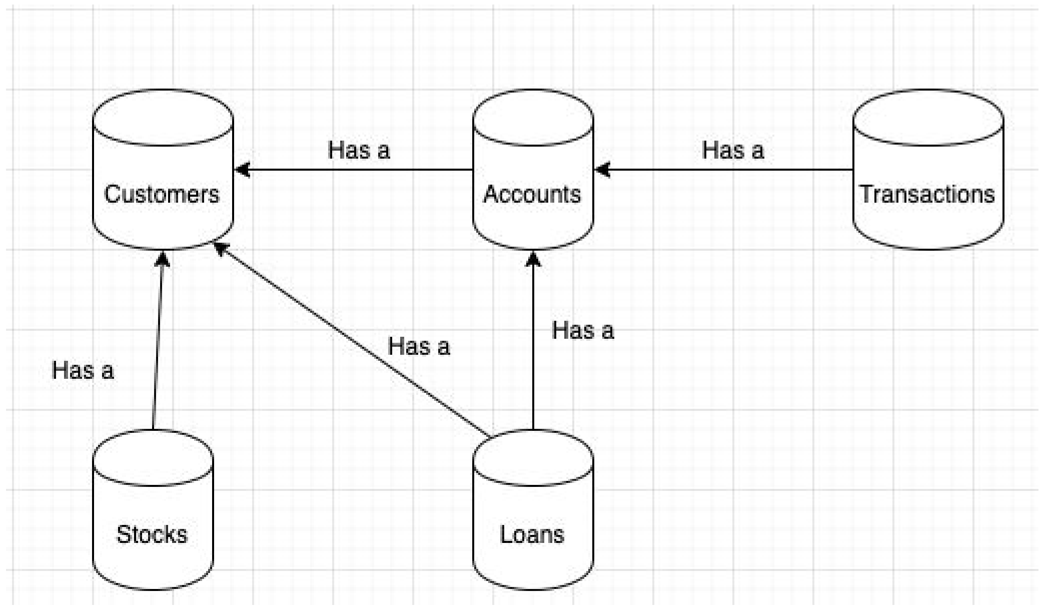
Team 6
Nathan Levy
Kelsey Myton
Minglan Zheng

Design Document

Relational Database Schema



Class files

- Account: Abstract class for various types of accounts. All accounts have a name, starting balance, an owner, and an opening date. We designed it so that Account is extendable to various accounts that servious various functionality other than the basic withdrawal and deposit.
    - Account Implements BankAccount interface
- CheckingAccount: Extends account and also takes in a currency.
- SavingsAccount: Extends account and adds in functionality for calculating the interest.
- SecuritiesAccount: in addition to the basic functionality, also provides a method that uses the static variables declared in Bank to calculate if the open condition of this account is met. So the class is extendable to future circumstances when the opening conditions change.

- BankAccount: Interface for an account. All Accounts should be able to deposit money, withdrawal money, and track the day the account was opened. It increases the readability of our program by specifying the behavior of bank accounts and parameters of methods to get the account name, account ID, and get money.

- Bank: creates static variables for rates such as loanRate and value such as savings interest benchmark, and provides methods to modify and access these variables so these conditions can be modified and increase the extendability of the program.

- User: Abstract class for all users of an online bank. Users have a username and password.
- Customer: Extends User. Customer implements the Comparable interface and provides a compareTo method which helps order customers in amount of savings. Customer also implements the OPObserver interface, by making the StockMarket (OPSubject) one of its attributes, StockMarket updates Customer who is registered with the new list of available stocks whenever a stock trade takes place. Customers have a first and last name, an ID for the database, stocks, Lists of checking accounts, savings accounts, and securities accounts. Customers can deal stocks and request loans
- BankManager: Extends User. The attributes of BankManager ArrayLists to specify customers, loans,etc. to make the scale Bank service scalable. Uses the LoanComparator object to create the list of customers dealing loans. Provides good encapsulation by allowing only manager the access to modify the Time in the bank with a method setBankTime that only allows for changing the time into the future. This implementation can be improved because it lacks the functionality of approving loans after customers submit them.

- Transaction: Abstract class that represents a transaction. It has an account, date, amount, memo, ID for the database, and currency.
- Withdrawal: Extends Transaction class. Can withdraw money from an account and update that account balance. Updates database with information.
- Deposit: Extends Transaction class. Can deposit money to an account and update that account balance. Updates database with information.

- Currency: Abstract class that represents currency. To provide good encapsulation sets the field conversionFactor with a method provided in Bank so Bank.java specifies methods for modifying or expanding the conversion factor

list, so the variety of Currency is unlimited. It is implemented with clarity by interpreting value in USD and providing methods to convert to and from USD.
- Dollar: Extends Currency and represents a Dollar currency.
- Euro: Extends Currency and represents a Euro currency.
- Yen: Extends Currency and represents a Yen currency.
- Pound: Extends Currency and represents a Pound currency.

- Service: Abstract class that represents a service that the bank offers for customers. Since the bank is highly service-oriented, we capture the basic behaviors of a service such as its interest, opening date, and close condition in Service to make it an extendable case.
- Loan: Extends the Service class and represents a loan. It has an ID for the database, memo, interest rate, due date, lastPayDate, and collateral. By keeping the lastPayDate, we effectively collect interest over the period.
- LoanComparator: This implements the Comparator interface. It allows us to order customers in order of their amount of loan.
- StockService: Extends Service. Provides methods to calculate the unrealized profit and implements the abstract method interestOnService to calculate the newly earned/lost profit in a trade.

- DBConnect: This class establishes connection to the MySQL database and contains static methods called throughout the code for reading and writing to the database.

- Clock: This class represents the time in the bank. It has a day, month and year. It implements Comparable interface so that clocks can be compared to each other to get the difference. You can set the clock's attributes and set it to a future time.

- OPOberver: Interface for the observer in the Observer Pattern, it should update the current available stock and clear securities account of this Observer. It also has an equals and getname method.
- OPSubject: Interface for the subject in the Observer Pattern. It should registers observer, unregister observer, and notify observer.
- Stock Market: Provides methods for adding or removing a Stock from the market. Implements the OPSubject interface and provides and calls a method notifyObserver() whenever a trade takes place which updates the list of currently available stocks with the registered observer. This implementation of the Observer pattern allows for abstract dynamic coupling between the StockMarket and the Customers, such that the StockMarket knows nothing about the concrete

class of Customer and provides good encapsulation since it only has access to the methods specified in OPObserver interface.
- Stock: Represents a stock. It takes an ID for the database, name, value of the stock, and the owner which is a customer. You can get the current value of the stock, it's name and ID.

GUI Components

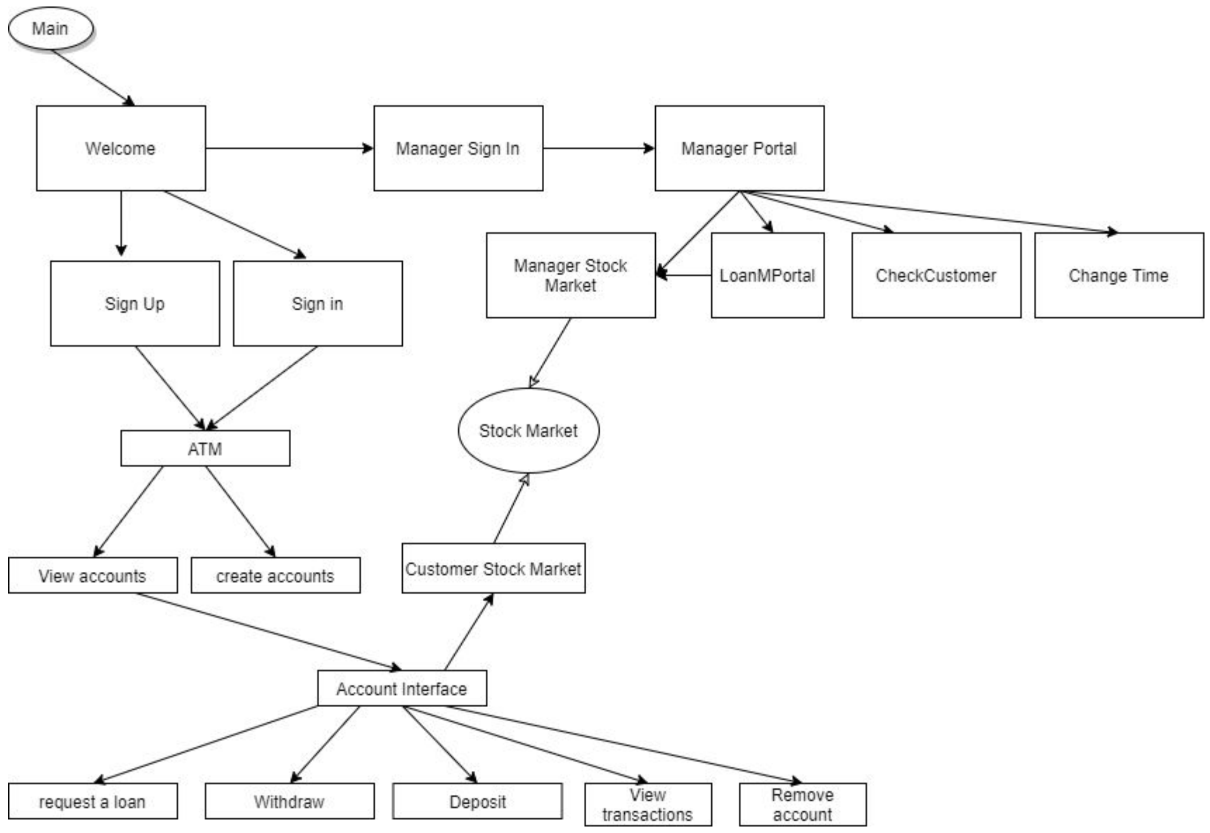
- Main: runs Welome.java.
- Welcome: It is the frame that directs users to either sign up or sign in. ManagerSignIn extends sign in but instead directs user to manager portal
- SignUp: Allows users to enter a First and last name, username and password to sign up and have access to their accounts.
- SignIn: Allows users to enter a username and password to log them in so that they can access their accounts.
- ManagerSignIn: Manager can sign in(Username: username, Password: password) and gain access to the manager portal.

- ATM: Main user interface for all customers. When a customer logs in and creates an account, they are directed here. Here, they can open a new account or view their already created accounts.
- CreateNewAccount: You can choose to either create a new checking account or a new savings account
- CreateNewSavingsAccount, CreateNewCheckingAccount: In each of these, a customer can enter an account name, starting balance, and currency to create a new account.
- viewAccounts: You can choose to view your checking accounts, savings accounts, and securities accounts.
- viewCheckingAccount, viewSavingsAccount, and viewSecuritiesAccount: In each of these, a customer can choose an account and that will lead them to the account interface.
- AccountInterface: The interface for a specific account. Must take in a customer and an account as an input in the constructor. It is at this frame where customers can withdraw (WithdrawFrame.java), deposit (DepositFrame.java), request loans to that account (LoanFrame.java), view that account's transactions, buy stocks and remove the account.
- WithdrawFrame: Allows customers to withdraw money
- DepositFrame: Allows customers to deposit money and chose a currency
- LoanFrame: Allows customers to request a loan by entering the amount and the collateral.

- BuyStock: In this frame customers can see available stocks and purchase them by entering the name of the stock and the number of shares. When purchased, the customer can see the openings summary displayed.

- ManagerPortal: Main interface for manager. From this interface the manager can navigate to CheckCustomer, CustomerView, CheckTransaction, LoanMPortal, TimeMPortal
- CheckCustomer: See any customer’s information and accounts by entering the customer’s username
- CustomerView provides customer information on frame for manager
- CheckTransaction: See all transactions from given day
- LoanMPortal: Frame where bank manager can view all loans
- TimeMPortal: Frame where bank manager can update time by entering month, day, and year

Graphical User Interface Flowchart


Main
Welcome
Manager Sign In
Manager Portal
Manager Stock Market
LoanMPortal
CheckCustomer
Change Time
Sign Up
Sign in
ATM
Stock Market
View accounts
create accounts
Customer Stock Market
Account Interface
request a loan
Withdraw
Deposit
View transactions
Remove account