CSC 130
Programming Project #4

Due Date: by *11:55pm* Wednesday, December 11, 2013

Objective:

- To practice developing program logic
- To practice creating and using a linked data structure
- To design and implement multiple classes

The Problem:

In this assignment, you will use arrays and linked lists to implement a simple version of Facebook. Your program will read a sequence of commands to create people, record that two people are friends, etc. These commands will be read from an input file and processed one at a time. All output will be placed in the console.

Specifications:

Your project (*project4*) should contain all classes, including exception classes (i.e. you will not need to link to other projects and include import statements.) The application class will be responsible for reading and processing the commands stored in the data file (commands.txt). Each is explained below.

- P <name> <security level> – Create a Person object; you may assume that no two people have the same name.
- F <name> <name> – Record that the two specified people are friends.
- U <name> <name> – Record that the two specified people are no longer friends.
- L <name> – List the friends of the specified person.
- Q <name> <name> – Check whether the two people are friends and display an appropriate message.
- V <name> - List the friends of the specified person and perhaps their friends depending on the security level.
- X – Terminate the program.

For example:

Input	**Output**
P Sue 1 P John 0 P Ellen 0 P Mike 0	

<call the toString method on facebook object>	`Sue is friends with:`
	`John is friends with:`
	`Ellen is friends with:`
	`Mike is friends with:`

F John Sue F Ellen Sue F Mike Sue F Mike John	
<call the toString method on facebook object>	`Sue is friends with: Mike Ellen John` `John is friends with: Mike Sue`

	Ellen is friends with: Sue Mike is friends with: John
U Sue John	
<call the toString method on facebook object>	Sue is friends with: Mike Ellen John is friends with: Mike Ellen is friends with: Sue Mike is friends with: John Sue
Q Ellen Mike	Ellen and Mike are not friends
Q Ellen Sue	Ellen and Sue are friends
V Sue	Sue is friends with: Mike Ellen and Mike is friends with: John Sue and Ellen is friends with: Sue
V Mike	Mike is friends with: John Sue
X	

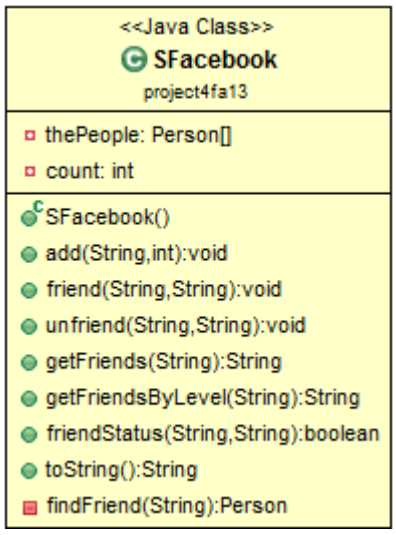
Input File:

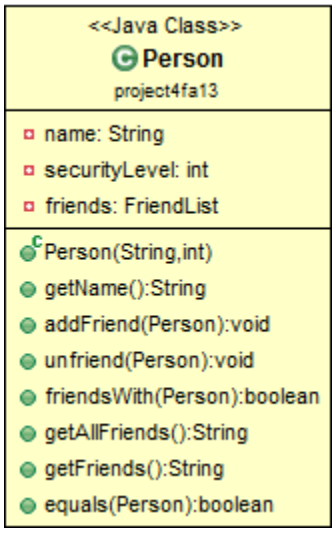
An input file will be provided at a later date which will allow you to further test your program. For now, you should create a test file and include some simple commands. You can add commands to this file as you create/write additional methods to process these new commands. The following guidelines will apply.

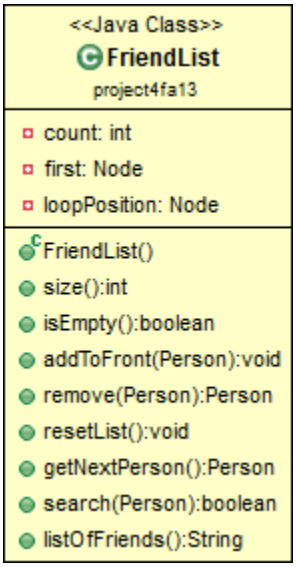
- You can assume that the data file is correctly formatted and will adhere to the formatting guidelines outlined previously.
- A FriendNotFoundException will be thrown if a name mentioned in an F, U, L, Q or V command is not found.
- The sequence of commands ends with X.
- If a person friends or unfriends himself, the program should do nothing; it should not add the person to his own list of friends.
- If a person unfriends someone who is not a friend, the program should do nothing.

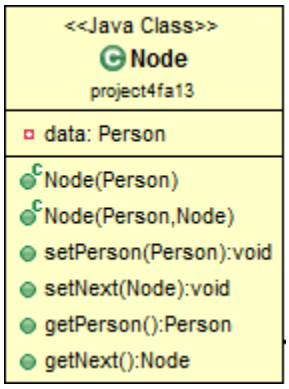
UML Class Diagrams:

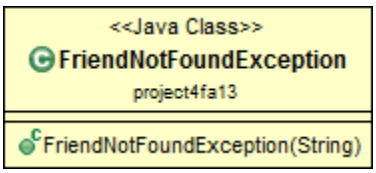
The classes that need to be developed are outlined below.











Notes:

- The FriendList class contains a linked list. The design of the linked list is different than the approach that we have taken in class/lab. You should design this class with header and trailer sentinels (see page 127 of our textbook on *Header and Trailer Sentinels*). The design simplifies coding by eliminating the special case of having to modify the reference to the first node when inserting/removing the first node.
- The Node class and the FriendList class should not be developed using generics.

The Application Class:

The application class is responsible for reading and processing the commands.

Output:

For each command processed you should display information about the command and provide output that clearly shows the effect of the command. All output should be displayed in the console and should be properly labeled.

Plagiarism Policy:

This assignment is to be completed individually (not with your pair programming partner). If you receive substantial help from another person you must give them credit for their contribution in a comment. It is not considered acceptable for you to share your code with another student in any way. You may discuss an approach from a conceptual perspective only.

General Requirements:

- You should spend some time making your output as neat as possible. **The overall appearance of the output will be graded.**
- Neatness Counts!!! Proper indentation is required.
- Be sure to select meaningful variable names.
- Include plenty of comments. The comments must be formatted according to **JavaDoc** standards. **Documentation will be considered when grading this assignment!**

Submit:

- **Create a .zip file to upload to Blackboard:**
  - **Right-click the project folder, choose Send To | Compressed (zipped) Folder**
- **Upload the .zip file to the Project #4 assignment in Blackboard by 11:55pm on Wednesday, December 11th.**

Late Assignments:
Assignments received by 11:55pm on Thursday, December 12th will receive a maximum of 90 points.
Assignments received by 11:55pm on Friday, December 13th will receive a maximum of 80 points.
Assignments will not be accepted after this date/time for any reason.