Saturday, 29 August 2015

OOP Final Exam Section C

Social Network!

# Background

# You and your friend have decided to start a company that hosts a gaming

# social network site. Your friend will handle the website creation (they know

# what they are doing, having taken our web development class). However, it is

# up to you to create the classes that manages the game-network information

# and to define several methods that operate on the network.

# In a website, the data is stored in a database. In our case, however, all the

# information comes in a big string stored in a text file. Each pair of sentences in the text

# is formatted as follows:

# <username> is connected to <name1>, <name2>,...,<nameN>.

# <username> likes to play <game1>,...,<gameN>.

# Your friend records the information in that string based on user activity on

# the website and gives it to you to manage. You can think of every pair of

# sentences as defining a gamer profile. For example:

# John is connected to Bryant, Debra, Walter.

# John likes to play The Movie: The Game, The Legend of Corgi, Dinosaur Diner.

# Consider the data structures that we have used in the course - Array/ArrayLists, Hashtable, and combinations of the two. Pick one which will allow you to manage the data above and implement the methods below.

# You can assume that <username> is a unique identifier for a user. In other

# words, there is only one John in the network. Furthermore, connections are not

# symmetric - if John is connected with Alice, it does not mean that Alice is

# connected with John.

1. Create data structure by reading the text file
2. Get Connections given a user as argument
3. Add connection give user A and user B
4. Add new user
5. Connections in common

Skeleton Code:

Create a Class SocialNetwork with the following methods:

1. createDataStructure(); takes a path to the file and return a reference to the network data structure
2. getConnections(); takes a user as argument and returns a list of users
3. addConnection(); takes two users as arguments and connects them on the network
4. addUser(); takes user and their game preference and adds them to the network
5. getCommonConnections(); takes two users and returns the list of users

Test Cases:

1. getConnections("Mercedes"); returns ["Walter", "Robin", "Bryant"]
2. addConnection("Mercedes", "John");
3. getConnections("Mercedes"); returns ["Walter", "Robin", "Bryant", "John"]
4. getCommonConnections("John", "Walter"); returns ["Bryant"]