Roles:

Txt files (deposits, outgoing, alerts, README) - **Angela**

* Implement read lines + outputs (alerts.txt + outgoing.txt)
* Testing files

1st Parent class (abstract) - **Jackson (will do around half) + 2 more people**

* Differentiate user / manager by format of usernames
* If bank manager: capital BM + nums
* If user: capital USER + nums
* Time (static variable):
  + The bank manager should only have to set the date once. You can assume that the program is shut down and restarted every night at midnight. When the user asks to see their account balance, it should be up-to-date as of the previous day.
    - // when shut down method is called, output the current time
    - // when program is started, read said time and set the time
  + If you can make it completely up-to-date, that is even better! (but optional).
  + If you want to include time stamps, you can do this any way you want; through the input file(s), using the system clock, or any other way.
* Handling cash (static variable):
  + The bank machine should know how many $5, $10, $20, and $50 bills it has. When the amount of any denomination goes below 20, your program should send an alert to a file called alerts.txt that the real-life manager would read and handle by restocking the machine. When a user requests a withdrawal, your program will have to decide which bills to give to the user and decrease the total of those denominations accordingly.
* Bank manager (final instance variable):
  + The bank manager is the only person who should be able to create a login and set the initial password for a user
  + Also, they can increase the number of $5, $10, $20, and/or $50 bills in the machine to simulate restocking the machine
  + Lastly, the manager has the ability to undo the most recent transaction on any asset or debt account, except for paying bills.
* user info: users should have the option to see on the screen:
  + a summary of all their account balances
  + the most recent transaction on any account
  + the date of creation of one of their accounts
  + their net total (The total of their debt account balances subtracted from the total of their asset account balances.)
* users can interact with their accounts by:
  + viewing their account balance(s)
  + transfer money between accounts that they own
  + withdraw money from an account (This will decrease their balance.)
  + transfer money from their account to another user's account (This will also decrease their balance.)
  + pay a bill by transferring money out to a non-user's account (This can be stored in an **outgoing.txt** file that is outside of the program. It also decreases their balance.)
  + deposit money into their account by entering a cheque or cash into the machine (This will be simulated by individual lines in an input file called **deposits.txt**. You can decide the format of the file. This will increase their balance.)
  + requesting the creation of an account from the bank manager

2nd parents Accounts (class) - accounts class extends parent class -**Maple + 1 more person**

* 4 subclasses
* Credit card accounts
* Line of credit accounts
* Chequing account
* Savings accounts

digital version of UML diagrams ?