**Programming Assignment 4**

**A** **micro-version of Facebook using Hashing with Chaining**

CS 146 – Intro to Algorithm and Data Structure

Design by Linjun Cao

**Table of Contents**

1. Explain micro-version of Facebook design and implementation details --------------------------------- 3
2. Explanation of the Classes/Functions
3. Classes ---------------------------------------- 5
4. Functions --------------------------------------- 11
5. The procedure of how to unzip files, install application, and run/test codes --------------------------------------- 21
6. Problems encountered during the implementation ---- 22
7. Lessons Learned --------------------------------------- 23

**Part I: Explain micro-version of Facebook design and implementation details**

First, I must make out how many classes and functions do this program need. There must be a class Person since this program is based on many ‘Persons’. And every node of a person has his or her name and friends. In addition, class Person must allow us add Person A as Person B’s friend or remove Person A from Person B’s friend. Also, it must tell whether Person A and Person B are friends or not and list all the friends of each person.

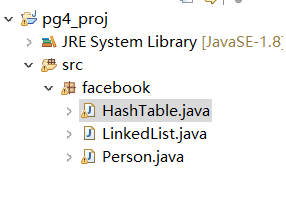
Next, to list all the friends of a person, there must be a data structure. I choose linked-list since it can link each person together. Inside the class linked-list, there is an inner class called Friend-Node. Friend-Node is a Node which can contain a Person inside. This inner class must be static so that we can use it directly in main method. Class linked-list also has following functions: insert, delete, exist and print-list.

Finally, we should make a global hash table. In this class we need to implement the idea from the textbook. Put the name of each Person to the hash table and manipulate them.

Part II: 2. Explanation of the Classes/Functions

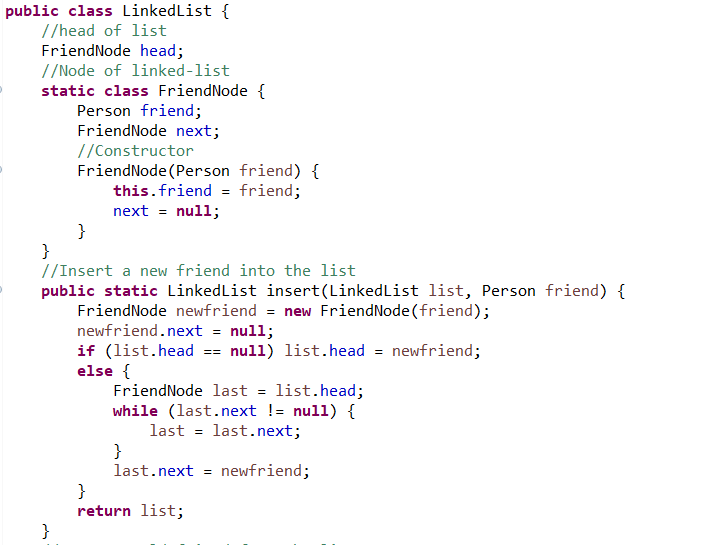
Classes:

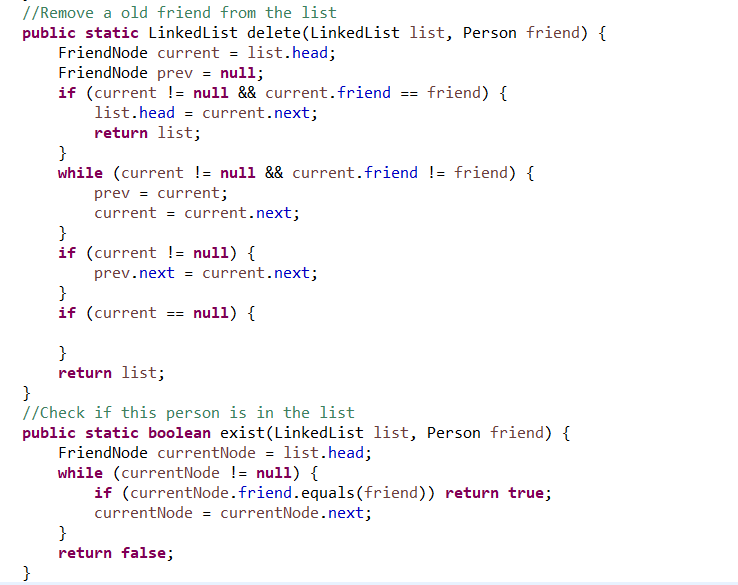
We have the following classes.

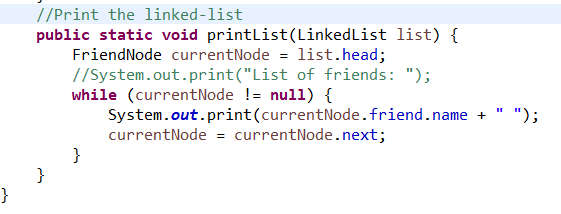


Class Linked-List:

Class Linked-List is a class that creates a data structure to contain a list of friends of each person. Here are some screenshots of this class.

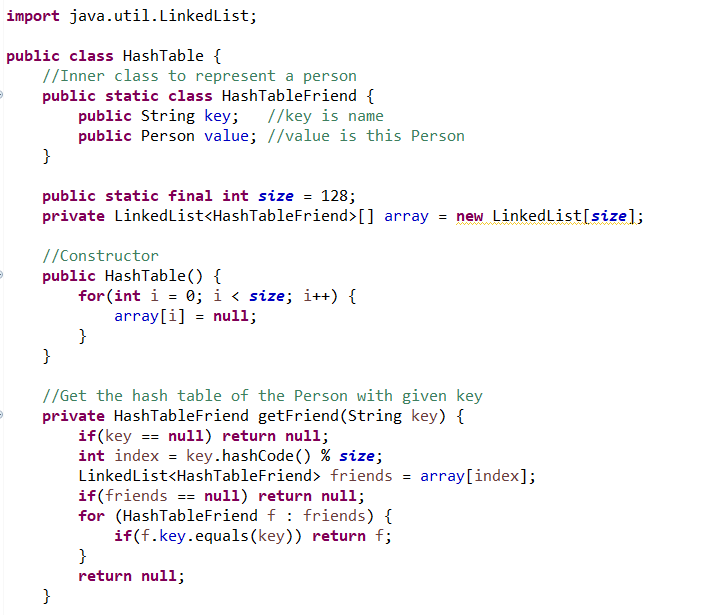


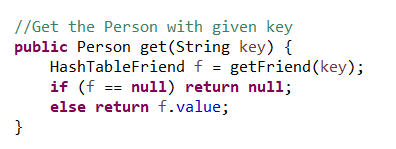




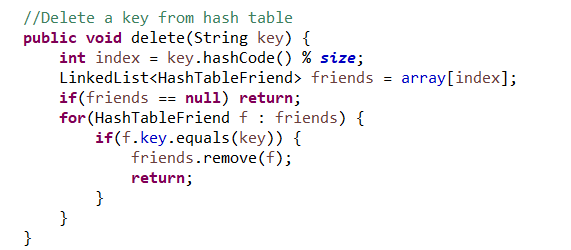
Class Hash-table:

Class Hash-table is a global hash table to manipulate the persons in linked-list. Here are some screen shots of this class.



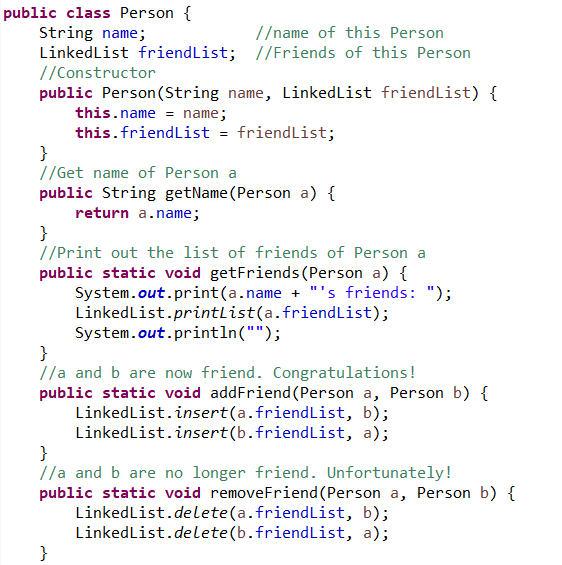


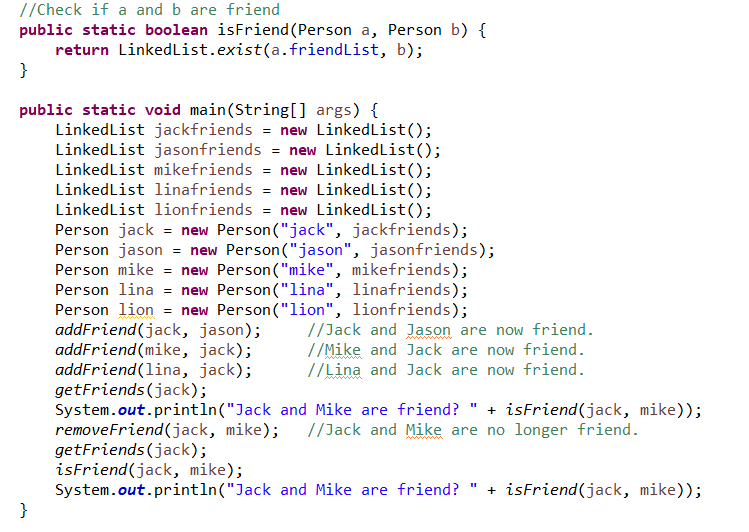




Class Person:

Class Person creates Person as an object that contains the name and the list of friends of a certain person. It is also an execute class of this program. Here are some screen shots of this class.

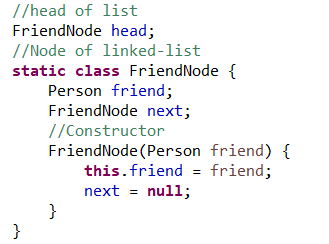




Functions:

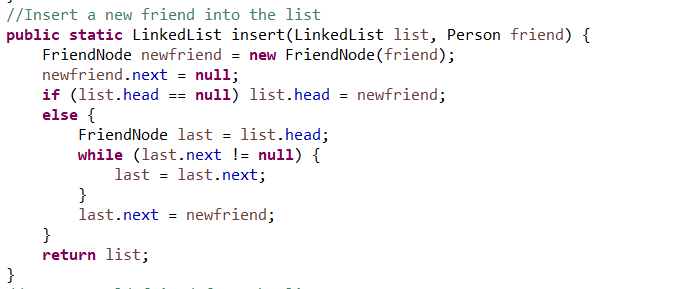
Class LinkedList:

Constructor FriendNode:



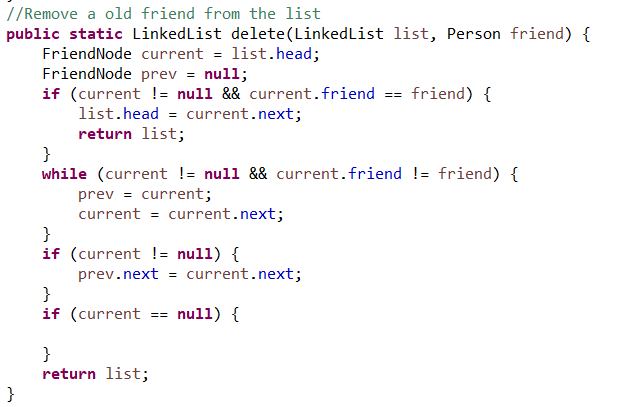
Friend-Node is an inner class of class LinkedList. It creates node of a person that can be store in the linked-list.

Function insert():



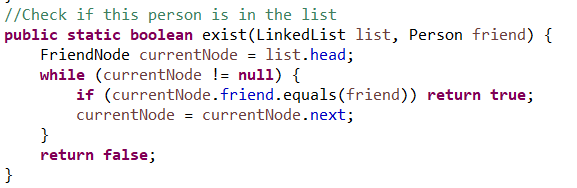
Insert is a function that inserts a Person into the linked-list.

Function delete():



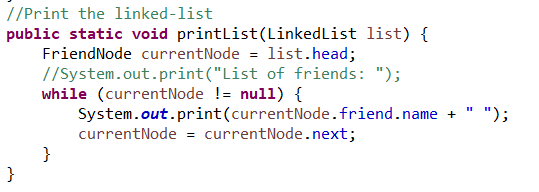
Delete is a function that deletes a person from linked-list.

Function exist():



Exist is a function that traverses the linked-list and checks if the given person is in the linked-list.

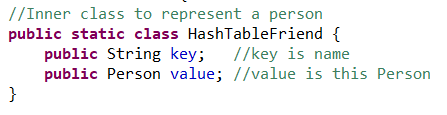
Function printList():



Print-List is a function that traverses over the whole list and prints out every person’s name in the list.

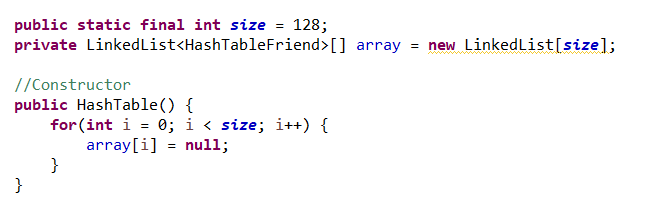
Class Hashtable:

Constructor Hash-table-friend:



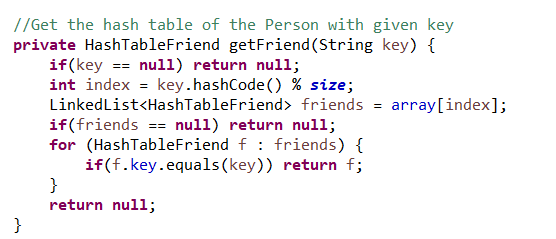
The constructor of friend node in the hash table.

Constructor Hash-table:



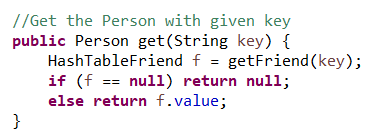
Creates an empty hash table.

Function get-Friend():



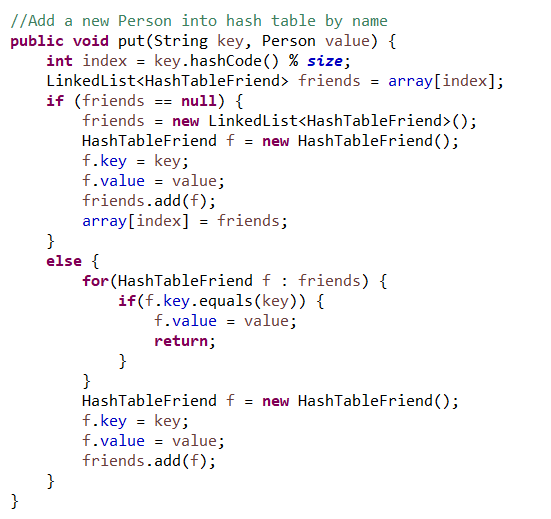
Return the friend node with the given name.

Function get():



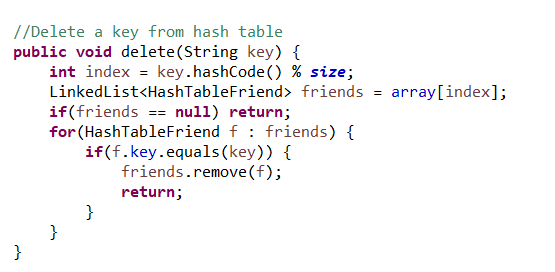
Return the Person with the given name.

Function put():



Put a Person with given name into the hash table.

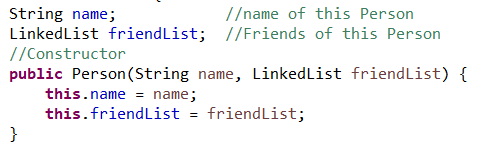
Put delete():



Delete is a function that remove a given name from the hash table.

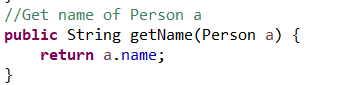
Class Person:

Constructor:



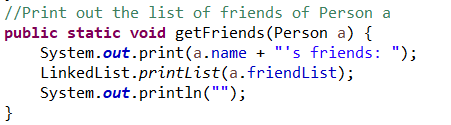
A person has his or her name and list of friends.

Function getName():



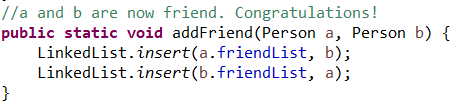
Return the name of given person.

Function getFriends():



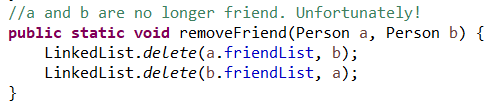
Print out all the friends of the given person.

Function addFriend():



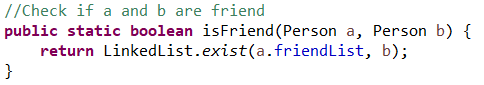
Make a and b become friend each other.

Function removeFriend():



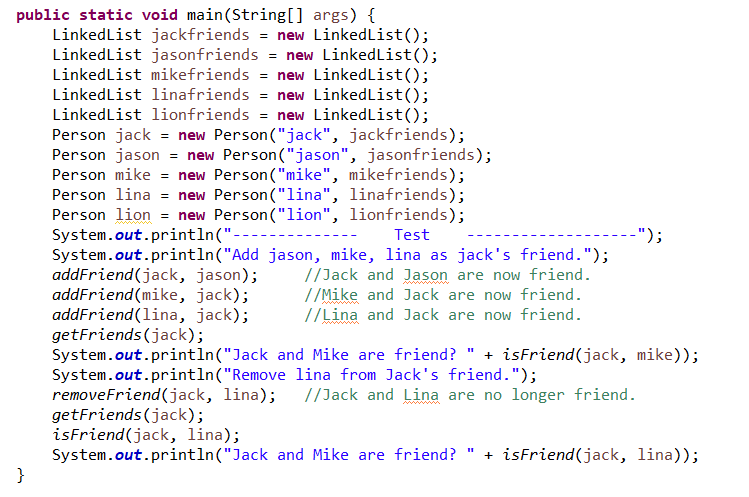
Make a and b no longer friend.

Function isFriend():



Return whether a and b are friend or not.

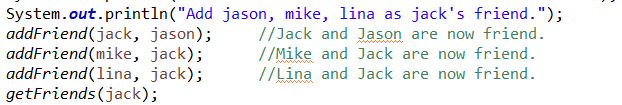
Main Method:

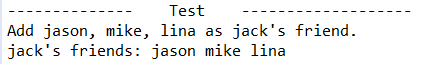


Execute and test the program.

Here are the screen shots of result:

1. Test insert:



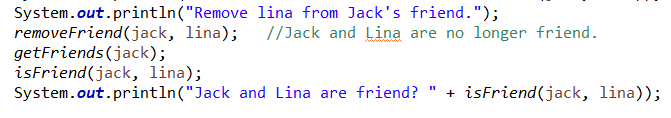


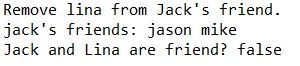
2. Test check method:





3. Test delete:





All functions are work.

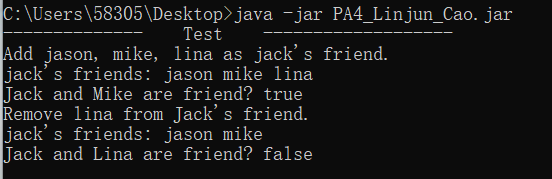
Part III. The procedure of how to unzip files, install application, and run/test codes:

First, unzip the zip and you will see a jar file called: PA4\_Linjun\_Cao.jar.

Then run the command and code in:

java -jar PA4\_Linjun\_Cao.jar

And the jar file will run itself.



The result as showing.

Part IV. Problems encountered during the implementation:

When I implemented delete function in linked-list, I meet with some problems. When I try to delete the first name in hash table and list or the last name, there are always bugs.

Also, when I implemented the hash table, one thing confuses me is what should be put in the hash table? In the textbook, it is key and value. But in this assignment, things are different. Finally I choose key as the name and value as the Person and it is work.

Part V. Lessons Learned:

After I meet problem with the linked-list, I divide the problem into many corner cases and solve these cases one by one. When the Person is in the middle, thing will be easy. When the person is the head, we should set a new head. Also, when the person is at the tail, we will have a new tail, etc.