```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.*;


public class FacebookNetwork {

  private static Map<String, LinkedList<String>> network = new HashMap<>();


  public static void main(String[] args) {

    // Example file path - Make sure this is correct on your machine

    String filePath = "C:\\Users\\Anusha\\OneDrive\\Desktop\\UserIDs.docx";  // Change if needed

    System.out.println("Starting data load...");  // Debugging

    loaddata(filePath);


    // Print out the most connected user

    System.out.println("Most connected user: " + getMostConnectedFriend());


    // Find and print pairs of users who don't know each other

    List<String> nonConnected = findPeopleWhoDontKnowEachOther();

    System.out.println("People who don't know each other: ");

    for (String pair : nonConnected) {

      System.out.println(pair);

    }

  }


  private static void loaddata(String filePath) {

    System.out.println("Inside loaddata()...");  // Debugging

    try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {

      String line;

      br.readLine(); // Skip header line

      while ((line = br.readLine()) != null) {
```

```java
            String[] parts = line.split(",", 3);
            if (parts.length < 3) {
                System.out.println("Skipping malformed line: " + line);  // Debugging
                continue;  // Skip lines that are incorrectly formatted
            }

            String userId = parts[0].trim();
            String friendList = parts[2].trim();
            friendList = friendList.replaceAll("\\s+", "");
            String[] friends = friendList.split(",");

            // Debugging: Print the user and their friends
            System.out.println("Loading user: " + userId + " with friends: " + Arrays.toString(friends));

            network.putIfAbsent(userId, new LinkedList<>());
            for (String friend : friends) {
                network.get(userId).add(friend);
                network.putIfAbsent(friend, new LinkedList<>());
                if (!network.get(friend).contains(userId)) {
                    network.get(friend).add(userId);
                }
            }
        }
    } catch (IOException e) {
        System.out.println("Error reading file: " + e.getMessage());
    }
}

private static String getMostConnectedFriend() {
    System.out.println("Inside getMostConnectedFriend()...");  // Debugging
    String mostConnected = null;
```

```
        int maxFriends = 0;
        for (Map.Entry<String, LinkedList<String>> entry : network.entrySet()) {
            int friendCount = entry.getValue().size();
            if (friendCount > maxFriends) {
                maxFriends = friendCount;
                mostConnected = entry.getKey();
            }
        }
        return mostConnected == null ? "No users found" : mostConnected + " has " + maxFriends + "
friends.";
    }

    private static List<String> findPeopleWhoDontKnowEachOther() {
        System.out.println("Inside findPeopleWhoDontKnowEachOther()...");  // Debugging
        List<String> nonConnectedPairs = new ArrayList<>();
        List<String> users = new ArrayList<>(network.keySet());
        for (int i = 0; i < users.size(); i++) {
            for (int j = i + 1; j < users.size(); j++) {
                String userA = users.get(i);
                String userB = users.get(j);
                if (!network.get(userA).contains(userB)) {
                    nonConnectedPairs.add(userA + " and " + userB);
                }
            }
        }
        return nonConnectedPairs;
    }

    private static boolean doesKnow(String personA, String personB) {
        LinkedList<String> friends = network.get(personA);
        return friends != null && friends.contains(personB);
    }
}
```