Tuong Chu

tuong.chu@sjsu.edu

Abstract

This program is a micro-version of Facebook. In this program, the user will be able to create profiles, add friends, remove friends, and so on.

[pROgramming assignment 4]

CS146-S7

Contents

[Specification 2](#_Toc532604096)

[Design 2](#_Toc532604097)

[Running Instruction 3](#_Toc532604098)

[Implementation 4](#_Toc532604099)

[Main.java 5](#_Toc532604100)

[LinkedHash.java 8](#_Toc532604101)

[Map.java 9](#_Toc532604102)

[Person.java 11](#_Toc532604103)

[Test Case 1 12](#_Toc532604104)

[Problem Encountered: 22](#_Toc532604105)

[Lessons Learned: 22](#_Toc532604106)

# Specification

This program assignment aims to make a micro-version of Facebook. In this program, you will be able to create profiles, check someone’s friend list, add friends, remove friends, and so on.

This program will use the Hashing with Chaining method provided from page 258 in the textbook in order to save the data. Specifically, the program must provide these options for the user to choose:

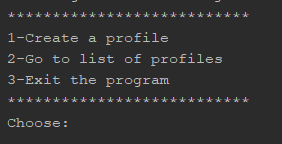
1. Create a person record of the specified name.
2. Record a person as a new friend. (Make a friend)
3. Remove a person from the friend list. (Unfriend)
4. List friends of a person.
5. Search a person’s name to list his/her friend list.
6. Enter two person’s names to check whether the two people are friends. (Print “Yes” or “No”)

More importantly, the program must create a global hash table that indexes each profile under the name and stores the friends of each person a linked list. A linked list must be created from scratch.

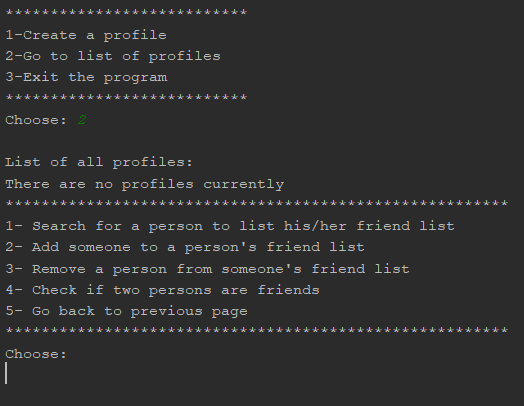
# Design

In terms of user interface:

The user can choose one from the 3 options on the terminal:



If the user picks 2, the program jumps to the next page where more options will be shown:



In terms of data structure, there will be three classes for the program:

* Class Person: This class creates Person objects that represent the Facebook profiles. Each Person object has one field for the name, one field for the linked list of friends, one field that points to the next Person object, and another field for the key. The key is derived from the first letter of the person’s name and converted into an integer. The key will be used to index the object into the hash table.
* Class LinkedHash: This class creates a linked list of Person objects which particularly contains a nil and head node. When a LinkedHash object is called, an empty linked list is created in which head is the nil node. This class also contains methods such as adding a node and removing a node.
* Class Map: This class creates a hash table of a specified size and each index in the table contains a single linked list of Person objects. This class will contain the three main methods that are associated with Hashing with Chaining: Chained\_Hash\_Insert, Chained\_Hash\_Search, Chained\_Hash\_Delete. Additionally, these methods will also use some of the methods in the LinkedHash class.
* Chained\_Hash\_Insert: insert a Person object to the hash table, particularly to the head of a LinkedHash linked list. The function will take the key (the 4th field of the class) of the Person object and divide by the size of the table. The remainder will be the index of the object. If other Person objects have the same index with this one, they will be appended to the head of the linked list.
* Chained\_Hash\_Delete: delete a Person object from a LinkedHash linked list. The function will iterate through the list until it finds the object and removes it from the list.
* Chained\_Hash\_Search: search for a Person object in the hash table. The function will iterate through each linked list in the table and each object in the list until it finds the object.

# Running Instruction

Run the program with file .jar:

* Extract folder Tuong\_Chu\_PA4 from Tuong\_Chu\_PA4.zip to anywhere you like
* Open the folder
* Copy the path of the fb.jar file. In cmd, type: java -jar then paste the path you just copied.
* The path depends on where you put the .jar file. For example: "java -jar C:\Users\kumaq\←- Documents\Tuong\_Chu\_PA4\fb.jar"

# Implementation

There are 5 different files for the program:

**Main.java**

**Person.java**

**LinkedHash.java**

**Map.java**

## Main.java

package Facebook;  
  
import java.io.IOException;  
import java.util.LinkedList;  
import java.util.Scanner;  
import Facebook.Map;  
  
*/\*\*  
 \* Main class  
 \*/*public class Main {  
 */\*\*  
 \* Main options for the user to choose  
 \** ***@return*** *an integer  
 \*/* public static int Options(){  
 int input;  
 Scanner reader = new Scanner(System.*in*);  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.*out*.print("1-Create a profile\n");  
 System.*out*.print("2-Go to list of profiles\n");  
 System.*out*.print("3-Exit the program\n");  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.*out*.print("Choose: ");  
 input = Integer.*valueOf*(reader.next());  
 while (input < 1 || input > 3){  
 System.*out*.print("Invalid input!\n");  
 }  
 return input;  
 }  
  
 */\*\*  
 \* Other options for user to choose after they go to the profile list  
 \*/* public static void otherOptions(){  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.*out*.print("1- Search for a person to list his/her friend list\n");  
 System.*out*.print("2- Add someone to a person's friend list\n");  
 System.*out*.print("3- Remove a person from someone's friend list\n");  
 System.*out*.print("4- Check if two persons are friends\n");  
 System.*out*.print("5- Go back to previous page\n");  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.*out*.print("Choose:\n");  
 }  
  
 */\*\*  
 \* Contains actions such as search for a profile, add friend, remove friend, etc.  
 \** ***@param*** *list a Map object that contains the hash table  
 \*/* public static void Actions(Map list){  
 Scanner reader = new Scanner(System.*in*);  
 int input = 1;  
 while (input>0 && input < 5){  
 *otherOptions*();  
 input = Integer.*valueOf*(reader.next());  
 if (input == 1){ // case 1, search for a person  
 *searchProfile*(list);  
 }  
 else if (input == 2){ // case 2, add friend  
 *addFriend*(list);  
 }  
 else if (input == 3){ // case 3, remove friend  
 *removeFriend*(list);  
 }  
 else if(input == 4){ // action 4, check if 2 people are friends  
 *checkIfFriend*(list);  
 }  
 }  
 }  
  
 */\*\*  
 \* Check if two people are friends.  
 \** ***@param*** *list a Map object that contains the hash table  
 \*/* public static void checkIfFriend(Map list){  
 Scanner reader = new Scanner(System.*in*);  
 System.*out*.print("Person 1: ");  
 String input1 = reader.next();  
 System.*out*.print("Person 2: ");  
 String input2 = reader.next();  
 // check if the 2 people are in the hash table  
 Person p1 = list.Chained\_Hash\_Search(input1);  
 Person p2 = list.Chained\_Hash\_Search(input2);  
  
 if (p1 !=null && p2!=null){ // if they are, check their friend lists  
 if (p1.getFriendList().search(input2) !=null){  
 System.*out*.print("Yes. " + input1 + " and " + input2 + " are friends\n");  
 }  
 else{ System.*out*.print("No. " + input1 + " and " + input2 + " are not friends\n");}  
  
 }  
 else {System.*out*.print("Oops! One of the persons does not exist\n");}  
 }  
  
 */\*\*  
 \* Add a person as friend to someone's friend list.  
 \** ***@param*** *list a Map object that contains the hash table  
 \*/* public static void addFriend(Map list) {  
 Scanner reader = new Scanner(System.*in*);  
 System.*out*.print("Add this person: ");  
 String input = reader.next();  
 System.*out*.print("To this person's friend list:");  
 String input2 = reader.next();  
 // Check if the 2 people are in the hash table  
 Person p1 = list.Chained\_Hash\_Search(input);  
 Person p2 = list.Chained\_Hash\_Search(input2);  
  
 if (p1 != null && p2 != null) { // if they are, add them together  
 p1.getFriendList().addFirst(input2);  
 p2.getFriendList().addFirst(input);  
 }  
 }  
  
 */\*\*  
 \* Remove a person from someone's friend list  
 \** ***@param*** *list a Map object that contains the hash table  
 \*/* public static void removeFriend(Map list){  
 Scanner reader = new Scanner(System.*in*);  
 System.*out*.print("Remove this person: ");  
 String input = reader.next();  
 System.*out*.print("From this person's friend list:");  
 String input2 = reader.next();  
 // use chained hash to delete  
 list.Chained\_Hash\_Delete(input,input2);  
  
 }  
  
 */\*\*  
 \* Create a profile for a person  
 \** ***@param*** *list a Map object that contains the hash table  
 \*/* public static void createProfile(Map list){  
 Scanner reader = new Scanner(System.*in*);  
 System.*out*.print("Enter name:"); // get the person's name  
 String input = reader.next();  
 System.*out*.print(input + "'s profile is successfully created!\n");  
 list.Chained\_Hash\_Insert(input); // insert the profile into the hash table  
 }  
  
 */\*\*  
 \* Search for a profile in the hash table  
 \** ***@param*** *list a Map object that contains the hash table  
 \*/* public static void searchProfile(Map list) {  
 Scanner reader = new Scanner(System.*in*);  
 System.*out*.print("Enter a name you want to search:");  
 String input = reader.next();  
 Person p = list.Chained\_Hash\_Search(input);  
 // check if that person is in the hash table  
 if (p!= null) { // if yes, go to their friend list  
 System.*out*.println("Name: " + p.getName());  
 System.*out*.print(p.getName() + "'s friend list:\n");  
 if (p.getFriendList().getSize() > 0) {  
 p.getFriendList().printList(); // print friend list  
 } else {  
 System.*out*.print(p.getName() + " currently has no friends.\n");  
 }  
 }  
 }  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\* MAIN \*\*\*\*\* FUNCTION \*\*\*\*\*\*\*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* public static void main(String[] args) {  
 // write your code here  
 Map profileList = new Map();  
 int input = 1;  
 while (input != 3) {  
 input = *Options*();  
 if (input == 1) {  
 // create a profile  
 *createProfile*(profileList);  
 } else if (input == 2) {  
 // go to list of profiles  
 profileList.printTable();  
 // other actions  
 *Actions*(profileList);  
 }  
 }

## LinkedHash.java

package Facebook;  
  
*/\*\*  
 \* This class creates a LinkedList of Person objects  
 \*/*public class LinkedHash {  
 private Person nil;  
 private Person head;  
  
 */\*\*  
 \* Default constructor  
 \*/* public LinkedHash(){ // creates an empty linked list  
 nil = new Person();  
 head = nil;  
 head.setNext(nil);  
 }  
  
 */\*\*  
 \* Get the size of the linked list  
 \** ***@return*** *an integer  
 \*/* public int getSize(){  
 int count = 0;  
 Person x = head;  
 while (x!=nil){  
 x = x.getNext();  
 count = count + 1;  
 }  
 return count;  
 }  
  
 */\*\*  
 \* Print the whole linked list  
 \*/* public void printList(){  
 Person x = head;  
 while (x != nil) {  
 System.*out*.print(x.getName() + "\n");  
 x = x.getNext();  
 }  
 }  
  
 */\*\*  
 \* Add a node to the head of a linked list  
 \** ***@param*** *s  
 \*/* public void addFirst(String s) {  
 Person p = new Person(s);  
 p.setNext(head);  
 head = p;  
 }  
  
 */\*\*  
 \* Search for a node in a linked list  
 \** ***@param*** *s Name of the person to be searched  
 \** ***@return*** *a Person object  
 \*/* public Person search(String s){  
 Person x = head;  
 while (x!=nil){  
 if (x.getName().equals(s)){  
 return x; // return the object if fount  
 }  
 x = x.getNext();  
 }  
 return null; // return null if the node cant be found  
 }  
  
 */\*\*  
 \* Remove a specific node in a linked list. Return true  
 \* if function successfully removes the node, else return false.  
 \** ***@param*** *s Name of the person to be removed  
 \** ***@return*** *true or false  
 \*/* public boolean remove(String s){  
 if (head==nil){ // if linked list is empty  
 System.*out*.print(s + "'s friend list is empty.\n");  
 return false;  
 }  
 if (head.getName().equals(s)){ // check the first node/head  
 head = head.getNext();  
 return true;  
 }  
 Person x = head;  
 while (x!=nil){ // check other nodes  
 if (x.getNext().getName().equals(s)){  
 Person z = x.getNext();  
 x.setNext(z.getNext());  
 return true;  
 }  
 else {  
 x = x.getNext();  
 }  
 }  
 return false;

## Map.java

package Facebook;  
  
import java.util.LinkedList;  
import Facebook.Person;  
  
*/\*\*  
 \* This class creates a hash table that indexes each profile under the name  
 \*/*public class Map {  
 private final static int *TABLE\_SIZE* = 11;  
 private LinkedHash[] table = new LinkedHash[*TABLE\_SIZE*];  
  
 */\*\*  
 \* Default constructor  
 \*/* public Map(){  
 //create an empty table  
 for (int i = 0; i < *TABLE\_SIZE*; i++){  
 table[i] = new LinkedHash();  
 }  
 }  
  
 */\*\*  
 \* Print the whole table from index 0 to 12. Each index contains  
 \* a linked list of Person objects.  
 \*/* public void printTable(){  
 boolean checkIfEmpty = true;  
 System.*out*.print("\nList of all profiles:\n");  
 for (int i = 0; i < *TABLE\_SIZE*; i++){  
 // each index i contains a linked list  
 LinkedHash items = table[i];  
 if (items.getSize() > 0) { // if linked list not empty  
 items.printList(); // print the list  
 checkIfEmpty = false;  
 }  
 }  
 if (checkIfEmpty){ // print error if list is empty  
 System.*out*.println("There are no profiles currently");  
 }  
 }  
  
 */\*\*  
 \* Insert a profile in the hash table using chained hashing. If two  
 \* elements have the same index, the second one will be appended to the  
 \* head of the linked list, at the same index.  
 \** ***@param*** *s The name of the person  
 \*/* public void Chained\_Hash\_Insert(String s){  
 Person p = new Person(s);  
 // divide the key by the table size to get index  
 int index = p.getKey() % *TABLE\_SIZE*;  
 LinkedHash list = table[index]; // get the linked list at that index  
 list.addFirst(s); // add the new profile to the linked list  
 table[index] = list; // update the linked list  
 }  
  
 */\*\*  
 \* Search for an element/profile in the hash table. Return true if found, false if not.  
 \** ***@param*** *n Name of the person to be searched  
 \** ***@return*** *a Person object  
 \*/* public Person Chained\_Hash\_Search(String n){  
 for(int i = 0; i < *TABLE\_SIZE*; i++){  
 LinkedHash items = table[i];  
 Person profile = items.search(n);  
 if (profile!=null){ // if the person is found  
 return profile;  
 }  
 }  
 // show error if cannot find the person  
 System.*out*.print("Sorry we cannot find " + n + " in the profile list.\n");  
 return null;  
 }  
  
 */\*\*  
 \* Remove a person from someone's friend list.  
 \** ***@param*** *n1 Name of person 1  
 \** ***@param*** *n2 Name of person 2  
 \*/* public void Chained\_Hash\_Delete(String n1, String n2){  
 // check if the two persons are in the hash table  
 Person p1 = Chained\_Hash\_Search(n1);  
 Person p2 = Chained\_Hash\_Search(n2);  
  
 if (p1 !=null && p2 !=null){ // if they are in the hash table  
 // remove each person off each other's friend list  
 if (p1.getFriendList().remove(n2) && p2.getFriendList().remove(n1))  
 { System.*out*.print(n1 + " and " + n2 + " are no longer friends.\n");}  
 else{ // if one of them is not on the other's friend list  
 System.*out*.print("Oops! One of the persons is not in the other's friend list.\n");  
 }  
 }  
  
 }  
}

## Person.java

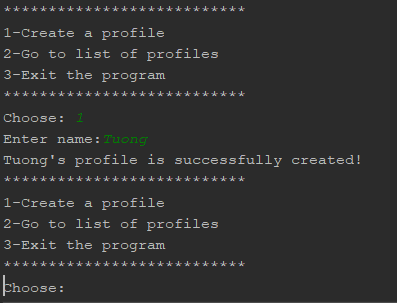
package Facebook;  
  
import java.util.LinkedList;  
import static java.lang.Integer.*MIN\_VALUE*;  
  
*/\*\*  
 \* This class creates Person objects that represent the profiles  
 \*/*public class Person {  
 private String name;  
 private int key;  
 private LinkedHash friendList;  
 private Person next;  
  
 */\*\*  
 \* Default Constructor  
 \*/* public Person(){  
 this.name = "";  
 this.friendList = null;  
 this.next = null;  
 this.key = *MIN\_VALUE*;  
 }  
  
 */\*\*  
 \* Another constructor that creates a Person object for  
 \* a person by the name. It takes the first letter of their name  
 \* and converts it to a number, which will be used as the key.  
 \** ***@param*** *s Name of a person  
 \*/* public Person(String s){  
 this.name = s;  
 this.friendList = new LinkedHash();  
 this.next = null;  
 this.key = 0;  
 for (int i = 0; i < s.length(); i++){  
 this.key = this.key + (int) s.charAt(i);  
 }

}  
  
 */\*\*  
 \* Set a Person object to the next of another Person object  
 \** ***@param*** *p a Person object  
 \*/* public void setNext(Person p){  
 this.next = p;  
 }  
  
 */\*\*  
 \* Get the next of a Person object  
 \** ***@return*** *a Person object  
 \*/* public Person getNext(){return this.next;}  
  
 */\*\*  
 \* Get the key of a Person object  
 \** ***@return*** *an integer  
 \*/* public int getKey(){  
 return key;  
 }  
  
 */\*\*  
 \* Get the name of a Person object  
 \** ***@return*** *a string  
 \*/* public String getName(){  
 return name;  
 }  
  
 */\*\*  
 \* Get the friend list of a Person object  
 \** ***@return*** *a LinkedHash object  
 \*/* public LinkedHash getFriendList(){  
 return friendList;  
 }  
  
  
}

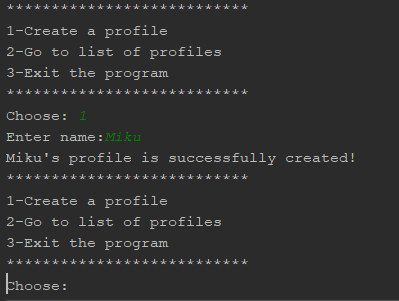
# Test Case 1

In this test case, I will create 30 profiles and show the list of profiles.

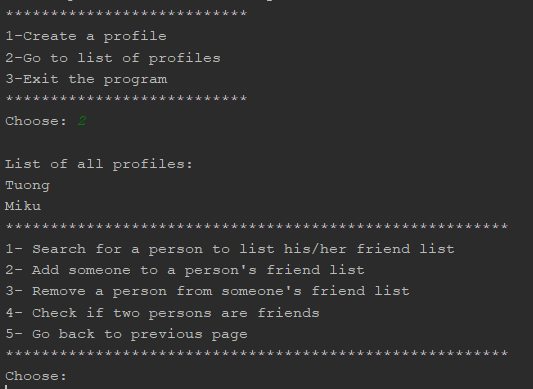
Create the first profile:



Create another:

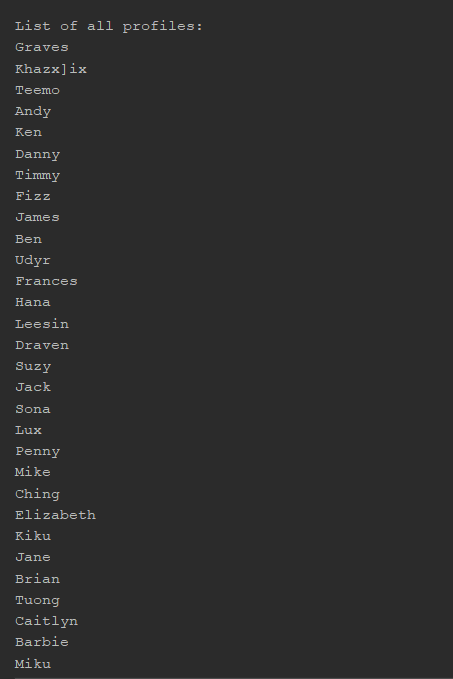


Check if the two profiles are in the hash table:

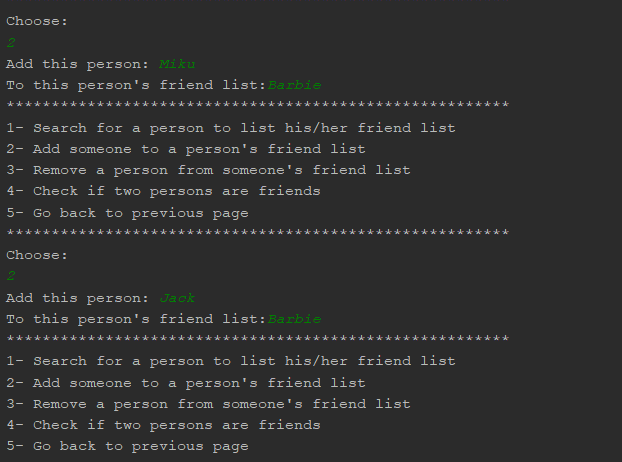


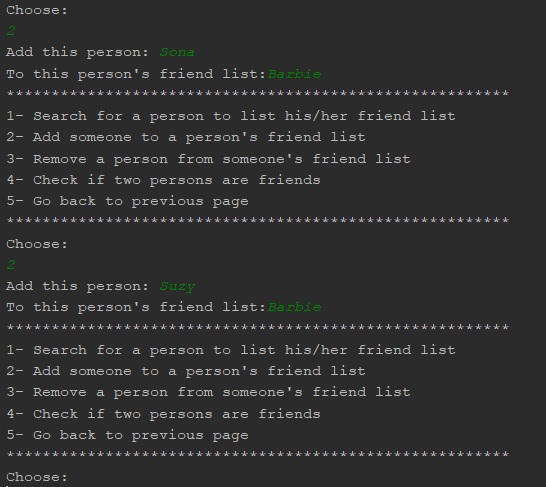
Tuong and Miku are in the hash table.

Let’s create more profiles until we have 30 profiles:

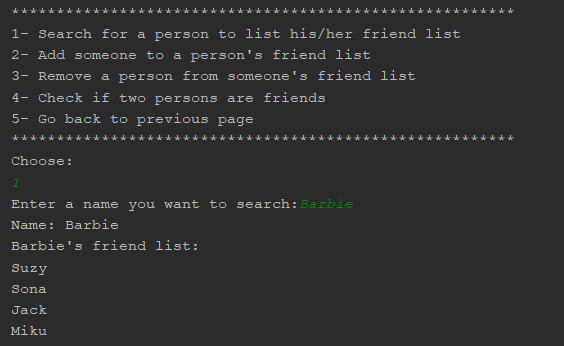


Now, let’s add Miku, Jack, Sona, and Suzy to Barbie’s friend list:

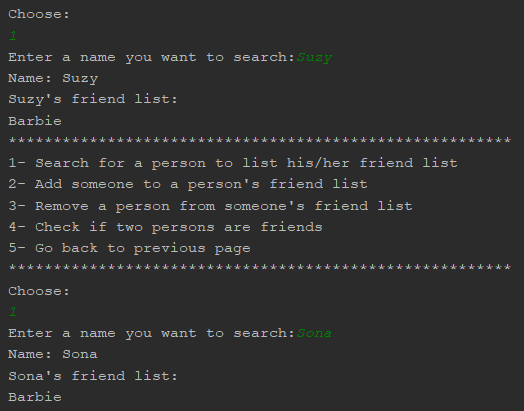




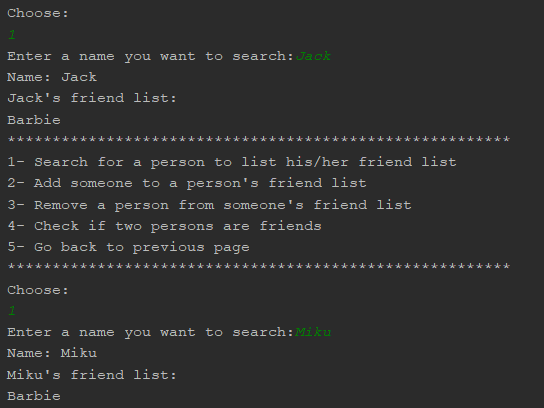
Check Barbie’s friend list to see if Suzy, Miku, Jack, and Sona are there:



They are all in Barbie’s friend list. Now, let’s check Suzy, Sona, Jack, Miku’s friend list to see if Barbie are there:

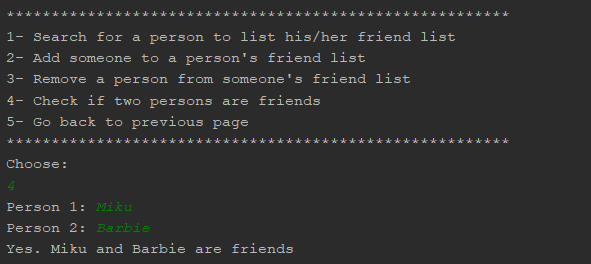


Both Suzy and Sona have Barbie in their friend lists.



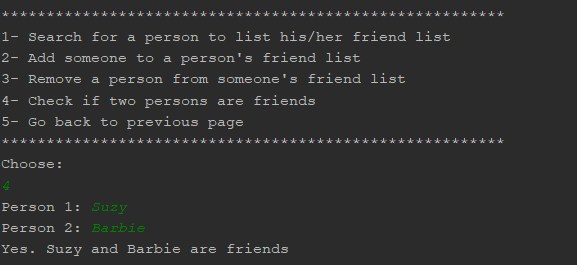
Amazing! They all have Barbie in their friend lists.

Now, let’s check again if Barbie and Miku are friends:



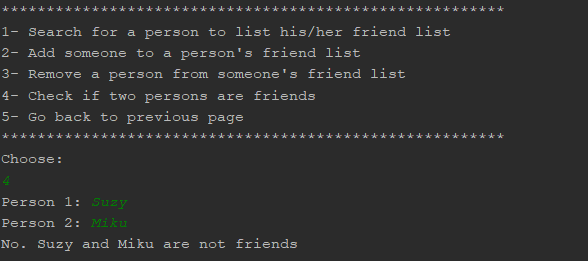
They are friends!

Check if Suzy and Barbie are friends:



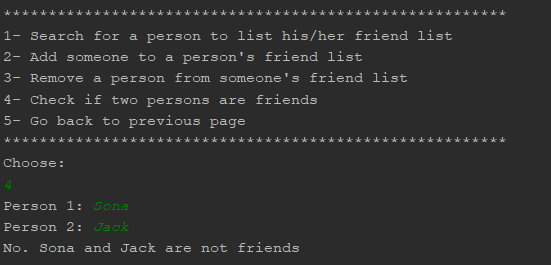
The result is correct as expected.

However, are Suzy and Miku friends?



No!

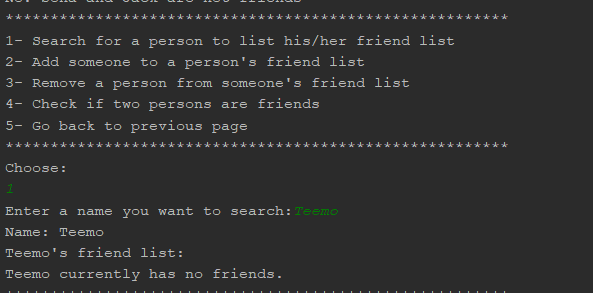
Are Sona and Jack friends?



No, they are also not friends!

This means that Hash\_Chained\_Insert and Hash\_Chained\_Search are working properly.

Let’s check a different person and their friend list:

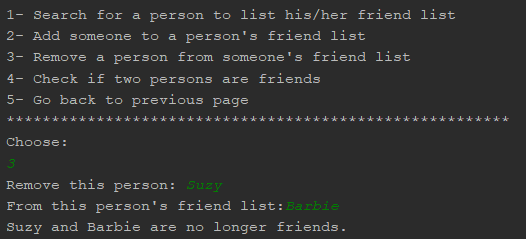


The result is correct because I never added anyone to Teemo’s friend list.

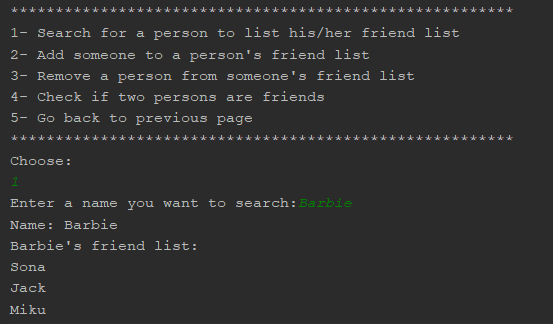
# Test Case 2

In this test case I will test Chained\_Hash\_Delete

Let’s delete Suzy from Barbie’s friend list:

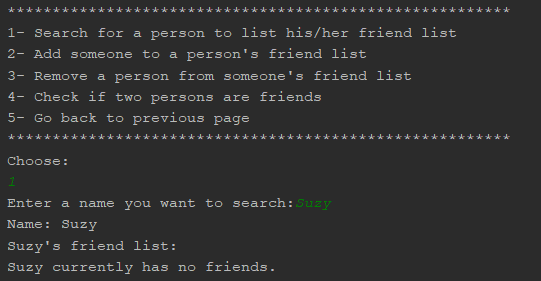


Check Barbie’s friend list to see if Suzy is there:



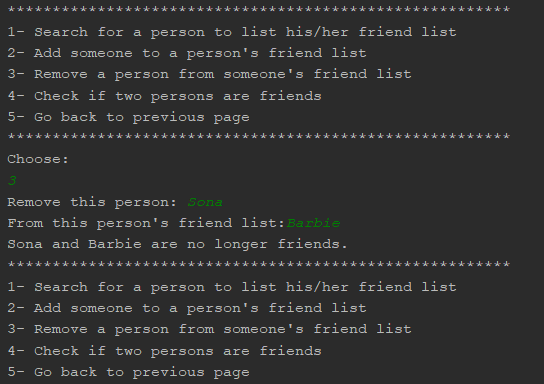
Suzy is removed from Barbie’s friend list.

Check Suzy’s friend list to see of Barbie is there:

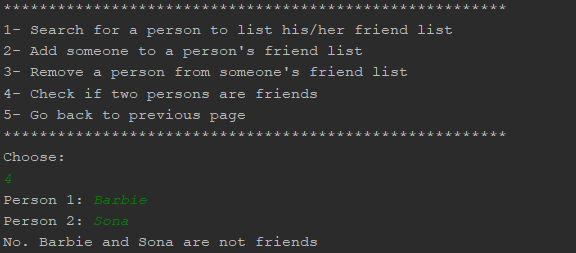


Barbie is not in Suzy friend list. Therefore, Suzy has no friends now.

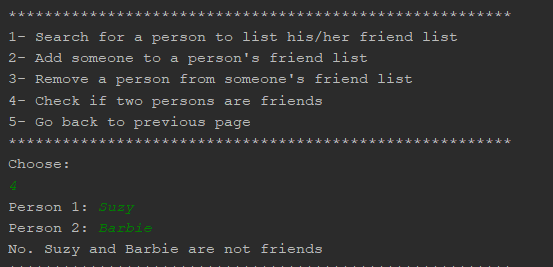
Remove Sona from Barbie’s friend list:



Now, let’s check if Sona and Barbie are still friends:



Continue to check if Suzy and Barbie are still friends since we removed Suzy from Barbie’s friend list:

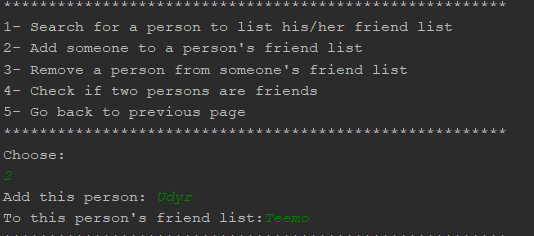


They are not friends. This means that Chained\_Hash\_Delete and Chained\_Hash\_Search are working properly.

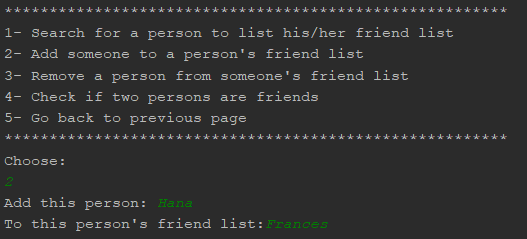
# Test Case 3

In this test case, I will add friends to other profiles.

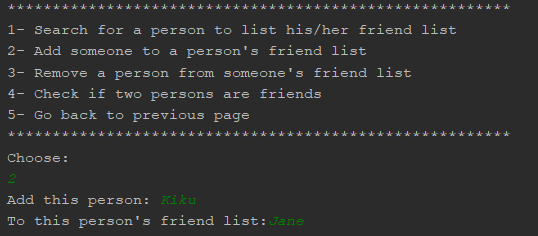
Udyr and Teemo:



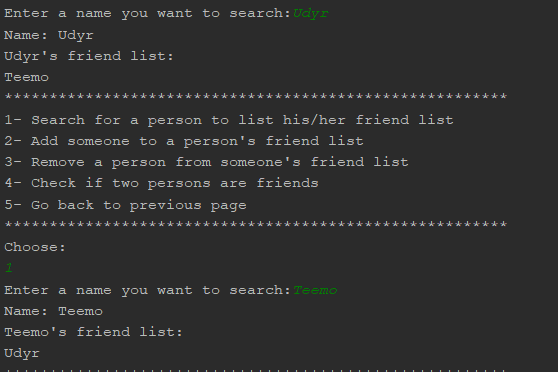
Hana and Frances:



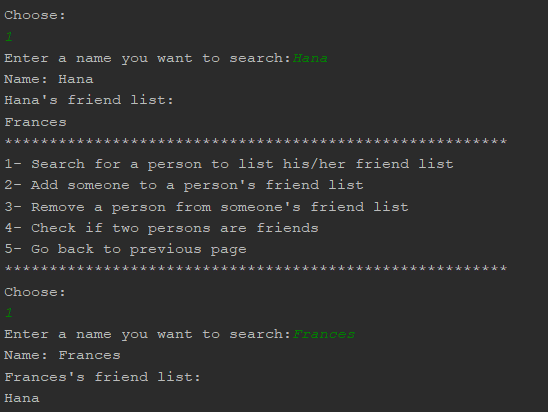
Kiku and Jane:



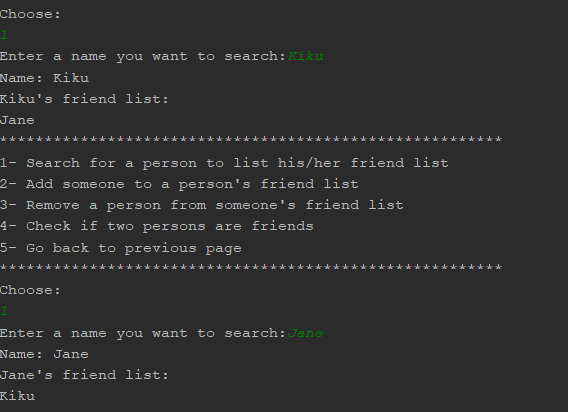
Let’s check each person’s friend list:



Udyr and Teemo are friends.



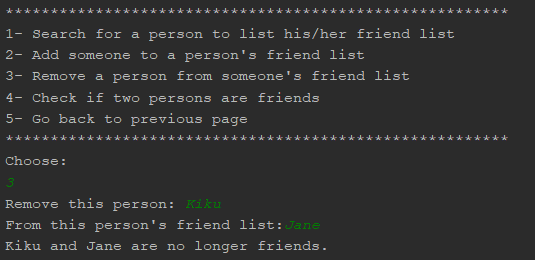
Hana and Frances are friends.



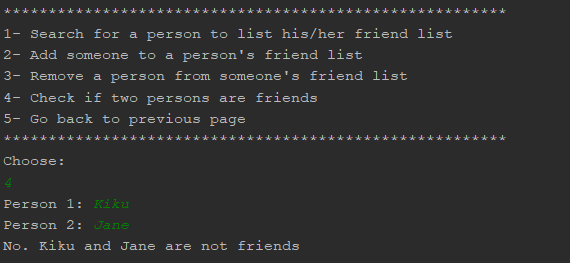
Finally, Kiku and Jane are also friends.

This means that Chained\_Hash\_Insert are working as intended.

Remove Kiku from Jane’s friend list:



Check whether if Kiku and Jane are still friends:



They are no longer friends. That means Chained\_Hash\_Delete are working prop

# Problem Encountered:

I had a lot of troubles trying to design the Person class and LinkedHash class. I thought I could use the LinkedList library for this assignment but I realized I was not allowed to do that wgen I was half-way through my program, so I had to redo everything again. Implementing my own Linked List class (Linked Hash) was not easy because I kept running into errors associated with Null Pointer Exception, so I had avoid making null objects, which was a bit complicated. When I got the null thing down, everything became easier.

# Lessons Learned:

While I learned about Hashing with Chaining, I got to enhance my knowledge about Linked List and null pointers. I think Hashing with Chaining is every effective in distributing elements into an array. It was fast and easy to implement as well. The only hard part was implementing a Linked List data structure for it