Design choices and benefit:

Low object coupling:

When we design the relationship between classes, we tend to use “has a” relationship over “is a” relationship, which lowers the coupling of each object. Because of the use of more “has a” relationship, our classes can be reused for future development and increase extendability.

Factory pattern:

We use the factory pattern to create accounts. This helps us put the creation accounts method into a single class, and it’s easier to manipulate those account creation methods, and easier to use by just calling different factory methods.

Model View Controller pattern:

For the GUI, we implemented a model view controller model, which breaks up the frontend and backend code into separate components. This way, it's much easier to manage and make changes to either side without them interfering with each other.

Object model:

Inheritance:

Service is an abstract class, inherited by LoanService and StockTradingService
Account is an abstract class, inherited by AccountChecking, AccountSaving, and AccountSecurity
Personnel is an abstract class, inherited by Customer and Manager
Currency is an abstract class, inherited by CurrencyEURO, CurrencyRMB, CurrencyUSD

Components:

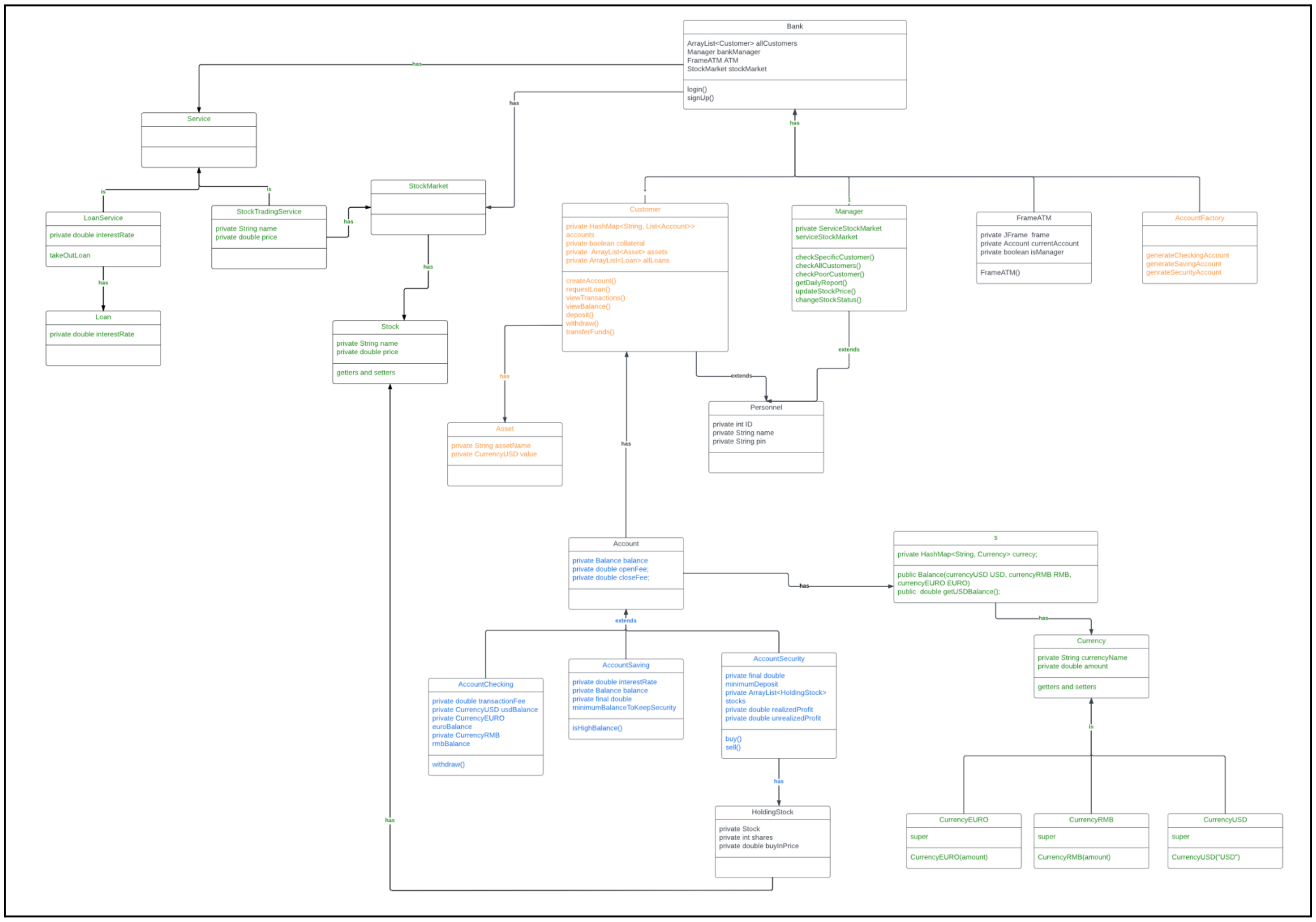
Other than the inheritance, other classes are components. For example, StockMarket has multiple Stocks, a Customer has multiple Accounts, Accounts have different Currencies.
In terms of actual JComonents, we also developed a reusable and scalable component called ButtonList, to which we can add labels and buttons line by line and access individual components based on line index.

Objects and GUI relationship:

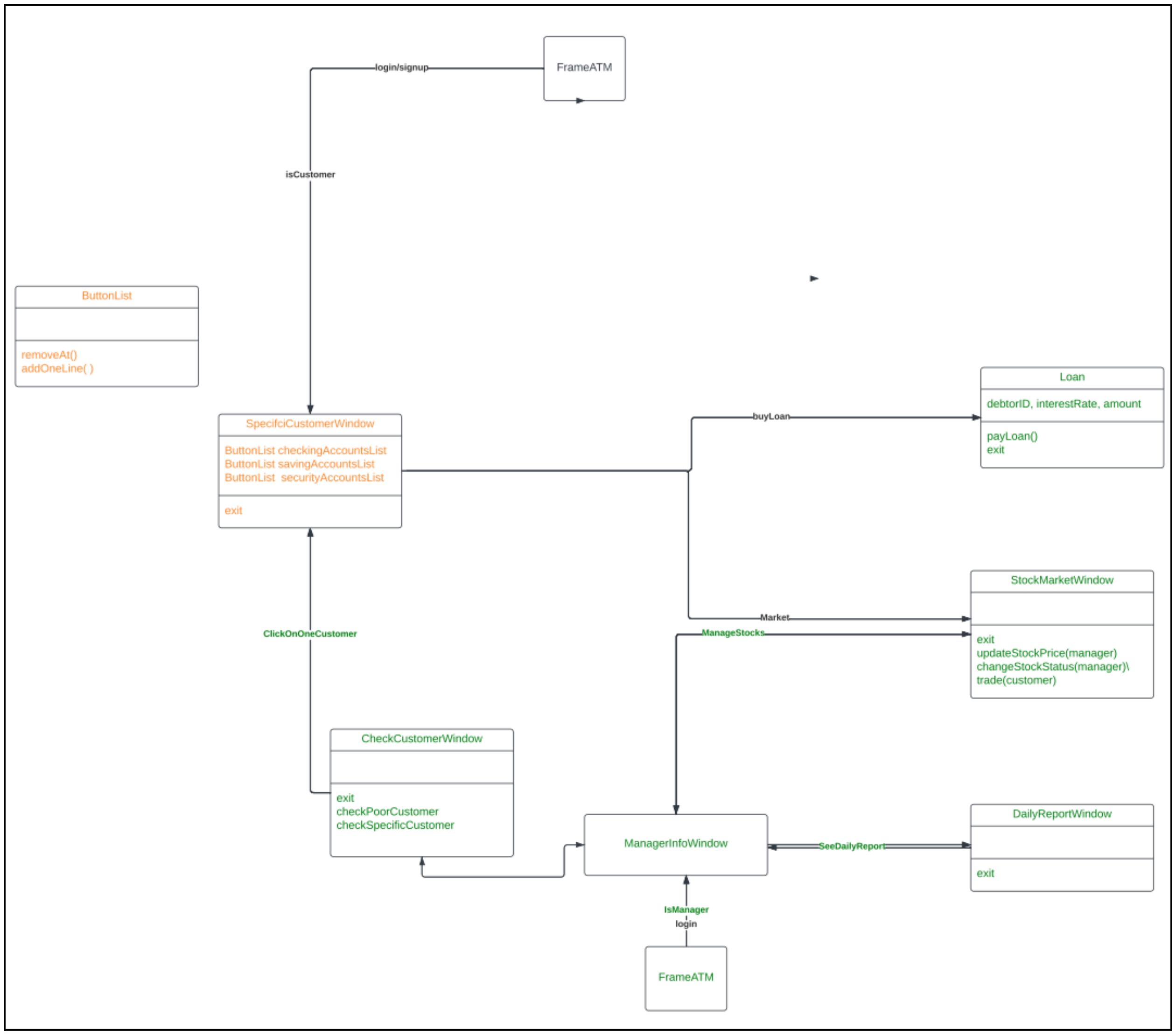
First we have a login/signup page, after login successfully, the application will create a Customer object or Manager object, and pass it into either the customer’s page or the manager’s page. By passing the Customer/Manager objects, the GUI can retrieve those personnels’ information, and use the information to perform corresponding actions.

UML:

Backend UML:



View(frontend) FlowChart:



FrameATM
login/signup
isCustomer
ButtonList
removeAt()
addOneLine( )
SpecifciCustomerWindow
ButtonList checkingAccountsList
ButtonList savingAccountsList
ButtonList securityAccountsList
exit
buyLoan
Loan
debtorID, interestRate, amount
payLoan()
exit
Market
ManageStocks
StockMarketWindow
exit
updateStockPrice(manager)
changeStockStatus(manager)\
trade(customer)
ClickOnOneCustomer
CheckCustomerWindow
exit
checkPoorCustomer
checkSpecificCustomer
ManagerInfoWindow
SeeDailyReport
DailyReportWindow
exit
IsManager
login
FrameATM