```
// Author: Jose Enriquez
// Email: jose.enriquez@okstate.edu
// 4/30/2021
// simulates a social network to demonstrate a Graph AdjacencyList structure
import java.util.*;

public class SocialNetwork {
    private static Scanner scan = new Scanner(System.in);
    private static BinarySearchTree bst = new BinarySearchTree();
    private static Graph graph = new Graph();
    private static HashMap<String, User> userCollection = new HashMap<String,
User>();

    /**
     * Prints a menu of possible operations for the user.
     */
    public static void printMenu(){
        System.out.println("Enter the corresponding number for the " +
        "action you want to perform.");
        System.out.println("1: Add a user in the network");
        System.out.println("2: Follow a user");
        System.out.println("3: Find all the followers of a user");
        System.out.println("4: Find all the followings of a user");
        System.out.println("5: Remove a follower of a user");
        System.out.println("6: Remove a user from the network");
        System.out.println("7: Exit");
    }

    /**
     * Adds the a new User
     */
    public static void addUser(){
        System.out.println("Enter the user name");
        String userName = scan.nextLine();

        while(true){
            //Continues until A valid ID is entered.
            System.out.println("Enter ID");
            String id = scan.nextLine();

            User newUser = new User(userName, id);

            if(!bst.search(newUser)){
                //ID is unique

                bst.insert(newUser);
                graph.addVertex(newUser);
                userCollection.put(id, newUser);
                //add to graph, binary search tree, and collection of users.
                break;
            }
            else{
                System.out.println("User ID has already been taken. Try again");
            }

        }
        System.out.println();
    }

    /**
     * Sets a user to follow another user.
     */
    public static void followUser(){
        System.out.println("Enter the ID of the follower and followee in the form of
(followerID, followeeID)");
        String[] input = scan.nextLine().split(", ");
        input[0] = input[0].substring(1, 6);
        input[1] = input[1].substring(0, 5);

        User follower = userCollection.get(input[0]);
        User followee = userCollection.get(input[1]);

        graph.addEdge(follower, followee);

    }

    /**
     * Prints the followers of a given User
     */
    public static void followers(){
        System.out.println("Enter name of the User to get its followers");
        String userName = scan.nextLine();

        System.out.println("Enter ID of the User to get its followers");
        String id = scan.nextLine();

        if(!userCollection.containsKey(id) ||
!userCollection.get(id).userName.equalsIgnoreCase(userName)){
            //Given username and or ID does not exist.
            System.out.println("User with username and ID: " + userName + "," + id +
" does not exist");
            return;
        }

        User user = userCollection.get(id);
        HashSet<User> followers = graph.getFolowers(user);
        Iterator<User> iter = followers.iterator();

        if(followers.isEmpty()){
            //user has no followers
            System.out.println(user.userName + " has no followers");
            return;
        }

        User temp = null;
        while(iter.hasNext()){
            //Goes through each follower.
            temp = iter.next();
            System.out.println(temp.userName + " is following " + user.userName);
        }
    }

    /**
     * Prints every User a given User is following.
     */
    public static void following(){
        System.out.println("Enter name of the User to find who they are following");
        String userName = scan.nextLine();

        System.out.println("Enter ID of the User to find who they are following");
        String id = scan.nextLine();

        if(!userCollection.containsKey(id) ||
!userCollection.get(id).userName.equalsIgnoreCase(userName)){
            //Given username and or ID does not exist.
            System.out.println("User with username and ID: " + userName + "," + id +
" does not exist");
            return;
        }

        User user = userCollection.get(id);
        HashSet<User> followers = graph.getFollowing(user);
        Iterator<User> iter = followers.iterator();

        if(followers.isEmpty()){
            //User is following no one.
            System.out.println(user.userName + " is following no one");
            return;
        }

        User temp = null;
        while(iter.hasNext()){
            //Goes through each User the given User is following
            temp = iter.next();
            System.out.println(user.userName + " is following " + temp.userName);
        }

    }

    /**
```

```
     * makes a User unfollow someone they are following
     */
    public static void unfollow(){
        System.out.println("Enter name of the follower");
        String followerName = scan.nextLine();

        System.out.println("Enter ID of the follower");
        String followerID = scan.nextLine();

        System.out.println("Enter name of the person to unfollow");
        String unfollowName = scan.nextLine();

        System.out.println("Enter ID of the person to unfollow");
        String unfollowID = scan.nextLine();

        if(!userCollection.containsKey(followerID) ||
!userCollection.get(followerID).userName.equalsIgnoreCase(followerName)){
            //Given username and or ID  of the follower does not exist.
            System.out.println("User with username and ID: " + followerName + "," +
followerID + " does not exist");
            return;
        }

        if(!userCollection.containsKey(unfollowID) ||
!userCollection.get(unfollowID).userName.equalsIgnoreCase(unfollowName)){
            //Given username and or ID of the followee does not exist.
            System.out.println("User with username and ID: " + unfollowName + "," +
unfollowID + " does not exist");
            return;
        }

        //Get referenced to the follower and followee
        User follower = userCollection.get(followerID);
        User unfollow = userCollection.get(unfollowID);

        //follower unfollows followee
        graph.removeEdge(follower, unfollow);
    }

    /**
     * Remove a given User
     */
    public static void removeUser(){
        System.out.println("Enter name of the User to remove");
        String userName = scan.nextLine();

        System.out.println("Enter ID of the User to remove");
        String id = scan.nextLine();

        if(!userCollection.containsKey(id) ||
!userCollection.get(id).userName.equalsIgnoreCase(userName)){
            //Given username and or ID does not exist.
            System.out.println("User with username and ID: " + userName + "," + id +
" does not exist");
            return;
        }

        //remove user from all things
        User user = userCollection.get(id);
        graph.removeVertex(user);
        userCollection.remove(user.userID);
        bst.delete(user);
    }


    public static void main(String[] args) {
        boolean flag = true;
        int choice = 0;

        while(flag){

            //prompts user with a list of operations
            printMenu();
            choice = scan.nextInt();
            scan.nextLine();

            switch(choice){
                case 1:
                    //add user to network.
                    addUser();
                    break;
                case 2:
                    //make a user follow another user
                    followUser();
                    break;
                case 3:
                    //print the followers of a user
                    followers();
                    break;
                case 4:
                    //print the Users someone is following.
                    following();
                    break;
                case 5:
                    //make a user unfollow another user
                    unfollow();
                    break;
                case 6:
                    //remove User form the network.
                    removeUser();
                    break;
                case 7:
                    //end program.
                    System.out.println("Goodbye");
                    flag = false;
                    break;
                default:
                    continue;
            }
        }
    }
}
```