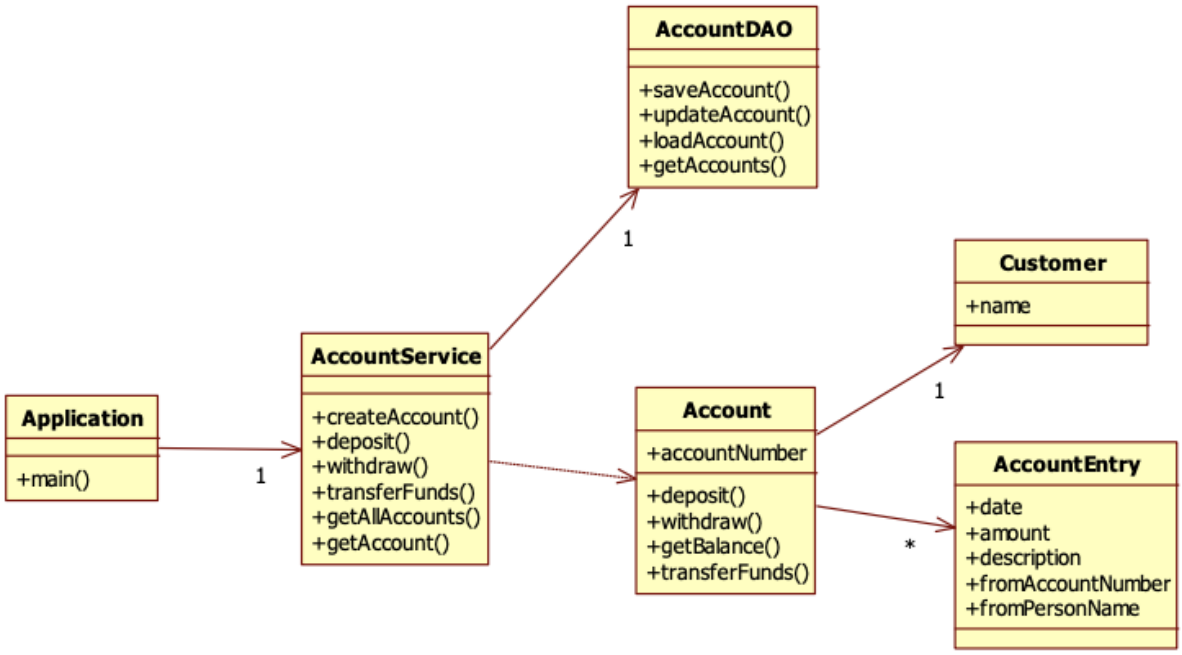
Question 1 [65 points] {70 minutes}

In a number of design pattern labs in this course, we have applied different patterns to the following given bank application:

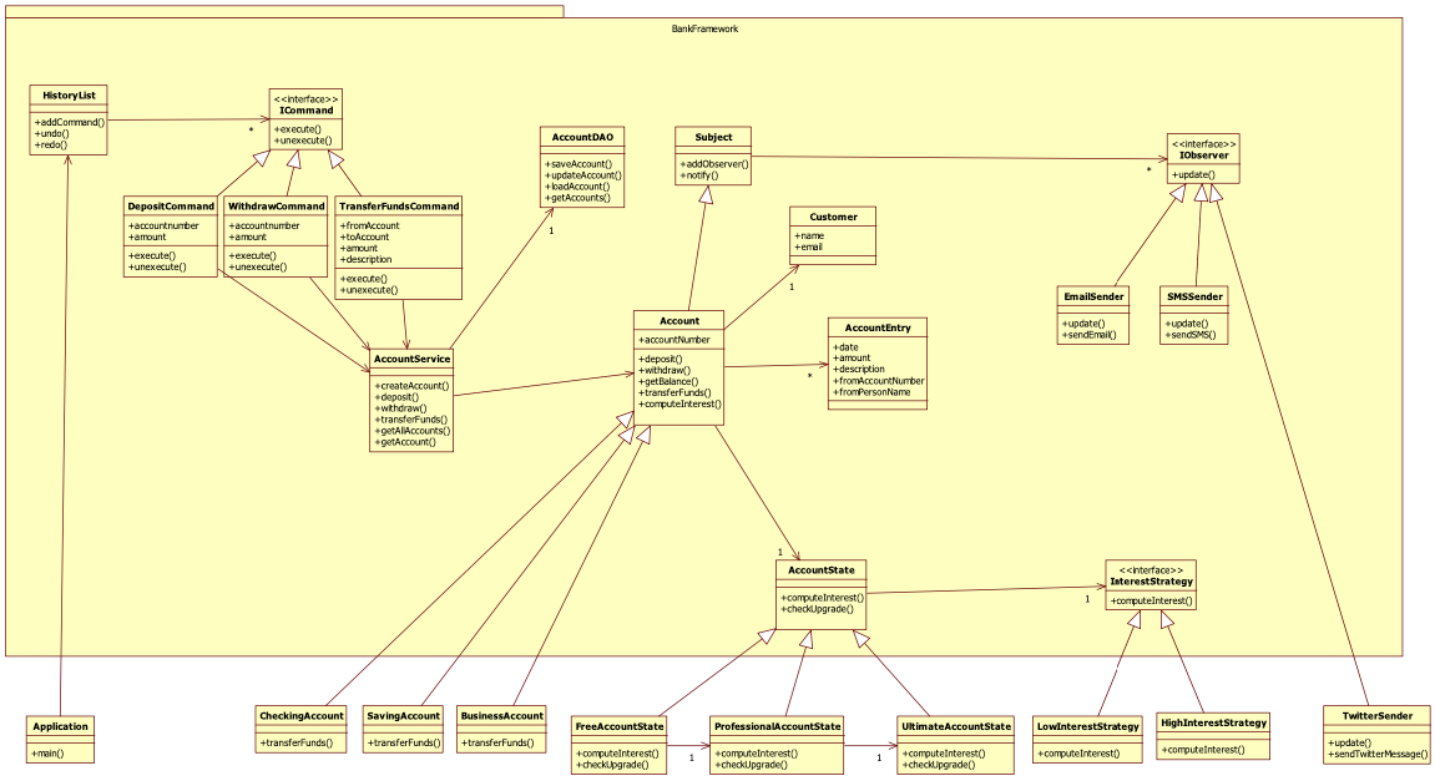


Design a Bank framework with the following requirements:

- The framework should support undo/redo functionality for the methods deposit(), withdraw() and transferFunds().
- The application that uses the framework supports checking, savings and business accounts. The business rules for transferFunds() are different for all 3 accounts.
- The customer can subscribe to the service that allows the user to be notified whenever the account balance changes. The framework should support Email and SMS notifications. The application that uses the framework should also support Twitter notifications.
- The framework should support 2 ways (simple and complex) to calculate the interest for every type of Account.
    - A simple way of calculating the interest is that for example CheckingAccounts get a low interest based on the account balance and SavingAccounts get a high interest based on the account balance.
    - A complex way of calculating the interest is based on the Account state. Accounts can have different states like a basic account, a professional account or an ultimate account. Based on some logic, customers can be upgraded from a basic account to a professional account, or from a professional account to an ultimate account.
        - If you have a basic account, your get the regular interest that is calculated as the simple way of calculating the interest described above.
        - If you have a professional account, you get the regular interest, plus some extra interest (let's say 1% extra).
        - If you have an ultimate account, you get the regular interest, plus some extra interest (let's say 2% extra).

The application that uses the framework should support basic accounts, professional accounts and ultimate accounts. It should also support 2 different ways of calculating the regular interest (low and high).

Draw a **class diagram** of the design of the message forum framework. **Make sure you add all necessary UML elements (attributes, methods, multiplicity, etc) to communicate the important parts of your design.**

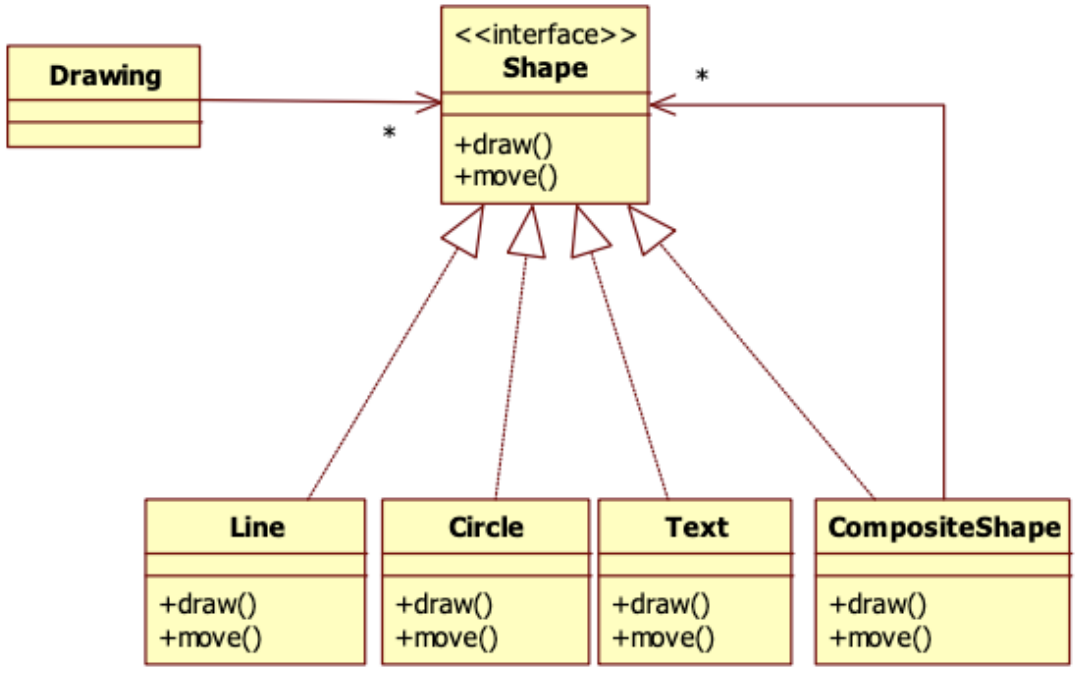

BankFramework
HistoryList
+addCommand()
+undo()
+redo()
<<interface>>
ICommand
+execute()
+unexecute()
DepositCommand
+accountnumber
+amount
+execute()
+unexecute()
WithdrawCommand
+accountnumber
+amount
+execute()
+unexecute()
TransferFundsCommand
+fromAccount
+toAccount
+amount
+description
+execute()
+unexecute()
AccountDAO
+saveAccount()
+updateAccount()
+loadAccount()
+getAccounts()
Subject
+addObserver()
+notify()
<<interface>>
IObserver
+update()
Customer
+name
+email
EmailSender
+update()
+sendEmail()
SMSSender
+update()
+sendSMS()
AccountService
+createAccount()
+deposit()
+withdraw()
+transferFunds()
+getAllAccounts()
+getAccount()
Account
+accountNumber
+deposit()
+withdraw()
+getBalance()
+transferFunds()
+computeInterest()
AccountEntry
+date
+amount
+description
+fromAccountNumber
+fromPersonName
AccountState
+computeInterest()
+checkUpgrade()
<<interface>>
InterestStrategy
+computeInterest()
Application
+main()
CheckingAccount
+transferFunds()
SavingAccount
+transferFunds()
BusinessAccount
+transferFunds()
FreeAccountState
+computeInterest()
+checkUpgrade()
ProfessionalAccountState
+computeInterest()
+checkUpgrade()
UltimateAccountState
+computeInterest()
+checkUpgrade()
LowInterestStrategy
+computeInterest()
HighInterestStrategy
+computeInterest()
TwitterSender
+update()
+sendTwitterMessage()

Question 2 [15 points] {20 minutes}

Suppose we need to design a drawing framework which allows us to develop drawing applications. Consider the following subset of requirements for this framework:

- The framework should support drawing simple shapes like lines, circles and text.
- We should also be able to move a shape to a different position
- The framework should support grouping and ungrouping of shapes. So we can group a number of shapes together, and then handle this group as one shape. We could for example move this grouped shape to another position on the drawing.

Draw the class diagram that shows how your design works. Do NOT draw the whole class diagram of the drawing framework, but only the class diagram that shows how your design implements the requirements given above. Make sure your class diagram contains all the important information to communicate your design.



Question 3 [15 points] {20 minutes}

Suppose we need to design a game framework which allows us to develop different games. All the games we write contain different levels. The points you receive during the game is a complex formula based on the current level and on the current number of points. The games we develop with this framework can contain any number of levels and different formulas to compute points you receive during the game.

Draw the class diagram that shows how your design works. Do NOT draw the whole class diagram of the game framework, but only the class diagram that shows how your design implements the requirements given above. Make sure your class diagram contains all the important information to communicate your design.