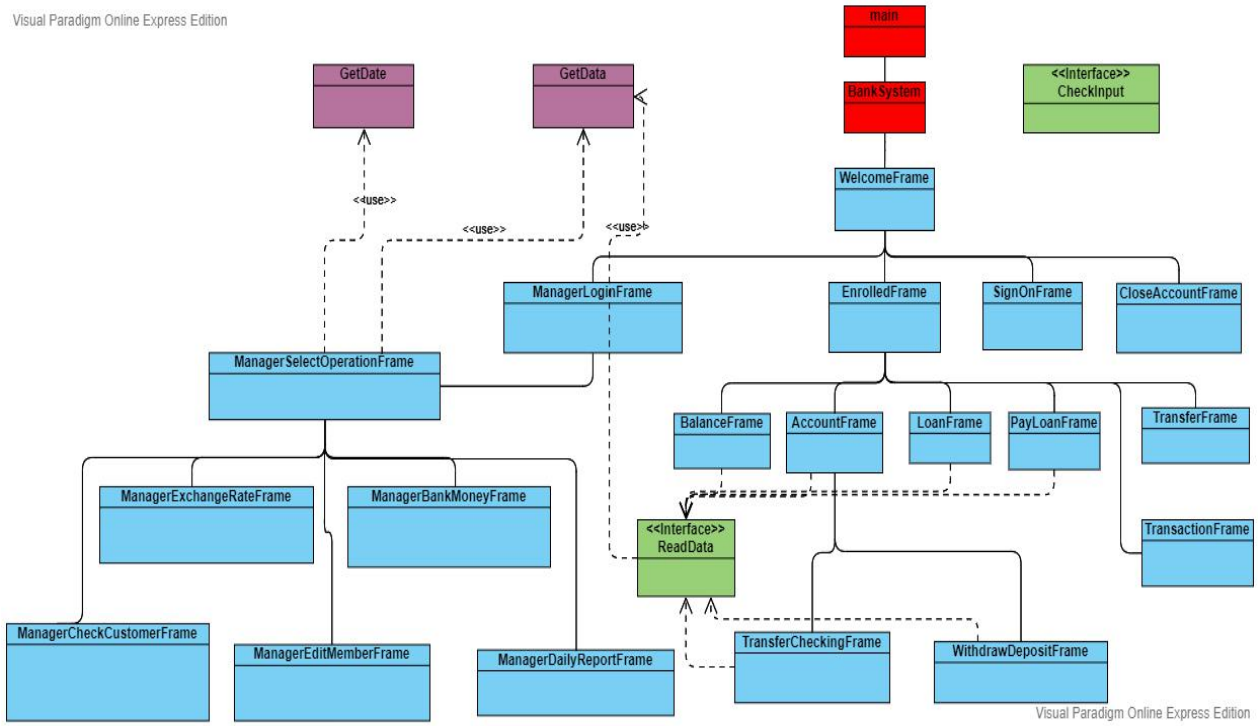
Object Design Documentation of Bank System

Object Design Diagram:



Description of each class:

AccountFrame:

A class representing the UI of the user account

Purpose: to show the UI for users to deposit or withdraw money

Benefit: users can see their money and currency type on it. And both checking account and saving account share the same UI. It is not necessary to create two separate objects for them.

Relationship with UI: It is the UI that shows up when users click the

check account button. Users can deposit or withdraw their money on it.

BalanceFrame:

A class representing the UI of user balance

Purpose: to demonstrate the information for users to check their accounts.

Benefit: users can check their balance and loan in it. Two kinds of accounts are included in The same UI object. So it is more convenient for a user to check his two accounts.

Relationship with UI: it is the UI that shows up when users click the balance/loan button. Users can check their money of both accounts in it.

BankSystem:

A class representing the bank system

Purpose: to control the flow of the bank system

Benefit: creating a single object of class BankSystem can make the code more understandable and readable. Complex logical code don't have to be put into it. It just need to load the data file and produce a welcome object introducing users to log in.

Relationship with UI: it creates the UI of welcome frame to help users to log in or sign up.

CheckInput:

An interface helping the check users' input

Purpose: to assist each UI object which needs input to check users' input

Benefit: many UI objects all need users to input some information. So checking whether their input is appropriate is very important which makes the system more robust.

Relationship with UI: UI class can implement the interface to check users' input when necessary.

DataModify:

A class helping to modify the data in the data file

Pupose: to assist users to modify their data and information in the file.

Benefit: both users and managers need to modify some data when they are using the system. So this class in very useful for them to modify their data and information in the data file.

Relationship with UI: UI objects can leverage this class to modify data and information in the data file.

EnrolledFrame:

A class representing the overall UI when users log in

Purpose: to demonstrate to users all operations they can make

Benefit: users options are listed as buttons in this UI object. It is very

clear for users to choose the operation they want to make just by clicking one of the buttons. And the names of buttons are understandable.
Relationship with UI: it is the UI that shows up after users log in which help users choose their options.

GetData:

A class helping to retrieve data from the data file

Purpose: to assist UI objects to acquire data from data files

Benefit: many UI objects need get data from data files. So this class can help them to acquire data. We don't need implement every time when objects need retrieve data and just creating a GetData object is enough.

Relationship with UI: UI objects can take advantage this class to acquire data from data files.

GetDate:

A class helping to changing date and creating new files according to date

Purpose: to get current date and determine whether a file created according a previous date has existed.

Benefit: sometimes the date has changed when reporting the logging. So it is necessary to create a new log file to record the new operations taking place in a new day. This class can get the current date and determine whether a file has already existed in case that some files are

created repeatedly.

Relationship with UI: UI objects can use this class to check current date and whether a file has existed so that they can decide whether to create a new file to restore the log information.

HandleEvent:

A class helping to make decisions

Purpose: to assist users to decide or cancel their options

Benefit: many operations are risk so that users have to be sure what their are doing. It is important for the system to ask the users whether they want to make this operation to lower the possibility that users make mistakes.

Relationship with UI: it is the UI that shows up when users make some operations to help users to decide whether they really want to do it.

LoanFrame:

A class representing the UI of loan

Purpose: to help users to borrow money from the bank

Benefit: users will need to borrow some money from the bank when they are short of balance. This UI class can help them make the loan

Relationship with UI: it is the UI that shows up when users click the Loan button to borrow money from the bank

Main:

A class that starts the bank system

Purpose: to create the bank system object to start it

Benefit: the main function is in it so it is used to created the bank system object. There are just two line codes in it which makes it very understandable and readable. And complex logical code is not put into it which also makes the design extendable.

Relationship with UI: no relationship

ManagerCheckCustomerFrame:

A class representing the UI of checking customers

Purpose: to help manager check his customers' account

Benefit: managers are responsible to manage all customers. They need to check all customers' status to make sure they are in the right condition. This class can help managers to search a specific customer for his transaction record and also assist them to check all customers' account including all their information.

Relationship with UI: It is the UI that shows up when managers click the check a customer button.

ManagerDailyReportFrame:

A class representing the UI of daily report for a manager

Purpose: to demonstrate transaction reports to managers

Benefit: managers should check everyday transaction report to get to know what all customers are doing. And managers also want to search for transaction report of a specific day. This class can perfectly help managers to finish these two jobs

Relationship with UI: This class is the UI that shows up when managers click the Get Daily Report button.

ManagerEditMemberFrame:

A class representing the UI for managing the manager accounts

Purpose: to manage(add or delete) a manager's account for managers

Benefit: sometimes it is needed for managers to add another manager account or delete one of current manager accounts. This class can help managers to do these jobs. Every manager has the right to check check all manager accounts and add one account or delete one which is not used any more.

Relationship with UI: It is the UI that shows up when managers click the Add/Delete managers button. It is used for managing the managers accounts.

ManagerLoginFrame:

A class representing the UI of managers' logging in

Purpose: to help managers log in

Benefit: There are two kinds of users of the bank system: ordinary users and managers. So it is necessary to separate the two kinds of users. Manager can click manager special log in button to enter the entrance for managers. After entering username and password, this class can check whether this account is actually a manager's account according to the manager account info file.

Relationship with UI: it is the UI that shows up when users click the Manager click here button.

ManagerSelectOperationFrame:

A class representing the overall operations that managers can make

Purpose: to demonstrate all options of managers after logging in

Benefit: managers can make many operations when they log in. They should choose one to process each time. This object lists all operations managers can make as buttons which are very readable for users. And it is also extendable if there are more operations needed to add to the system in the future.

Relationship with UI: it is the UI that shows up after managers log in according to which managers can choose one operation to do.

PayLoanFrame:

A class representing the UI of paying off the loan

Purpose: users can pay back their loan to the bank through it

Benefit: After users borrow money from the bank. They have to pay off the loan sometime until all loan has been paid off. This class could help users to pay the loan. They only need to click the button and enter the money they want to pay. The system will automatically check the amount for them.

Relationship with UI: it is the UI that shows up when users click the Payloan button.

ReadData:

An interface helping to read data from files

Purpose: to assist UI objects to read data from data files

Benefit: many UI objects need to acquire money and loan of users. This interface can help them to get the data in an easy way. So there is no need to write the read function every time when we want to read currency from files. Implementing this interface is enough for this work.

Relationship with UI: UI class can implement this interface to possess the ability to read money and loan from data files.

SignOnFrame:

A class representing the UI of signing up

Purpose: to help users to create accounts

Benefit: if a user don't have an account before, he has to sign up for a new account. Users can click the sign up button on the welcome UI. They should enter the personal information and the type of account they want to create. And additionally, they can deposit some money in the mean time they create the account. The UI can check for them whether this account has already existed and their input is right or not.

Relationship with UI: it is the UI that shows up when users click the sign up button which can help them to create an account.

TransactionFrame:

A class representing the UI of transaction records

Purpose: to demonstrate the transaction records to users

Benefit: users want to inspect their transaction records sometime. This UI object can help them to check their current transaction. They can get details of every transaction such as time, operations and amount which can make it convenient and secure for users to manage their accounts.

Relationship with UI: it is the UI that shows up when users click the Transaction button in the account UI.

TransferFrame:

A class representing the UI of transferring

Purpose: to help users transfer their money to another account

Benefit: Transferring money from an account to another account. Users can leverage this UI object to finish this job. They only need to enter the account they want to transfer the money to and the amount of money. The UI will check the amount and compare it to the balance and it will also make sure the transfer process is done in a secure way.

Relationship with UI: it is the UI that shows up when users click the Transfer button in the account UI.

WelcomeFrame:

A class representing the UI of logging in for ordinary users

Purpose: to welcome users and help them to log in to the bank system

Benefit: it is meant for ordinary users to log in. Users need to enter their username and password into the text field. The log in system will automatically check whether this account is in the system. If the account is legal, then users can enter the account UI.

Relationship with UI: it is the welcome UI when users launch the system. Users can enter the account UI after entering the right username and password.

TransferCheckingFrame:

A class representing the UI of transferring the money between two accounts

Purpose: to assist users in transferring the money from the saving account to checking account

Benefit: The case is very often when one account of a user is short of money but the other account has enough money. Therefore, users can leverage this UI object to transfer their money from saving account to their checking accounting to compensate the shortness of money in one of their accounts. And the UI will automatically check whether this transferring is legal as well as make sure it is done in a secure way.

Relationship with UI: it is the UI that shows up when users click the Transfer to checking button in the saving account UI.

WithdrawDepositFrame:

A class representing the UI of withdrawing and depositing money

Purpose: to assist users in depositing money into the account or withdrawing some money from it

Benefit: it is the main function of the bank system. Users can deposit or withdraw their money in either on of the accounts. They just need to enter the amount of money they want to deposit or withdraw. This system will automatically check whether their balance is enough for the withdrawal process when they choose to get some money from it.

Relationship with UI: it is the UI that shows up when users click the Deposit/Withdraw button in one of the accounts.

WriteData:

A class helping to write data into files

Purpose: to equip some UI objects the ability to write data to files

Benefit: In many cases, the UI object should write some data and transaction information into data files. This class could help UI finish these jobs. When there is need to write some data, the objects can easily call the static functions in this class to write the data into data files in a consistent and readable way. We don't need to implement writing process code every time when we want to restore some information. Calling this class is enough.

Relationship with UI: UI can call static functions to write data into files in an convenient way.

Design Pattern:

MVC Pattern (Model-View-Controller):

Model: utilitarian classes such as GetData, CheckInput, ReadData. These classes can help to store data and update controller.

View: all frame classes such as AccountFrame, TransferFrame. These classes are used to generate friendly user interfaces which make the

bank system more convenient to use.

Controller: BankSystem class is the controller which controls the whole work flow of the bank system.

Benefit: MVC design pattern can separate the whole system into three main module: model, view, controller which lower greatly the coupling of the system. We could separately modify,add or delete some features of one of the three parts with doing harm to other two parts. It makes the bank system more scalable and extendable.