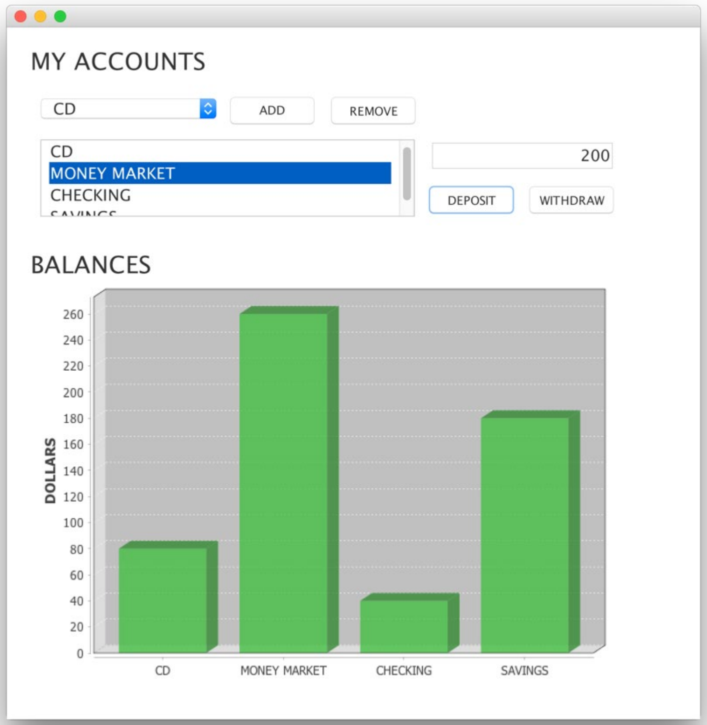
PERSONAL ACCOUNT MANAGER DESIGN
In-Class Activity

Full Names of Team Members: ______________________________________

Activity Description

Create a UML class diagram design for your Personal Account Manager application. The program description is given below.

Create a personal bank account manager program. The figure on the right gives an indication of what the program needs to be able to do. Here are the formal requirements:



FACTORY

- The program must use a simple factory that produces different types of accounts.
- Optional: New account types can be added to the program without requiring recompilation. You can use the DrawShapeFactory as an example.
- Clients must be able to query this factory to get a list of account types that it can produce.

ACCOUNTS (THE SUBJECTS)

- Must have a balance as well as deposit and withdraw methods.
- Must be observable. Implement your own "Subject" code (attach, detach, inform). Implement at least checking, savings, and CD account types. For this program, it is fine if they all do the same thing (i.e. don't worry about interest or compounding or other activities).

MAINWINDOW (THE OBSERVER)

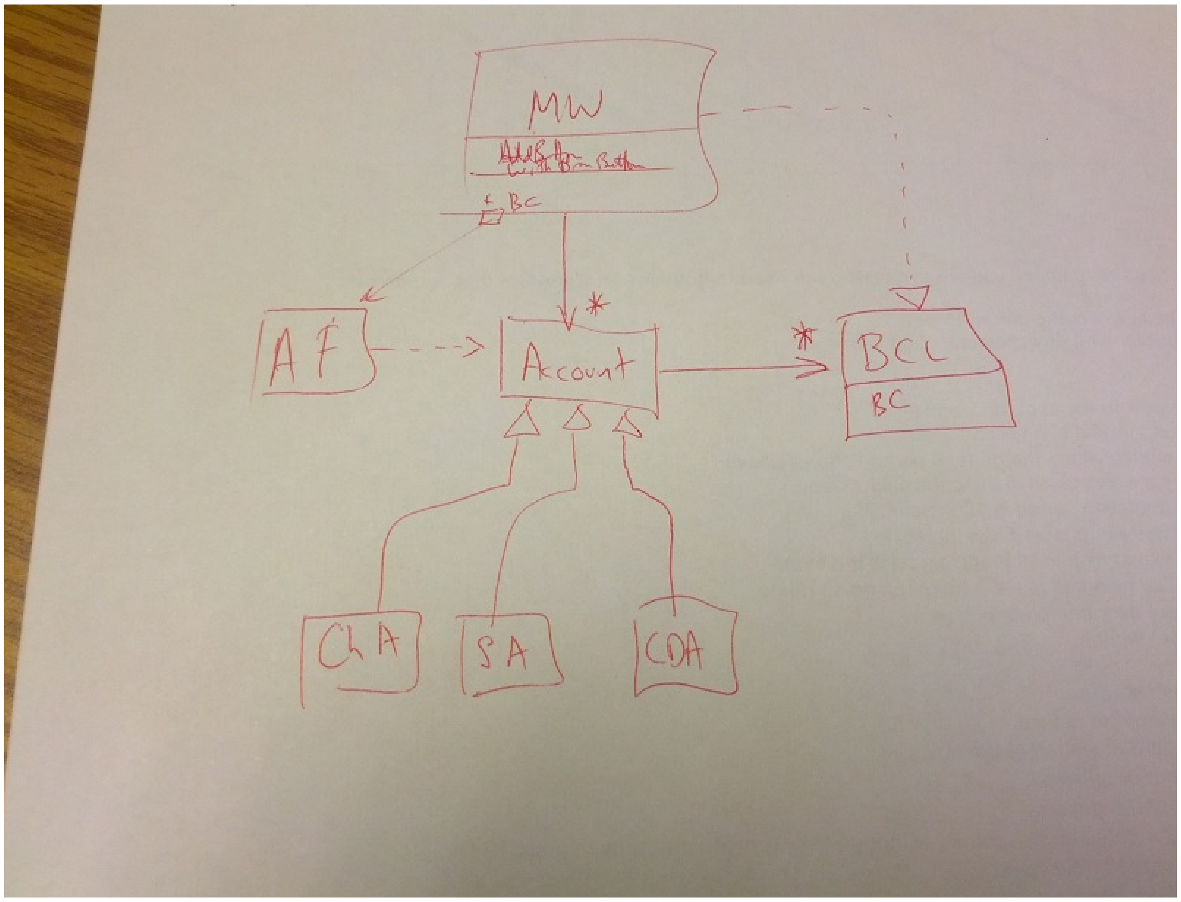
- Must allow the user to add and remove accounts.

- Must display a bar chart with account types on one axis and balance on the other. There are a number of free charting programs out there, including http://www.jfree.org/jfreechart/.
- Must allow the user to deposit into or withdraw from any account. The deposit and withdraw buttons should call the deposit/withdraw methods in the Account class, but must not update the bar chart. Updating the bar chart must be handled via the Observer pattern (see below).
- The window should attach to each account when it is created. When an account is balance is changed, the account should inform its observers (in this case just the MainWindow) that its balance has changed. Update the bar chart in the update() method. This approach frees the button handlers from having to know about all of the views that might be affected by a deposit or withdrawal. When an account is removed, the MainWindow must detach from the Account.

NAMING CONSIDERATIONS

You *can* use the generic Observer pattern method names: **`attach`**(), **`detach`**(), **`inform`**(), and **`update`**(), but that leads to some less-than-readable code. An example of a better set of names would be **`addBalanceChangedListener`**(), **`removeBalanceChangedListener`**(), **`fireBalanceChanged`**(), and **`balanceChanged`**().

PARTIAL ANSWER: doesn't show methods


MW
+ BC
*
AF
Account
*
BCL
BC
ChA
SA
CDA