
    - Position: Represents the details of a stock that the buyer have
    - Saving: Extends Account to represent a Savings account
    - Security: Extends Account to represent a Security Account.
    - Stock: Represents a stock
    - StockOrder: Represents a stock order.
    - Transaction: Represents a transaction that the user could have within an account
    - User: Extends Person to represent a user of the bank.

- Types:
    - AccountType: Enumerates the types of Accounts and their string representations.
    - CurrencyType: Enumerates the types of currencies and their values.
    - Status: Enumerates the status.
    - StockOrderType: Enumerates the types of stock orders.
    - TransactionType: Enumerates the types of transactions

- Util:
    - AccountManager: Helper class to manage accounts.
    - CurrencyConverter: Helper class to convert money.
    - DataManager: Helper class to read and write to text files
    - MD5Util: Helper class to encrypt our passwords

Explaining your design choices, object model, object and GUI relationship, benefit of your design.

Design Choices:

- Data persistence: To ensure that data persists across different runs of the app, we chose to keep it simple and use text files to store our data. Our text files include the following with each field separated by a " | ":
    - accountLog.txt
        - Date of account creation
        - Time of account creation
        - UserID associated with account
        - Type of account (Checking, Savings, or Security)
        - AccountID
        - Account balance
        - Account status (Active or Inactive)
        - Currency type (USD, EUR, INR, GBP)
    - loanLog.txt
        - UserID associated with the loan
        - Loan amount
        - Paid amount
        - Date and time of loan creation
        - Loan status (Active or Inactive)
    - managerLog.txt
        - Date
        - Time
        - Username
        - Password
        - UserID
    - positionLog.txt
        - Stock symbol
        - Total value of position

        - Quantity of stock
        - AccountID associated with position
- stockLog.txt
    - Stock name
    - Stock symbol
    - Current price of stock

- stockOrder.txt
    - Date and time of order
    - Stock symbol
    - Quantity of stock in order
    - Price per stock
    - Type of order (Sell or Buy)
    - AccountID associated with the order
- transactionLog.txt
    - Date
    - Time
    - UserID associated with transaction
    - AccountID associated with transaction
    - Name of transaction
    - Amount of transaction
    - Type of transaction (Deposit or Withdrawal)
- userLog.txt
    - Date
    - Time
    - Username
    - Password
    - UserID

** The benefit of using text files to store our data was that we did not have to go through the hassle of setting up a database. It also allowed us to organize our data in a way that made sense to us rather than conforming our data to a model to fit other formats.

** There were some challenges that came along with this design choice. For example, there were often small bugs that came when reading from the files. If there was a space after the userID at the end of the entry, the space would be included when we read in the entry and created the user object.

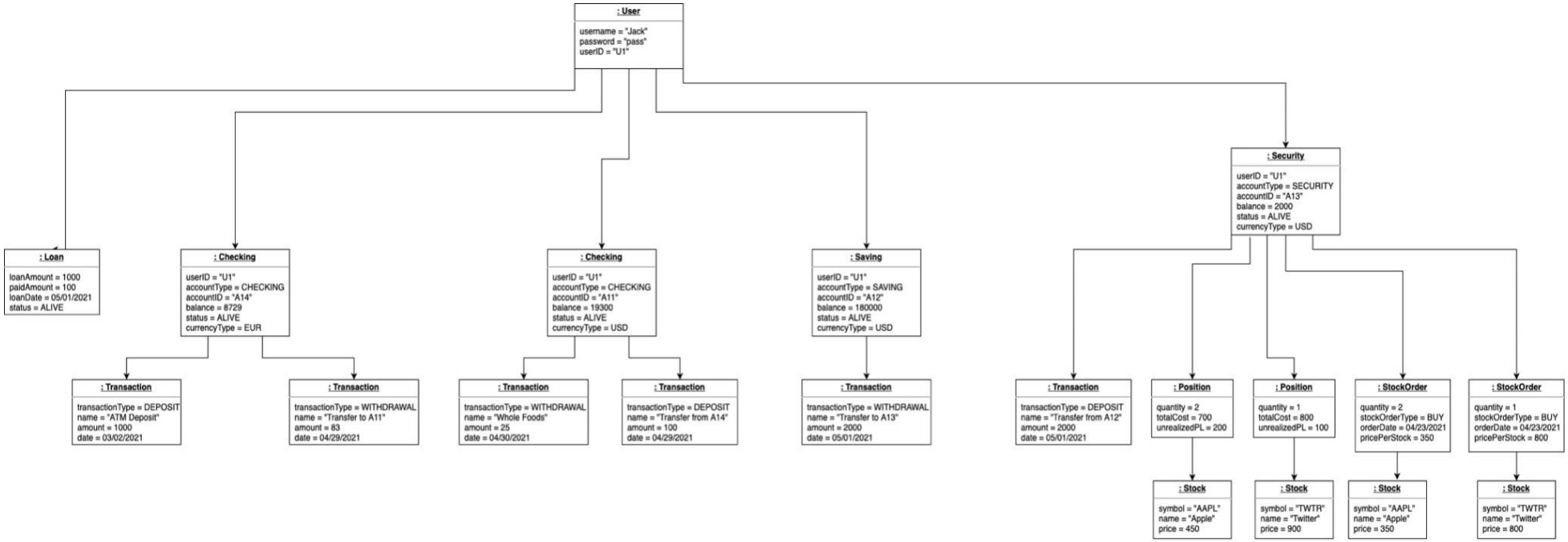
- We tried our best to keep the logic of the Bank and the GUI as separate as possible. We put most of the user input validation on the GUI side and the reading and writing to our “database” on the Object side.

  ** The benefit of keeping the logic and the GUI separate was that doing so provided a level of abstraction as well as kept the code more organized throughout the project.

Object and GUI Relationship:

Most of the GUI pages have an instance of the logged in person. This allowed us to know which user’s information to display across the application. However, this created some issues in updating the GUI to reflect changes when users interacted with the Bank as we instantiated all the components in the constructor of each GUI file. We created a workaround for this issue by creating an update function in the GUIs that is called when the user makes a change that should be reflected.

Objects have their own methods that are called when a user interacts with the GUI, such as by clicking a button or inputing new text. The information for each object is stored in its respective Log, which is handled by the DataManager. The DataManager takes the values of the variables associated with a given object and writes them to the proper log. The DataManager is also able to load in one or all records of an object and instantiate them.



This is an object diagram representing an instance of a User.