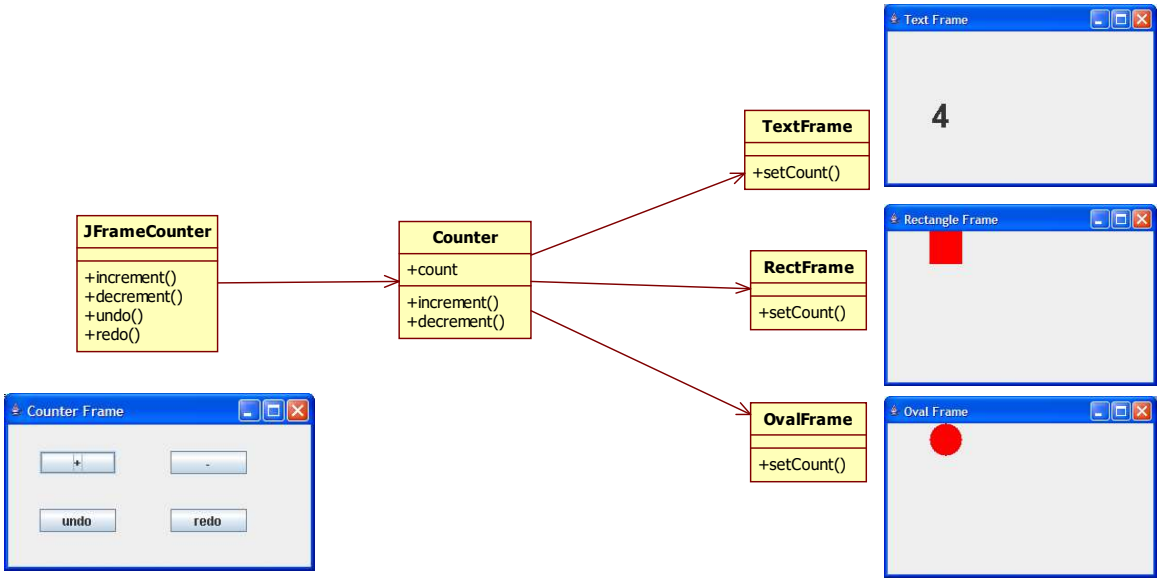
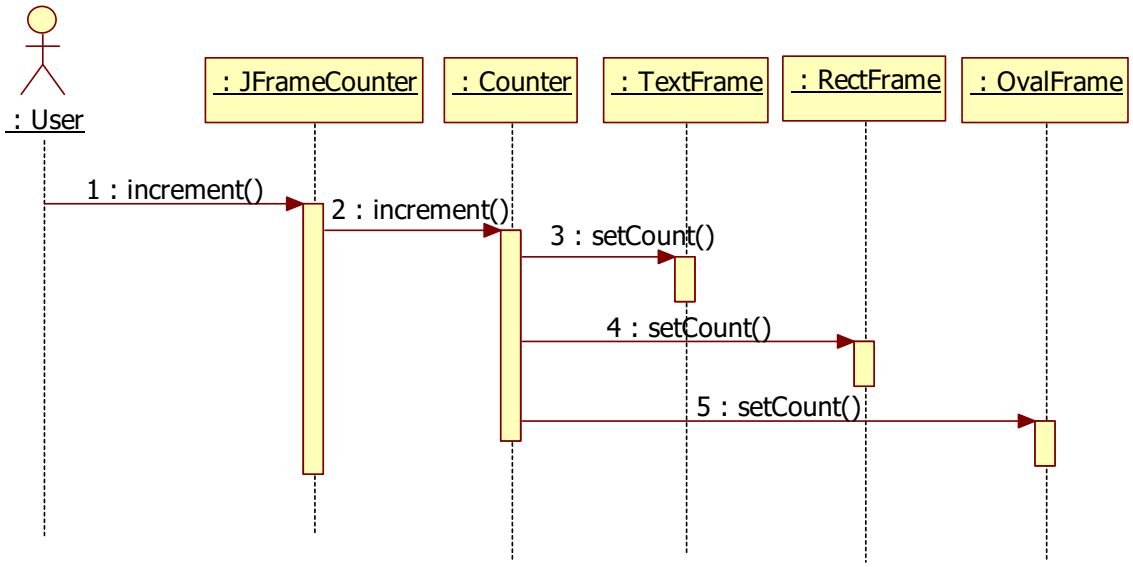
Lab 3

a. Given is the following counter application:





The undo and redo methods do not work yet. We will implement them in a later lab.

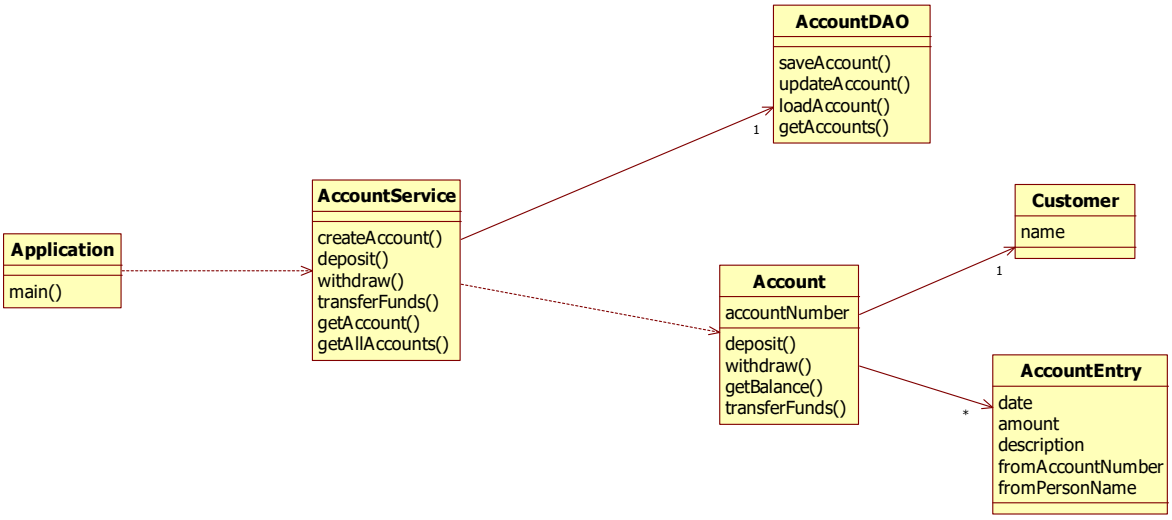
The problem with the given application is that the Counter class is tightly coupled with the UI classes TextFrame, RectFrame and OvalFrame. If we want to add another view of the counter, for example a binaryFrame that shows the value of the counter in binary, then we have to change the increment() and decrement() method in the Counter.

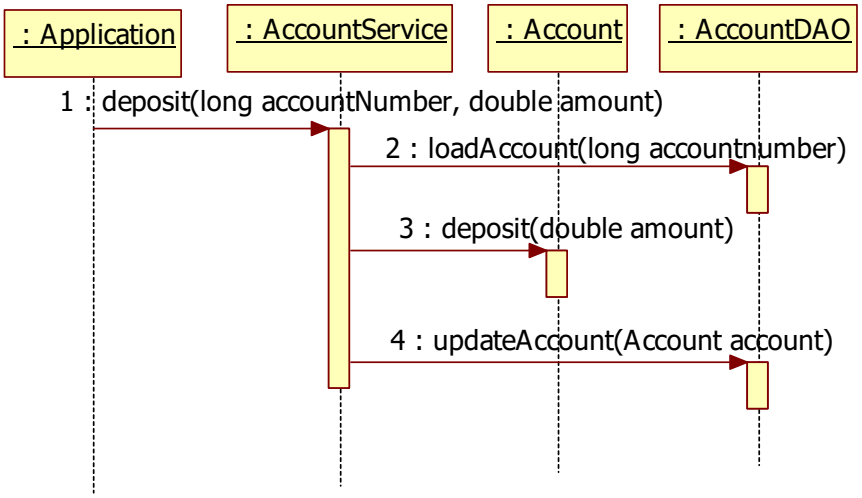
Draw the class diagram of a better design so that it will be much easier to add different views of the counter value.

b. Draw the sequence diagram that shows the following scenario:
   1. The user clicks the increment button
   2. The user clicks the decrement button

c. Implement your new design in the given code in Java.

d. Given is the following bank application:





We want to add new functionality whenever the Account balance is changed. Implement the observer pattern in the given code. Add the following observers:

- Add a Logger class that logs every change to an Account. (The Logger should do a simple System.out.println() to the console)
- Add a SMSSender that sends a SMS at every change to an Account. (The SMSSender should do a simple System.out.println() to the console)
- Add an EmailSender that sends an email whenever a new Account is created. (The EmailSender should do a simple System.out.println() to the console)

e. Draw the modified class diagram with the observer pattern applied.
f. Draw a sequence diagram that shows how your new design works. On the sequence diagram show the following scenario:
    1. First create a new account
    2. Then deposit $80 on this new account

g. Implement the observer pattern in the given code.

d.