COMP1010 FUNDAMENTALS OF COMPUTER SCIENCE

ASSIGNMENT 3 - E-VOTING SYSTEM WITH RECURSIVE LINKED LISTS

Last updated: October 12, 2021

Assignment Details

- The assignment will be marked out of 100 and it is worth 15% of your total marks in COMP1010.
- Due date: Sunday, 7^{th} November 2021 at 21:00:00 AEST.
- Late penalty: 20% per day or part of. Eg., late by 1 second to 24 hours $\rightarrow$ 20% (3 mark) penalty, late by 24 hours and 1 second to 48 hours $\rightarrow$ 40% (6 mark) penalty, ...
- Topics assessed: Recursive Linked List, Classes and Objects.

Background

In this assignment, you are going to code a simulation of an electronic voting (e-voting) system using recursive linked lists. The objective of the e-voting system is to:

1. deliver unfilled voting cards,
2. collect voting cards,
3. check cards for validity (i.e., remove unfilled, blank and incomplete votes), and,
4. deliver election results (e.g., candidate with majority votes or votes for each candidate).

The e-voting package consists of six main components, including **three** that you need to update to complete the assignment:

1. **`Ballot.java`**,
2. **`EmptyBallots.java`**,
3. **`FilledBallots.java`**,
4. `Graded.java`,
5. `Timestamp.java`, and,
6. `UnitTest.java`

The process of the vote is as follows:

- Before the vote can take place: the system generates enough official empty voting cards (`Ballot.java` and `EmptyBallots.java`).
- During the voting period (`FilledBallots.java`): each voter extract a valid voting card, and after filling it with their favorite candidate, submit it for the ballot. The e-voting system adds a time stamp to record the vote, and all votes are stored in a different data structure.

- When the voting period concludes (`FilledBallots.java`): all votes are present in the filled ballots, the e-voting system must first check all votes for validity (e.g., valid voting card, filled section, non-blank and valid candidate names) and discard all invalid, incomplete or blank votes.
- Results announcements: the last step is to declare the candidate with the majority (more than 50% of the valid votes), if any, and count the results for each candidate.

Your Task

Your task is to implement the e-voting system by completing the methods in the three classes that correspond to the e-voting components described earlier. Namely:

- The class `Ballot` in file `Ballot.java` represents voting cards (i.e., ballot) with their methods to help the voting process (e.g., construction, filling, validity).
- The class `EmptyBallots` in file `EmptyBallots.java` controls the creation of enough empty voting cards with an official identity to be stored in a recursive linked list of unfilled voting cards, ranked **in increasing identity order**.
- The class `FilledBallots` in file `FilledBallots.java` controls the voting and reception of filled voting cards to be stored in a (different) recursive linked list of filled voting cards. Each voting card receives a timestamp to record the time of vote. The linked list keeps all filled votes ranked **in increasing order of received timestamps**. The class controls the validity of Ballot draw by checking the validity of the voting cards on the FilledBallots linked list, discarding the incomplete (incorrect name), blank (no candidate filled) or invalid cards (invalid card identity). Formally, a ballot is valid if it is: filled, time-stamped, have an official id, and a vote for one of the candidates listed. Finally, the class declares the candidate with a majority (i.e., **more than 50% of the valid** cards in the linked lists), if any, and provides results of the other candidates.

  For the advanced Pass part, your code is also tested for efficiency (using time limits to run certain methods).

  For the `High Distinction part` of the assignment, there could be more than one voting place (i.e., more than one `FilledBallots` list), and you need to merge the votes from different voting places (i.e., by merging linked lists in correct order),

Finally the classes `Graded.java` and `Timestamp.java` are already written for you, and are not part of your submission (**do not modify them**). `Timestamp.java` is the class that provides a unique time stamp (a random number that represents the time of the vote). The Class guarantees a larger value at each call (i.e., timestamps are produced in increasing order). The class `UnitTest.java` is also already written for you to test your code out of 100. (It is using `Graded.java` for automarking.) You can change it if you wish test particular parts of your code (it is not for submission), but it provides hints on what should be tested in your code. We will change values and sizes of each test when we mark your code.

In summary, there are two kinds of linked lists (EmptyBallots and FilledBallots) of voting cards (Ballots) to implement and manipulate. There is a unique EmptyBallots list and one FilledBallots list. For the High Distinction attempt, there will be more than one voting place, hence several FilledBallots list (one for each voting place) that will need to be merged before the checking and counting occurs.

To implement the linked lists, you are free to add more attributes (class variables) and methods to the three files that you have to submit, except that you are not allowed to use arrays unless the method requires it (e.g., candidates array). You are also **not allowed to import any packages**. See the section Limitations further below for more details.

You can find the exact marks allocated for each method that you have to implement in Marks and Grade Distribution section further down in this document. The details of what each of these methods do can be found in the code template given to you. It is recommended to code each missing parts and methods in the order provided there. Although you may not be able to follow this exact order because there are inter-dependencies between the classes. E.g., once you complete these pass-level methods, you can start concentrating on the rest of the Credit and Distinction level methods.

Limitations and Constraints

The assignment assesses your understanding of linked lists, therefore you are NOT allowed to use an array in any of your methods unless it is given as an input, or if you are required to return an array.

You are not allowed to use any function from outside the provided template. Creating helper functions of your own is absolutely fine, but those in turn cannot call outside functions either.

JUnit Tests

As in previous assignments, you are given a JUnit test (`UnitTest.java`) which you should use to ensure that your code meets all the requirements. The assignment will be auto-marked using a similar test suite. The values used in the tests will be modified when your assignment is being marked, but the tests themselves should be very similar.

Marks and Grade Distribution

Your assignment is going to be marked out of 100 coming from the automarker from 19 assessed parts.

For the automarking, here is the list of methods and tasks that you need to complete and their marks:

- Pass Level (64 marks):
  - in `Ballot.java`:
    - (5 marks) — `fill`
    - (5 marks) — `isValid`
  - in `EmptyBallots.java`:
    - (5 marks) — `EmptyBallots` (class constructor)
    - (5 marks) — `remove`
    - (5 marks) — `isValid`
  - in `FilledBallots.java`:
    - (5 marks) — `addVote`
    - (5 marks) — `countVoteFor`
    - (5 marks) — `size`
    - (5 marks) — `isValid`
    - (5 marks) — `insertBallot`
    - (5 marks) — `removeLateBallots`
    - (5 marks) — `addVote (advanced)`
    - (4 marks) — `size (advanced)`
- Credit Level (10 marks):
  - in `FilledBallots.java`:
    - (5 marks) — `removeBlankBallots`
    - (5 marks) — `removeInvalidBallots`
- Distinction Level (10 marks):
  - in `FilledBallots.java`:
    - (5 marks) — `findPercentages`
    - (5 marks) — `findMajority`
- High Distinction Level (16 marks):
  - in `FilledBallots.java`:
    - (8 marks) — `Constructor, from 2 FilledBallots`
    - (8 marks) — `Constructor, from n FilledBallots`

Submission Format

You need to submit three files (**no zip files**):

- `Ballot.java`
- `EmptyBallots.java`
- `FilledBallots.java`

Upload these files to the submission box.

Penalties

Similar to the other assignments, there are some other strict conditions that you must follow:

- Standard rules applicable to all submissions apply to assignment 3.
- Enter personal details (Student ID, Name) in each of the submitted file. You will receive a 20% penalty if you fail to do so. You also need to put an 'x' inside the box the state that the assignment is your own work (if this doesn't show, Eclipse may be 'folding' (hiding) the lines, so please double-check. Following is the relevant part at the top of the concerned files:

```
//ID, NAME (For example, "40404040, Janet Kim")
//Put x inside [] below:
//[]    This assignment is entirely my own work and
//      I have not seen any other persons code or design
```

- Any compilation errors and/or infinite loops in any of the submitted files will result in an automatic zero.
- Submissions must be self-contained and does not require any other file besides the one provided (and the JUnit test file). Any dependency on an external file (e.g. using functions from outside the submitted files) will result in an automatic zero. You must also not modify `Graded.java` and `Timestamp.java` since there are not part of your submission.
- Changing any package declaration in any submitted file will result in an automatic zero.
- Using any functions from outside the project will result in an automatic zero. You may add helper functions.
- Modifying the visibility of any methods or attributes in any of the submitted file will result in an automatic zero (please leave everything as public so that the JUnit tests can access them directly).

Change Log

If there are any changes to this documentation after its release, it will be listed here. Please check to ensure you have read the latest documentation. All changes will be announced on iLearn.

Frequently Asked Questions and Hints

Various Question&Asnswers of interest, Hints and other comments will be posted on Assignment 3 forum on iLearn when needed (i.e., please read these first before asking).