CSCI 1101
Computer Science II

Assignment No. 4
Date Given: Monday, March 20, 2017
Due: Monday, April 3, 2017, 11.55 p.m.

This assignment will test your skills in creating and using linked lists and file processing. You should not use the Java built in linked list class. Instead, you must use the Node class and LinkedList class discussed in the lectures, and make necessary modifications to these classes. First download Node.java and LinkedList.java. Also download LinkedListDemo1.java and LinkedListDemo2.java from the assignment webpage. Study the programs before you begin the assignment.

The specification for this assignment is long, but the assignment itself is not difficult once you start understanding the basic structure and the methods, and implementing the classes.

FriendList is a new social media tool. It keeps track of all of its users and their friends. It can calculate the total users of the social medial tool, the user with the most friends, the user with the oldest friend, and can also find the common friends between two users. Users can also be added and removed from Friendlist. In order to join FriendList a user must be at least 13 years old and when a user is removed, not only is the user removed from FriendList but also from any user's friend lists.

Each user has a name, their current location (name of the city or town), and birth year. Users also have a list of friends (a linked list of a User's friends). If a user adds another user as a friend, then the other user will also add this user as a friend (that is, if A is a friend of B, then automatically B is a friend of A - similar to Facebook).

You will need to implement two classes for this social media tool: FriendList, and User. The attributes and methods for each class are listed below (as well as a short explanation of some of the methods).   You will implement all the lists in this program as linked lists. In addition, you will need to modify the Node and LinkedList classes so that data stored/retrieved is of User type.

Make sure you do proper error testing as well, for example you should not add a user to FriendList if they already exist. And new members need to be at least 13 years old. You can add extra methods or attributes if you find it necessary.

FriendList

This class has one attribute – allUsers (a Linkedlist of Nodes that stores Users).
It also can implement the following methods:

- FriendList( ): //no args constructor that creates a new empty list
- addUser (User u) : void //adds a user after checking that they are at least 13 years old and do not already exist
- removeUser (User u) : void //remove user from FriendList and remove that user as a friend of any other users
- totalUsers ( ) : int //returns the total number of users on FriendList
- getUsers () : LinkedList //returns the list of all users
- mostFriends ( ) : User //returns the user who has the most friends. If two or more users have the 'most friends' (i.e., a tie for most friends, return the first User with that number (similar to how ArrayLists handle this case)
- oldestFriend() : User //returns the user(s) with the oldest friend. If two or more users have 'the oldest' friend (i.e., same birthday – return the first User (similar to how ArrayLists handle this case)
- commonFriends (User, User): LinkedList //find common friends between users and returns new list

User

Attributes:
String name
String location //this is where they currently live (city or town)
int birthYear
LinkedList friends //this is a LinkedList that holds Nodes of Users who are their friends

It implements the following methods:

- User (String, String, int): //set name, location and birthYear, initialize the LinkedList
- getName ( ) : String
- getLocation ( ) : String
- getBirthYear ( ) : int
- isEqual (User) : boolean
- getFriends ( ) : LinkedList

- getNumFriends ( ) : int //returns the number of friends the user has
- toString ( ) : String //returns the name and location of the user
- addFriend (User u) : void //adds user u as a friend to their list and user u adds this user to their friend list
- removeFriend (User u) : void //removes user u as a friend from their list and user us removes this user from their friend list
- oldestFriend ( ) : User //returns the friend who is the oldest (e.g., 30 years old). If there is tie (i.e., same birthday) return the first friend with that birthdate.

In order to use Node class you will need to change it to hold User objects instead of just Strings. You will also need to make the necessary changes to set and get to accommodate the data stored as type User.

For example, the new Node class could look like the UML diagram below:

Node
- user : User - next : Node
+ Node (User ud, Node n) + setUserData (User) : void + setNext (Node) : void + getUser ( ) : User + getNext ( ) : Node + toString ( ) : String

You will also need to adapt the LinkedList class to accommodate the fact that Nodes store User data (see the UML diagram below for a list of suggested methods that you could include and you can add other methods)

LinkedList
- front : Node - count : int
+ LinkedList ( ) //set front to null and count to 0 + size ( ) : int + isEmpty ( ) : boolean + clear ( ) : void //make the list empty + addNodeToFront (User) : void + getFront ( ) : Node //get the first Node in the list + enumerate ( ) : void //scan list and print the content + removeFront ( ) : void //remove first Node in list + removeLast ( ) : void //remove last Node from list + addNodeToEnd (User) : void //add a Node with User data to end of list + contains (User) : int //check to see if the list has a User and return the index of the user or return -1 if not in list + removeNode (int) : void //remove the Node at a given index + getUserAt (int) : User //return the User of the Node at the given index from the list + toString ( ) : String

You should write a demo program that reads data from two text files, users.txt and friends.txt and creates the FriendList. You don't have to make the screen dialog exactly as shown in the demo sample output shown at the end, but your demo should contain a menu that the users can select from (see Sample Output at end). The file users.txt contains the name, location and year of birth of each user, one on each line. The file friends.txt contains a list of friends of each user. The first word in each line is the name of the user and the remaining words are the friends. Sample users.txt and friends.txt files are given below. You may assume that the names are not repeated, that is, each user is unique.

<u>Sample users.txt input file (name, location and year of birth)</u>

```
Bob    Halifax             1992
Fred   Toronto             1996
Chris  Truro               2000
Xiao   St.John's           1997
Amar   Halifax             1995
Ben    Montreal            1991
Sara   Toronto             1993
Karen  Vancouver           1990
Hamed  Toronto             1991
```

<u>Sample friends.txt input file (for example, Bob from Halifax has Xiao, Sara, Hamed, Fred, and Chris as friends)</u>.

```
Bob    Xiao   Sara   Hamed  Fred   Chris
Fred   Bob    Ben    Chris
Chris  Fred   Bob    Sara
Xiao   Bob    Sara   Amar
Amar   Xiao   Karen
Ben    Fred
```

```
Sara   Bob    Xiao   Chris
Karen  Amar
Hamed  Bob
```

After you run the program (see the below table with Sample output) and the user chooses to quit the program, you should rewrite the two files (users.txt and friends.txt) so that the changes are saved.

For example, the two above files after the performing the Sample output would look like:

Sample users.txt input file (after changes)
```
Bob    Halifax       1992
Fred   Toronto       1996
Chris  Truro         2000
Xiao   St.John's     1997
Amar   Halifax       1995
Ben    Montreal      1991
Sara   Toronto       1993
Karen  Vancouver     1990
Hamed  Toronto       1991
Kali   Dartmouth     1987
```

Sample friends.txt input file (after changes).
```
Bob    Xiao   Sara   Hamed  Fred   Chris  Kali
Fred   Bob    Ben    Chris
Chris  Fred   Bob    Sara
Xiao   Bob    Sara   Amar
Amar   Xiao   Karen
Ben    Fred
Sara   Bob    Xiao   Chris  Kali
Karen  Amar
Hamed  Bob
Kali   Bob    Sara
```

Sample output displayed on the console shown in a table to save space (updates to users and friends will be written into a file)

```
Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 1

 OUTPUT

Users of FriendList:
Bob from Halifax --> Fred from Toronto --> Chris from Truro --> Xiao from
St.John's --> Amar from Halifax --> Ben from Montreal --> Sara from Toronto --
> Karen from Vancouver --> Hamed from Toronto -->

Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 2

 OUTPUT

Total users on FriendList: 9
```

```
Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 4

 OUTPUT

Enter the user's name, their home town, and year of birth: May Bedford
2000

Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 4

 OUTPUT

Enter the user's name, their home town, and year of birth: Pete Bedford
2014
Pete from Bedford  you are only 3. You need to be at least 13 to join
```

```
Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 3

 OUTPUT

Enter a user name: Bob
Bob's friends:
Xiao from St.John's --> Sara from Toronto --> Hamed from Toronto --> Fred
from Toronto --> Chris from Truro -->

Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 4

 OUTPUT

Enter the user's name, their home town, and year of birth: Kali Dartmouth
1987

Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 6

 OUTPUT

To create a friend, you need to enter two user names
Enter the name of the the first user: Sara
Enter the name of the second user: Kali

Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
```

```
FriendList

Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 6

 OUTPUT

To create a friend, you need to enter two user names
Enter the name of the the first user: Bob
Enter the name of the second user: Kali

Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 9

 OUTPUT

Enter the name of the first user: Sara
Enter the name of the second user: Bob
Common Friends between Sara and Bob are Xiao from St.John's --> Chris
from Truro --> Kali from Dartmouth -->


Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 11

 OUTPUT

User(s) have the oldest friend: Sara from Toronto --> Bob from Halifax -->
Their oldest friend is Kali who was born in 1987


Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
```

```
Selection: 6

 OUTPUT

To create a friend, you need to enter two user names
Enter the name of the the first user: Chris
Enter the name of the second user: May

Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 8

 OUTPUT

User with the most friends is Bob from Halifax
Friends:  Xiao from St.John's --> Sara from Toronto --> Hamed from Toronto --
> Fred from Toronto --> Chris from Truro --> Kali from Dartmouth -->
```

```
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 5

 OUTPUT

Enter the name of the user to remove: May
Welcome to FriendList! What would you like to do (press a number)?
1. Print out all the users.
2. Print the total number of users.
3. Print out all the friends of a user.
4. Add a new user.
5. Remove a user.
6. Add a friend.
7. Remove a friend.
8. Print the user with most friends.
9. Find common friends between two friends.
10. Find the oldest friend for a user.
11. Find the user with the oldest friend on FriendList.
12. Quit.
Selection: 12

----jGRASP: operation complete.
```

You can make appropriate changes and additions to Node.java and LinkedList.java we created in class – name your new files Node2.java and LinkedList2.java. Do NOT use java's Node and LinkedList classes.

Submit a zip file containing Node2.java, LinkedList2.java, User.java, FriendList.java, Demo.java, users.txt and friends.txt (before and after the changes), and sample demo output that use all the menu items and show good error checking. One sample demo that covers all the menu items such as the one shown above is sufficient.