Programming Assignment 1

Assigned: Sept. 7
Due: Sept. 28

In this assignment, you will implement a micro-version of Facebook.

Specifically, your program will accept from input a sequence of commands of the following forms, one command to a line:

- P ⟨name⟩ – Create a person record of the specified name. You may assume that no two people have the same name.
- F ⟨name1⟩ ⟨name2⟩ — Record that the two specified people are friends.
- L ⟨name⟩ — Print out the friends of the specified person.
- Q ⟨name1⟩ ⟨name2⟩ — Check whether the two people are friends. If so, print "Yes"; if not, print "No"
- X – terminate the program.

For instance, this is one possible input and output:

```
Input                  Output
P Sam
P Liza
P Mark
P Amy
F Liza Amy
F Liza Mark
F Amy Sam
L Amy                   Liza Sam
L Sam                   Amy
Q Liza Mark             Yes
X
```

Data Structures

You will have two classes: a `Person` class, which is similar to but not the same as the one in the course sample code; and a `MicroFB` class, which is the driver class, with the `main` method.

In the `Person` class:

A. Define the static constant `maxFriends=10`, the maximum number of friends any one person is allowed.

B. Define three data fields: `name`, a `String`; `numFriends`, an `int`; and `friends`, an array of `Person` of size `MaxPeople`. Note that the field `p.friends` field must be an array of the `Person` objects corresponding to the friends of `p`, and *not* an array of their names (which are `Strings`). The data fields should be `private` or `protected`.

C. Define getters for all three fields and a setter for `name`.

D. Define a constructor `Person(String n)`.

E. Define a method `void addFriend(q)` that does the following when `p.addFriend(q)` is called:

   i. Checks that `q` is not null; that they are not equal; and that they are not already friends.

   ii. Checks that neither `p` nor `q` has maxed out the number of friends.

   iii. Adds `q` at the end of the array `p.friends` and vice versa.

   iv. Increments `p.numFriends` and `q.numFriends`.

In the `MicroFB` class:

F. Define the static constant `maxPeople=100`, the maximum number of people.

G. Define a global array `Person allPeople[maxPeople]` holding all the people that have been created, and a global `int peopleCount` holding the number of people that have been created.

H. Create a method `Person findPerson(String name)` which searches through `allPeople` to find the person of the given name. Return `null` if no such person has been created.

I. Create a `main` method that loops through the input one line at a time, and does the following:

   To execute a "P" command,

   - Use `findPerson` to check that no `Person` with that name already exists.
   - Check that `peopleCount` is less than `maxPeople`.
   - Create a `Person` object for the name, save it in `allPeople`, and increment `peopleCount`.

   To execute an "F" command:

   - Use `findPerson` to find the two `Person` objects in `AllPeople`
   - Call `addFriend`.

   To execute an "L" command", use `findPerson` to find the person and loop through the list of friends.

   To execute a "Q" command, use `findPerson` to find the two people, and check whether the first is on the list of friends of the second.

   It is good programming style to write a little submethod for each of these functionalities, rather than one huge main program, but it is not required.

Input/Output

You may assume that the input is correctly formatted. That is:

- Each line consists of a command character "P", "F", "L", "Q", or "X" followed by a blank followed by one or two names separated by a blank. A name is a sequence of alphabetic characters. Do not worry about normalizing case.
- The sequence of commands ends with "X".

What, if anything, you want to do about invalid inputs is up to you. It is OK for the program to crash.

However, the program *should* do the right thing under the following circumstances:

- The command "P" is called with a name already given to a person.
- The command "P" is called when `peopleCount` is equal to `maxPeople`.
- Any of the commands "F", "L", or "Q" gives a name that has no associated person.
- The command "F" is given where one of the people already has `numFriends` equal to `maxFriends`

In any of these cases, the program should print an appropriate error message, do nothing more for this line of input, and continue on the next line of input.

The program may take its input either from standard input or from a text file in the same directory as the program named "input.txt" – your choice.

The Java library class `Scanner` is a handy one for reading either from the terminal or from a text file. You can find examples of its use in the files `wc.java` and `wcFile.java` on the course web site.

If this form of input seems to you too dreary for words, you can feel free to design a snazzier one, as long as it supports the above functionalities and it is immediately obvious to the grader how to work it. It is entirely up to the grader to decide what is obvious to him; I am not going to overrule that.

Submission

Since the class `MicroFB` has the `main` method, it should be in a file called MicroFB.java. The class `Person` may be placed, either in that file or in the file Person.java. Upload the source code files to the NYU Classes site. If there is anything about the code that requires explanation, upload a README.txt file in plain text format. *Nothing else should be submitted.*