Class list (might not be 100% complete)

* Connection Manager – for connecting to db
* Repository Interface – for implementing in repository classes
* Once repository class for each POJO (AKA Model or Entity)
* One POJO (AKA model or Entity) for each database table, so probably 2
* Console Renderer
* One View class for each menu the user might interact with (main menu, login, register, withdraw, …)
* Custom List Interface – for the inheritance
* Custom List Class – inheriting from interface
* Maybe need a data store class to hold data

Project 0 steps to complete:

* Custom list
  + interface – done, copy paste from assignments in notes repo
  + choose array or linked and implement the 7 methods.
    - (unless we don't need a method, then forget it)
* JDBC persistence
  + start with the connection manager - conn string, properties file, and establishing the connection to the db. Just make sure you do the connection, and don’t get an exception.
  + Move on to the simplest repo. Repo is the design pattern we went over Wednesday for persistence. The simplest repo is the one that corresponds to the simplest POJO.
    - You may even make up a super simple POJO just for this. Maybe a POJO with just an id, and a single string just for practice. Then move on to the real ones.
  + Aim to complete all 4 CRUD for one POJO/Repo, and then simply repeat this process. 80% or more of the code will be the same.
  + The easiest way to do POJO CRUD is to do what I did in class Wednesday, manipulate, store, and retrieve the whole POJO at once. So for instance if you need to withdraw money from an account:
    - Query the database for that account by calling Repository methods, and put those results into an Account POJO.
    - Make the change to the balance in java memory by using the Account POJO setter method.
    - Then save/update that POJO by calling the Repository’s methods.
  + We may also need a few other queries, for instance one to get all accounts for a user…
* Console I/O
  + Steal the renderer class from Kyle, definitely comb through the code to understand
  + Build your Views. Most of your interaction with user via the console will happen here within the render() method.
  + Store your data somewhere, maybe in a data store class of some kind.
  + Make sure to register each view with the renderer in main method.
* Put a loop in the main method that runs as long as the renderer field “running” remains true.

