

Snapshots demonstrating functionality required for Assignment 4.

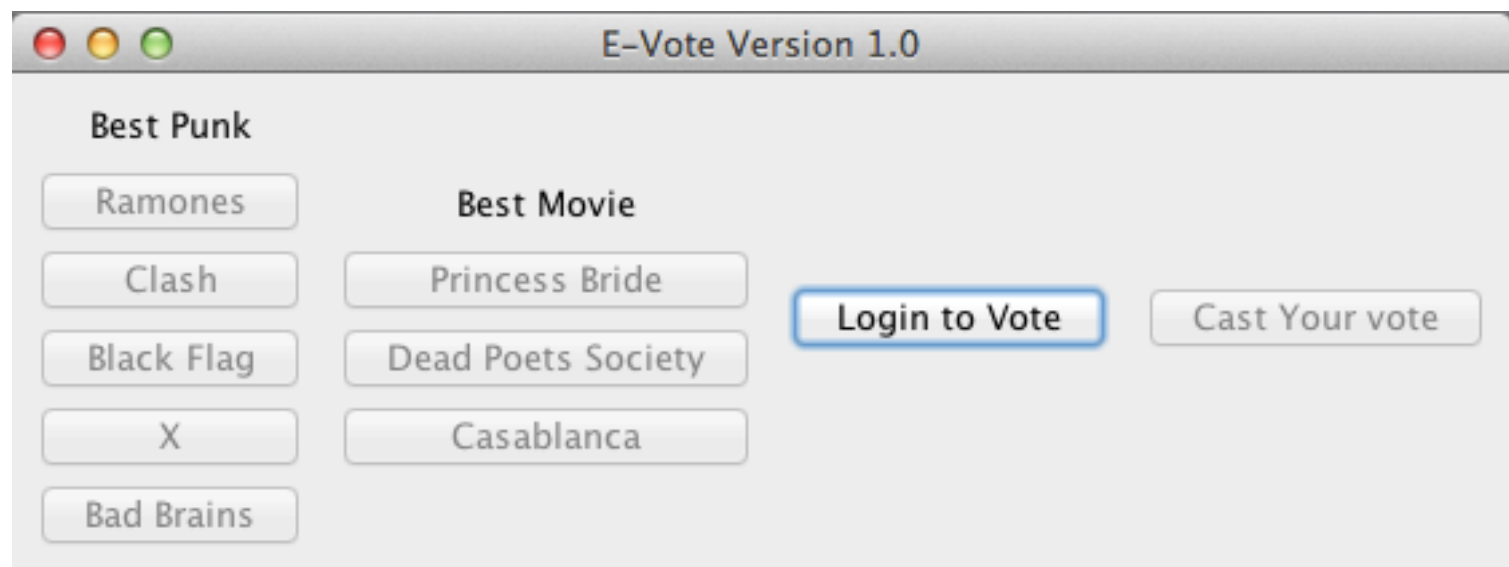
Read this document very carefully so you know what is necessary in your implementations.

Note 1: Your GUI does not have to look exactly like mine. However, it should be functionally equivalent and should be easy and intuitive to use. Look over Lab 8, Counters2.java and other class handouts for some help with this assignment.

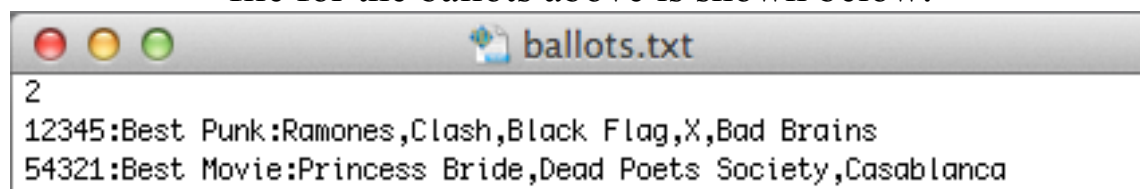
Note 2: This document focuses on the functionality of the assignment. Please see the Assignment 4 document for details on implementation requirements.

The program is initially invoked with a command line argument containing the name of the ballot that will be used. For example,

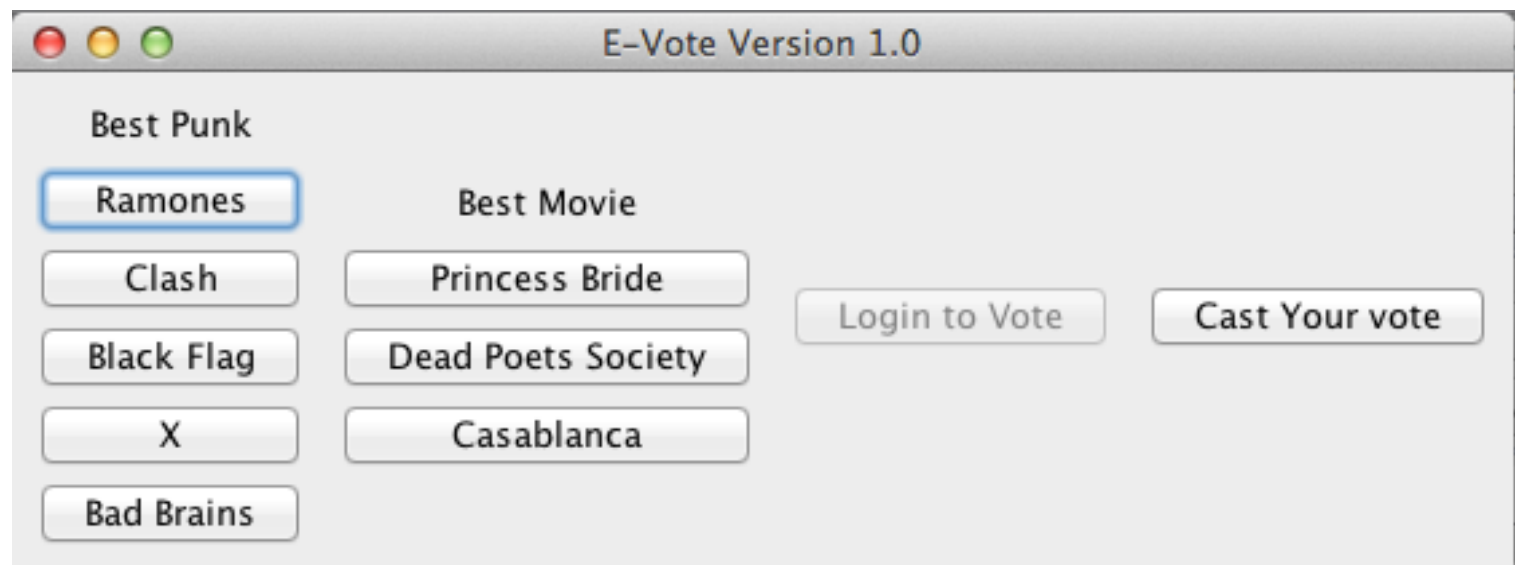
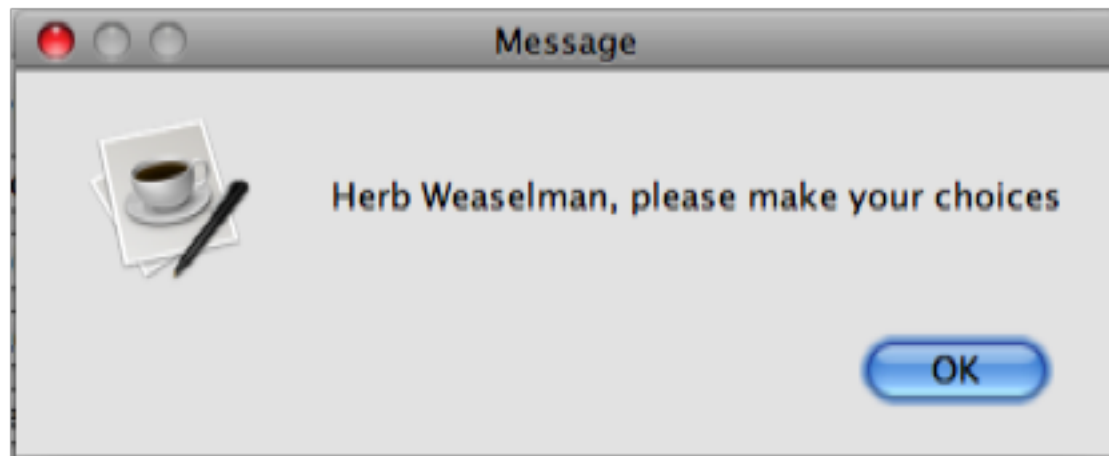
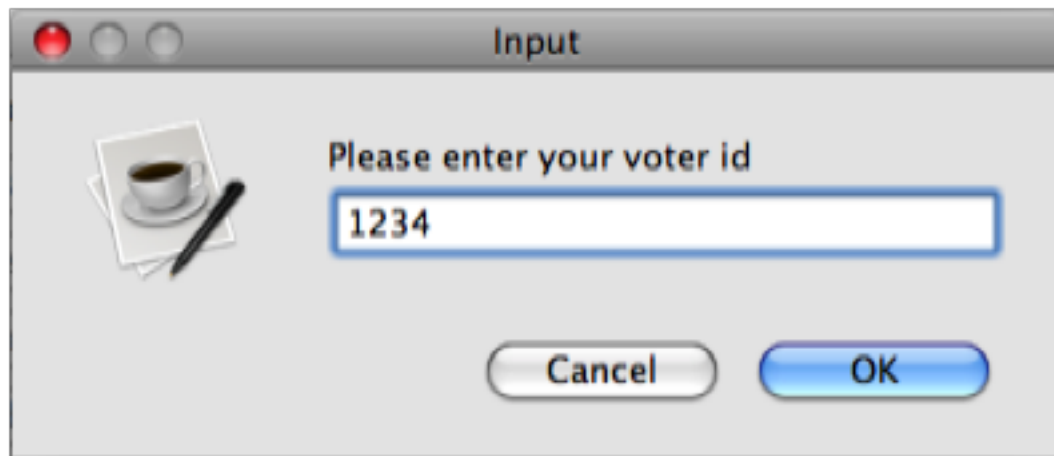
```
java Assig4 ballots.txt
```





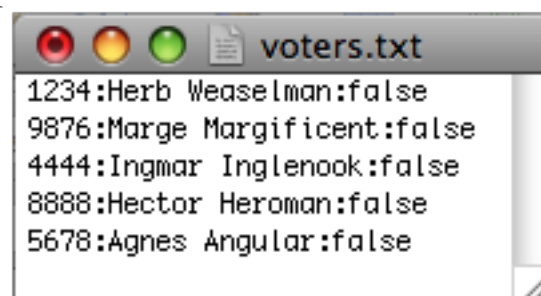
Initially the window shows each ballot in a Ballot object (which is a subclass of JPanel). Each Ballot object contains the id of the ballot (not shown to user) followed by the name of the office followed by a JButton for each candidate. After the Ballots in the window two additional JButtons are shown – one for logging in and one for casting the vote. The number of ballots and the number of candidates per ballot should be arbitrary (based on the file read in). The only JButton that is initially enabled is the Login to Vote button. The file for the ballots above is shown below:



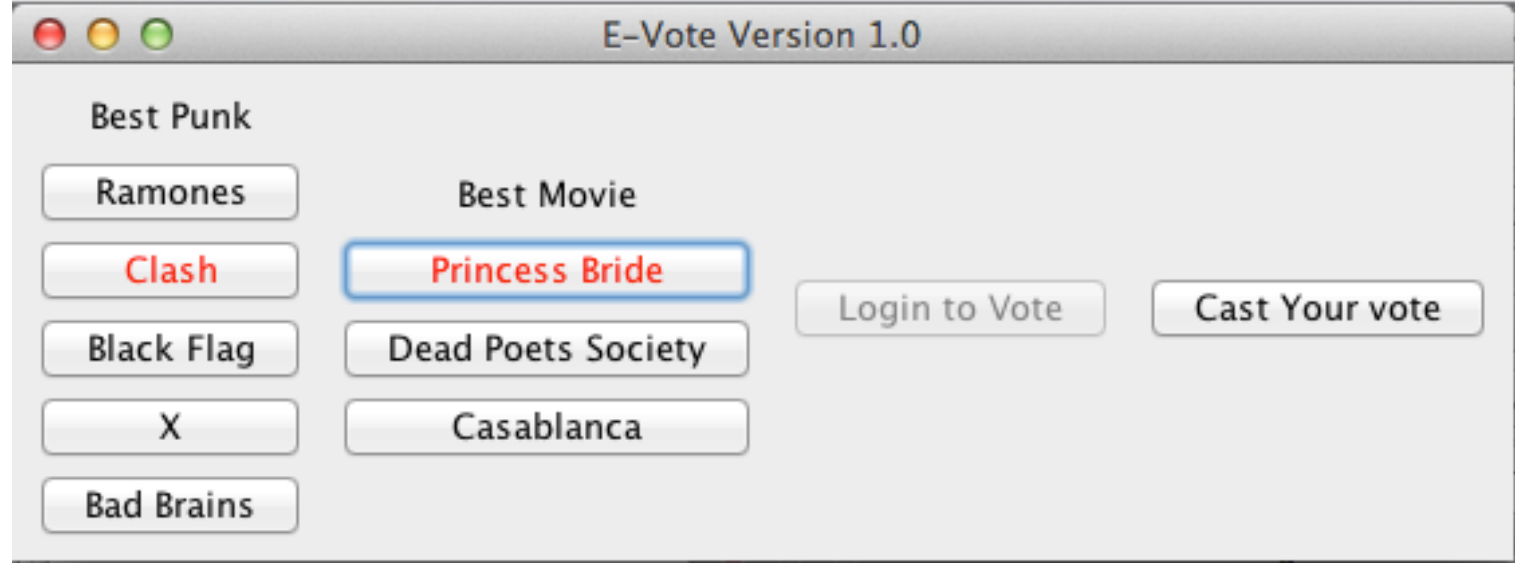
Note the format of the file – this is required. The first line contains the number of ballots in the file. For each ballot, the first field is the id for the ballot, which is not shown to the user but which must be present, since the file name to store the votes is created from that id. We may test your program with our input files, so be sure to follow the format exactly. To see how to parse this file correctly, see course handouts and the split() method in the String class.



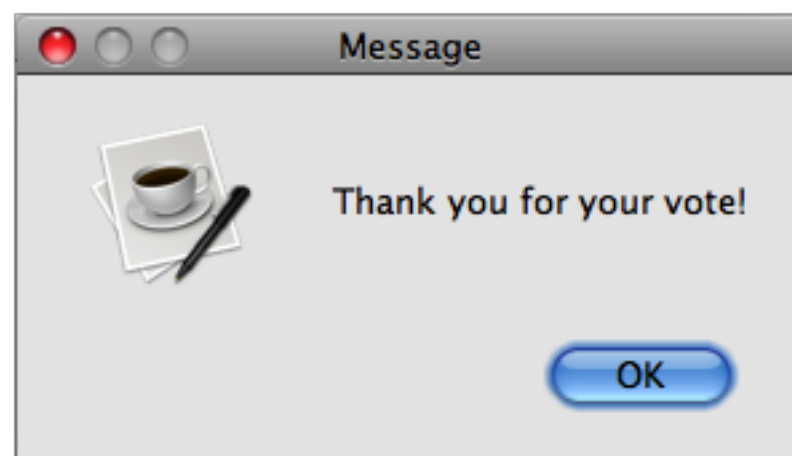
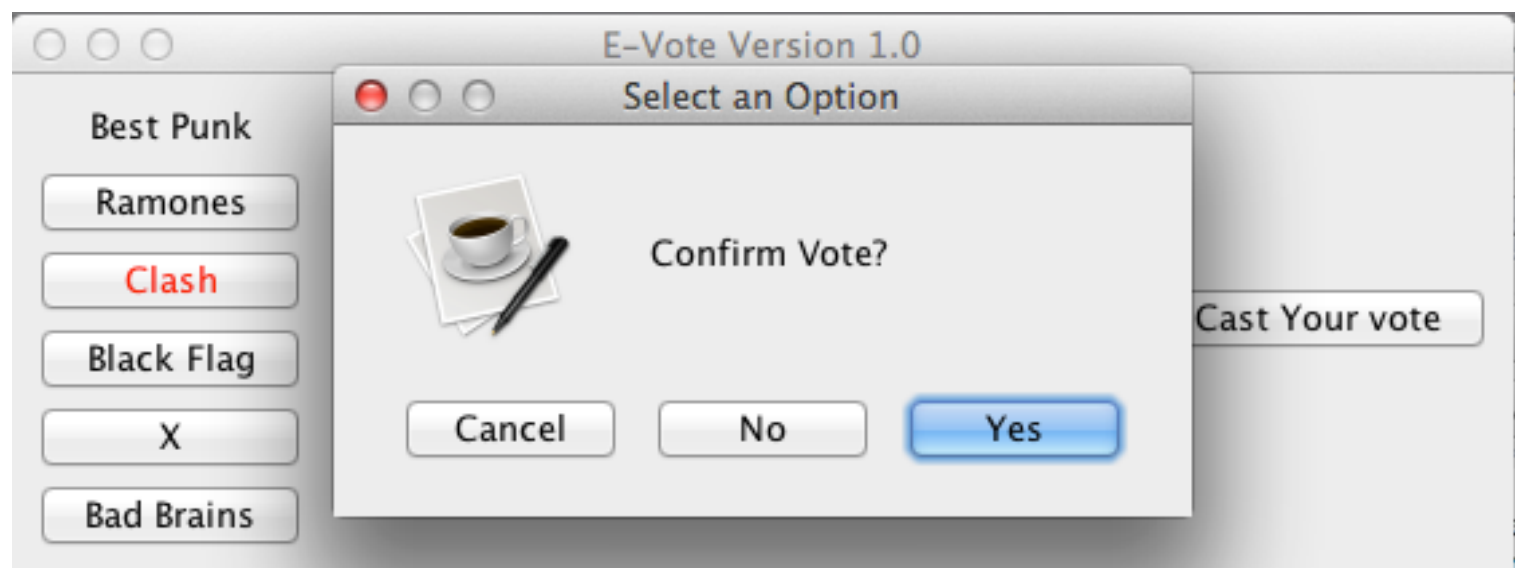
When a voter clicks on the  Login to Vote  button, he/she is presented with a dialog box asking him/her to enter his/her voter id. After this is input, your program should check to make sure the voter is registered (i.e. in the `voters.txt` file) and has not yet voted. Show the user the appropriate message. If the voter is valid, enable the buttons so the user can vote. Note above that the Login to Vote button is now disabled. An example `voters.txt` file is shown below.

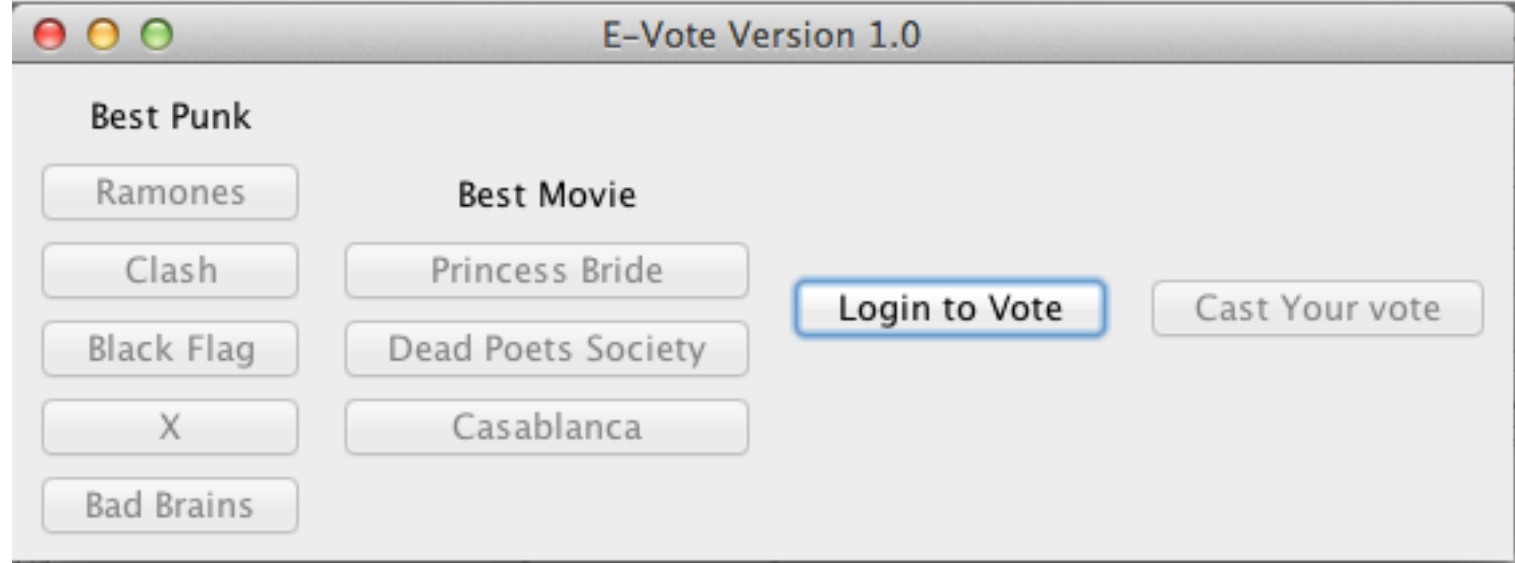


Note the format of the file. The first field is the voter id, followed by the voter name and that voter status (true if the voter has voted and false if the voter has not voted). Colons separate all of the fields.

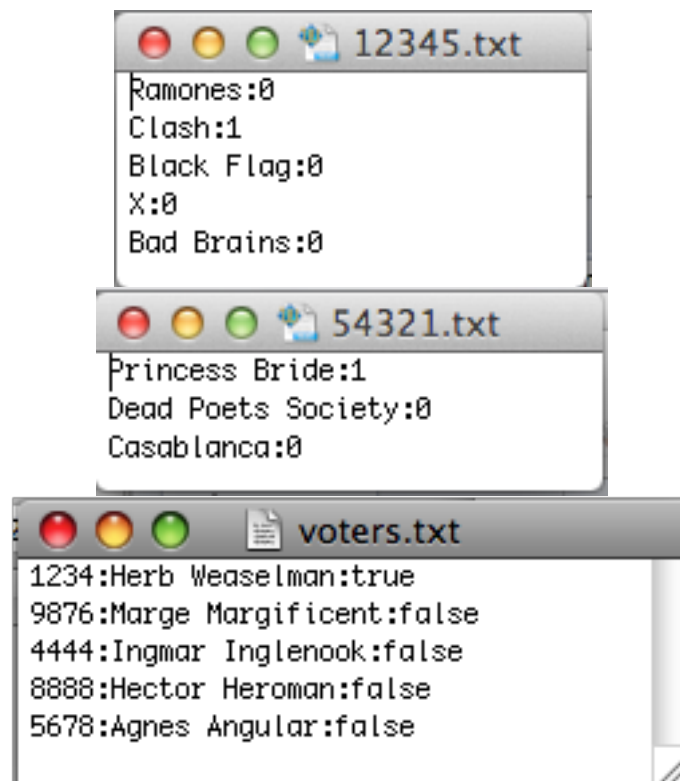


The user has voted for both offices but has not cast the ballots yet. Note that the color change indicates the voter's choices. The buttons should be toggles, so that if the user clicks a selected button it will then be unselected. This allows a voter to select someone, then change his/her mind and decide not to vote for anyone for that office. It is allowed for a voter to cast his/her ballots without selecting a candidate for one or more offices.

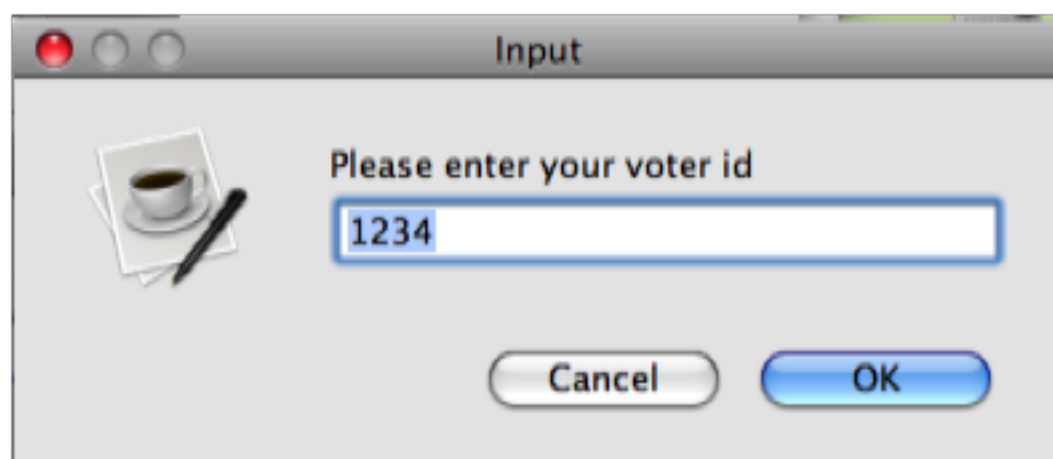


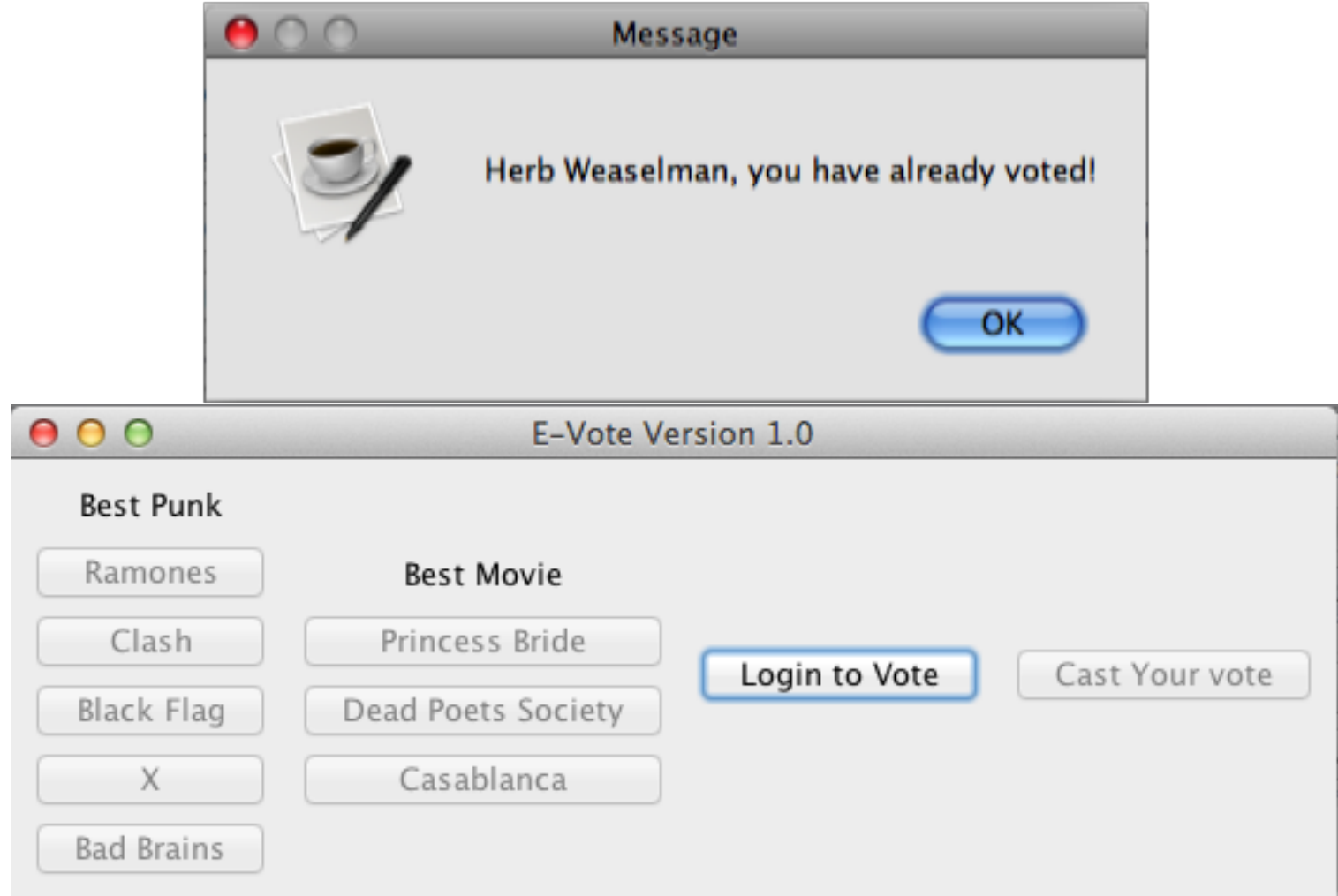


After the voter casts his/her vote, a confirmation is shown. If the voter clicks "Yes" the ballots will be cast, an acknowledgement is shown and the window reverts to its original state. Also, several files are immediately updated in a **safe** way. See the Assignment 4 sheet for an explanation of **safe** file updates. If the voter clicks "No" or "Cancel" in the confirmation dialog, the voter can resume making choices on the current ballots. For this example, the voter confirms and the updated files are shown below:



As the program runs these files continue to be updated (in the safe way described). Thus, the ballot id files will reflect the current vote counts for each ballot and the "voters.txt" file will reflect the current voting status of all voters.





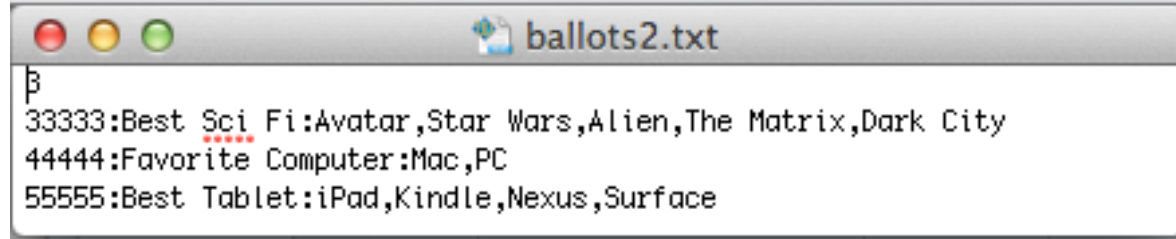
If the same voter tries to vote again, a message will be shown and the login will not be accepted. Clicking on the X in the window will not quit the program. To quit, the user (i.e. vote administrator / officer) must terminate the program from the command prompt.

The administrator could invoke the program with a different ballot file, as shown below:

```
java Assig4 ballots2.txt
```



where the `ballots2.txt` file is



The voters.txt file does not distinguish between ballots, so with this new ballot Herb Weaselman (ID: 1234) could not vote unless his status was set back to "false" in the voters.txt file. The idea is that in any given election there would be only one ballot file, containing all of the ballots for that election. Prior to the next election, the voters.txt file would be reset by the administrator so that all voters could vote in that election.