COM212 Spring 2020 - Final Project

Due Monday 05/11 at NOON for seniors and NOON on Wednesday 05/13 for non-seniors.

MySocialProfile

Your Project Teams: Pick your own teams of 3 by Sunday 04/26 and indicate so in this sheet.

In this project, you will create a console-based social networking profile that focuses on a single user's account details, events, friends, and a timeline of posts. It will run in the terminal or command prompt and will store the user's profile data in a single text file. Think of this project as a simplified component of a larger social networking platform that manages many user profiles using a mobile app or a web-based user interface and a database that stores the data on a web server. In this project, your platform will focus on a single user instance (profile), with a defined set of interactions.

An instance of a **MySocialProfile** will hold the following pieces of data for the user:

1. **Full Name**
2. **Email Address (Used as a unique user ID)**
3. **Password**
4. **Class Year**
5. **List of upcoming events that the user plans to attend**
6. **List of the user's "timeline" posts**
7. **List of the user's friends (using their user IDs - email addresses)**

When saved in the text storage file, let's name it **mysocialprofile.txt**, this profile data **might** look like:

Katherine Bergeron
kbergeron@conncoll.edu
Camels2016
2020
"12 03 2019 16 15 Faculty Meeting in Cro",...
"Keeping calm...", "Go Camels!", "Out raising money for the college", ...
"gparker@conncoll.edu","aross@conncoll.edu","mzim@conncoll.edu",...

In the text file, each **line** is a different field of data for the user profile.

Every time the program is run, the text file data should be read in, and every time it is exited/shut down, the updated data should be written back out to the file. When the data is read from the file, it will be stored into an instance of the MySocialProfile class, where each field will be created from the file data. This instance of the profile will be used during program interactions.

The data structures you choose to use for these fields are up to you, except that the list of events should be a priority queue. For this, you can modify the priority queue that we implemented in class using an array-based heap. Such that while the program is running, all events that the user plans to attend will be kept on a heap-based priority queue, where the highest priority event is the next one to take place (based on its date and time). Once the event time passes, the event is removed from the heap. (You can use built-in Java tools for getting the current local date/time as well as for comparing two dates/times to see which comes first.) You should update the heap to reflect the events for the current date/time every time the program is run; that is, delete any passed events that are read in from the file.

The user-interactions are described below, but make sure your interface is very readable - i.e. the user prompts and program outputs that get printed to the console. So, format your print-outs appropriately for easy readability.

From the main menu prompt, the user can either:

1. Create a new account/profile,
2. Load an existing profile (if one exists in the file), or
3. Exit the entire program.

A. If the user chooses to create a new account, they will be prompted for their info and a new MySocialProfile will be created for them.
B. If they are an existing user, then their profile details should be loaded from the mysocialprofile.txt file.
C. Exiting quits the entire program.

After a new or existing user is loaded, their "home screen" is displayed. Which shows the following information:

- their next event that is scheduled to take place, if available,
- their 3 most recent timeline entries
- a listing of all the events they plan to attend.

The user is then prompted with the following secondary list of options:

1. posting to their timeline,

2. adding an event to their list that they plan to attend,
3. viewing a listing of their friends,
4. adding/removing a friend, or
5. logging out.

I. If they choose to post to their timeline, they are then prompted to input text that will become the latest item in their timeline.
II. If they choose to enter a new event, they will be asked for the event info, including date and time, and it should get written to their record. (When you eventually write this info back to the text file you will want to follow some standard format that will allow you to easily extract the time and date of the event, so consider this when you are prompting the user and storing the data.)
III. If they choose to list their friends, their friends are displayed using their IDs (email addresses)
IV. If they choose to add a new friend or remove an existing friend, they are then prompted for the email address of the friend. If the email address matches an existing friend, that friend gets removed from their list of friends. If not, then the email address is added to the list of friends.

The user is returned to their own "home screen" after completing any of these options. At this point, the home screen is re-displayed with the latest info, followed by the secondary list of prompts.

V. If they choose to log out, the profile data is saved into the mysocialprofile.txt file, essentially persisting the current profile data for next time. The main menu prompts are then displayed, giving the user options to start another session or exit the program.

The object-oriented design of the project is up to you and your development team. As a suggestion, you will need to design at least 3 classes: MySocialProfile, Event, and the main class that drives your program.

You will need to do a lot of planning and design work as a group initially. This will allow you to break the project up into pieces and assign tasks to each member of your team. You should do this within the next few days so everyone can get started on implementing their part ASAP. Coming up with a very clear specification for each method of each class during the design phase is important. This can change as you go (as long as all changes are communicated to the relevant team members), but you must have a clear place to start from.
It will be a good idea for everyone to work and collaborate together as much as possible so you can easily discuss/approve small tweaks in the design/specs with each other on the fly as you are implementing.

You will want to have very regular group meetings so that you can regroup, give each other updates, get input/help from one another, and modify the overall design as necessary. I strongly encourage you to come up with a team name, a schedule with planned goals/milestones, and a division of certain smaller tasks once a full design picture is agreed upon.

Appendix

Sample programs. (We have explained what's going on in these programs in the comments. Let me know if you have any questions!)

- See the "BST Code Files" on Moodle under Class #21 for a code demo on how to create an interactive console session.
- Here is a sample java program that reads some data from this sample text file.
- Here is a sample java program that demonstrates basic file reading and writing.
- Here is another sample java program that creates and manipulates java calendar objects the way you will need to for this assignment.

If you want to read/learn more about Java i/o, here are some links...

- Here is a quick and easy-to-read page on taking input from and writing output to a file in Java: http://en.wikiversity.org/wiki/Java_File_IO
- Some others if you want to read/learn more:
    - http://www.toves.org/books/java/ch22-file/index.html
    - http://www.seas.upenn.edu/~cis1xx/resources/java/fileIO/introToFileIO.html

Some date/time stuff:
https://www.mkyong.com/java/java-convert-date-to-calendar-example/

If you find helpful online references for java file i/o, java date/time stuff, or other java specific tutorials/tools, please post them to Moodle and share with your classmates!

Submission Requirements and Grading

A final project submission link will be added to Moodle, and your final project is to be submitted the same way all other projects have been: with all files (and subdirectories, if any) collected into one folder, which is then compressed into one .zip file. Name your submission folder using the usual convention (username_finalproj).

Only one group member should submit each project, but make sure all group members' names are displayed prominently in the comments at the top of each .java file.

In your submission folder, you will also need to include a **readme.txt** file, which will indicate how to compile and run/use your program. Think of it as an abbreviated user's manual that will

tell me which .java files to compile, and inform me of anything that is not *obvious* about the user interface.

Your project will be evaluated on **style and functionality**, as usual. In terms of functionality, all of the features described on this assignment page should be functional and working according to specifications. NB: Anywhere that a particular implementation instruction is given above (e.g., "use a priority queue for the user events"), you will be held accountable for implementing it according to how it was discussed in our class during the semester.

All classes and their methods should be supplied with **Javadoc** comments. In-line or block comments, where appropriate, are also expected. You should also mention in the comments above each class what role that class plays, i.e., how that class fits-in or interacts with the rest of the classes.

Tentative Grading Rubric:

Overall design and documentation (including Javadoc, inline and block comments) of the entire project	10%
Implementation/coding style	10%
MySocialProfile class The list of timeline posts The list of friends	20% 5% 5%
Priority Queue (for the user's events) expired events get removed upon each profile load (or login) **The Event class**	20% 10%
User Interface (combining all the pieces together) meets specs, satisfies the description of the functionality output: friendly, readable user input: friendly, easy-to-use reasonably robust to bad user-input	20%