Here is the Assignment:

Even though the election is over, we’d like to get an idea of how a voting program might work (if your program is good, it can always be used next year!).   In this program you will implement a simple yet flexible “voting machine” with the following abilities:

1)    It can accept an arbitrary number of offices, each with an arbitrary number of candidates, and allow users to vote for a candidate for each office.

2)    It allows users to vote only if they are “registered” and allows them to vote only one time.

3)    It keeps track of the number of votes for each candidate in each office in a well-organized way, by associating a file name with each office on the ballot.

4)    It updates all data files (vote count files and voter file) as soon as possible and in a way that (mostly) protects the data if the program crashes during the update.

**Execution / Implementation Details:**

**Your voting program should have a similar look, feel and functionality to the one demonstrated in the snapshots file A4snap.html.**  Look over the A4snap.html file in detail to see how your program should handle various situations.  **I have some additional comments in the A4snap.html file, so read it over very carefully.**  Below are some general guidelines:

·    All interaction with the user should be handled graphically, using JButtons (within JPanels) and JOptionPanes.  **No console I/O should be done** (except for debugging purposes).

·    **The buttons on the window should be presented in a nice, easily viewable layout.  Each ballot must be displayed in its own JPanel and the data for each ballot MUST be encapsulated within its JPanel (so you must extend the JPanel class).  In other words, you should create a class called Ballot which extends JPanel, and the user will see a Ballot in the window for each office.   Since it is not known in advance how many offices there will be, your main program should have an array of Ballot, which will be sized using the ballot file.  See Counters2.java for an example of encapsulating data within a JPanel.**

·    The name of the ballot file can be arbitrary and should be read in from the command line when the program is started.  For details on reading from the command line, see Section 8.12 in the Gaddis text.

·    A button should only be enabled when appropriate during the execution of your program.  For example, a user should not be able to vote until he/she has logged in. See the setEnabled() method for JButtons.

·    Files should be formatted as shown in A4snap.html.  Note the separator characters in each line of each file.  For information on how to parse these files, see the split() method in the String class and course handouts.

·    After a user votes, the files (both for the offices and for the users) should be updated immediately in a “safe” way.  You should do this in the following way:

·      Read in the original file line by line

·      If a line needs to be modified (ex: a vote count), do so in your program

·      Write the lines back to a new file (i.e. you are copying the file, but with the appropriate modifications).  The temporary name for this new file should be something that is not likely to be a name of a regular file.  For example: temptemp.txt or something similar should work.

·      When the new file is complete, delete the original file and rename the new file with the original file name

·    The idea is that if something bad happens while you are writing the new file, the original file will not be affected, so only the most recent vote will be lost.

·    For some help with this, see the API for the File class and its various methods