CS 171: Introduction to Computer Science II
Project (Option 1): Emory FaceSpace
Due: Friday, May 4, 11:59pm
(Optional) demo due: Sunday, April 30, 11:59pm

1 Introduction

For this project, you will create a social network application called FaceSpace. A social network, in the simplest sense, is a means of keeping track of a set of people (each of whom have a "profile" in the social network) and the relationships (usually involving friendship) between them. A common way to represent this network is using a graph. Each node in the graph represents a user profile, and an edge connects two users if they are friends.

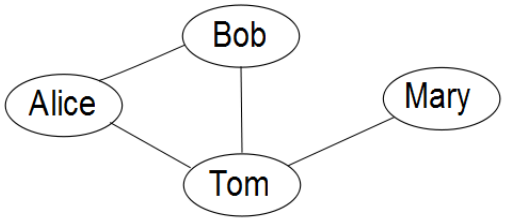

Figure 1: Example of a social network. Alice, Bob and Tom are friends with each other. Mary and Tom are also friends with each other.

Other than the basic requirements outlined below, the project is very open ended. A sample GUI and a suggested design are provided. You can work as a team of up to 2 students.

2 Requirements

The basic requirement of the project is a graphical application that supports the following operations:

Add a User: Adds a new user profile with a given name to the network. All user profiles should be uniquely identified by their names. Your program should not allow a user profile to be added if the given name already exists in the network.

Select a User: Retrieves the user profile with a given name and displays all its information including the name, a list of its friends, and other optional information (such as current status and profile picture).

Add a Friend for the Selected User: Adds a profile with a given name to the the list of friends of the selected user. We assume all the friendships are reciprocal. In other words, if Bob adds Alice into his list of friends, Bob will also be added into Alice's list of friends. Moreover, only users present in the network can be added as friends.

Find Shortest Path to a User from the Selected User: Computes the shortest path (the minimum number of hops) between the selected user and a user with a given name. Note that the graph is unweighted so the shortest path is simply the path with minimum number of edges between the two nodes and can be computed using a breadth-first search algorithm. Note that there may be cases where two users are not connected at all.

A README file is also required with the instructions for starting and using the application as well as a discussion of your design choices.

2.1 Sample Graphic Interface Design

A sample user interface is shown in Figure 2. This is only intended to be an example, please feel free to modify it or design your own. In this interface, the screen is divided into three areas. The top part contains the buttons and text fields (implemented using `GButtons` and `JTextFields` objects) for operations that edit the network (i.e. add, delete and select/look-up a profile). The left part contains the buttons and text fields for operations that edits the selected profile (i.e. editing the current status, finds shortest path to a given user). The middle part (implemented using a `GCanvas` object) displays the information about the selected profile (e.g. for a selected profile Alice, it shows she has Bob as a friend, and her status is "'Coding..."').

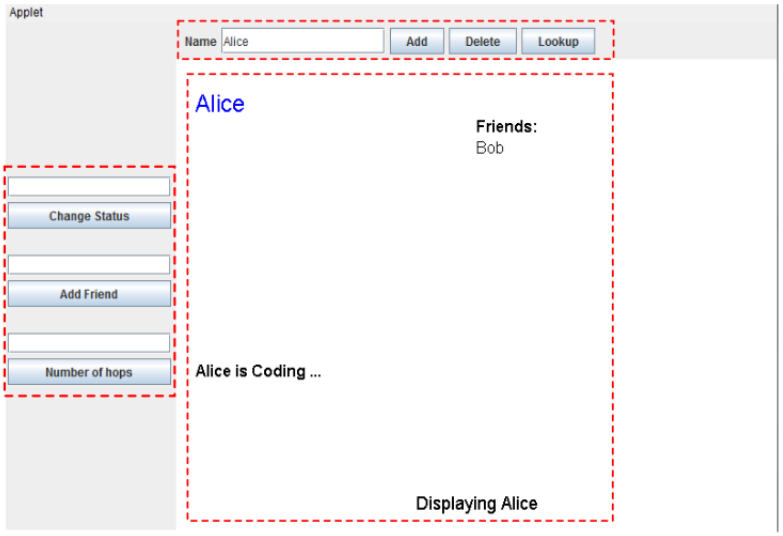

Figure 2: A sample interface partitioned in three areas (dashed lines)

2.2 Suggested Class Design and Data Structures

You may consider implementing your program using several classes with the suggested data structures. Please feel free to modify it or design your own (in which case please justify it in your README file).

1. A class that represents a user profile. It stores all the information about a profile including name (the key of the profile), a list of friends (you can use an `ArrayList` of `String` to store the list of the names of the friends), and other optional information.

2. A class that represents the network, i.e. all the profiles and their relationships in the social network. You can use an `ArrayList` to store the profiles (the nodes in the graph). For better efficiency, you can implement a Binary Search Tree (BST) for storing the user profiles. Note that the list of friends in each profile stores the edge information in the network. You can implement the methods needed to find, insert, and delete (delete is not required) a node (i.e. a profile), and method to find the shortest path in this class.

3. Classes for the GUI interface. You may have a main GUI class for building and managing the mouse click events (you can extend `Program`). You can also have a class for the GUI component that displays a user profile (extend `GCanvas`).

3 Grading

Base credits:

- Functionally correct program. (70 credits)
- Well organized code (good design of classes/methods). (10 credits)
- Application quality (Easy to use, clean interface, good use of the data structures and algorithms). (15 credits)
- Error condition handling. (5 credits)

Extra credits:

- Additional functionality and exceptional efforts and creativity. (up to 10 credits)

4 Submission

Please make sure that you have added the following statement at the top of each source code file you wrote and include the name of each team member.

```
/*
  THIS CODE IS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING
  A TUTOR OR CODE WRITTEN BY OTHER STUDENTS. Team Member Names.
*/
```

Please submit your project to Amy as a tar or zip file containing: your source files, the README file and any other files you are using in your project.

The project is due on May 4. However, you are encouraged to submit a demo version of your program by April 29. Based on the submitted demo versions, selected teams will be invited to present/demonstrate their project in class on May 1 (for 5 extra points). This demo version will not be used for grading.