GTU CSE 222 HW-8 REPORT

Name : Yahya Kemal

Surname : Kuyumcu

ID : 220104004106

Program Structure :

Instead of using the demo code provided by the assistant in my program, I wanted to write everything from scratch myself, which made my work easier in many areas. Let's start by explaining the classes contained in the program.

Suggestion Class:

I used the `Suggestion` class, which is not included in the demo code, to store suggested friends and their score values. This class helped me list the top N people with the most common interests in descending order in the friend suggestion algorithm. It made my code more readable and simplified the ordering process.

Hobby Enum Class:

Like the `Suggestion` class, this class was also not included in the demo code. While writing the program, I realized that asking people to choose from predefined enums rather than taking and storing their hobbies as strings would result in fewer user errors (such as incorrect input or discrepancies between two strings). The `Hobby` enum class contains 20 predefined hobbies, and when adding a person to the network, I take hobbies as comma-separated index numbers starting from 1 instead of strings. For example, if a user selects hobbies as 1, 3, 4, 5, I add the corresponding hobbies to the `List<Hobby>` in each `Person` class. This not only simplified my work but also made the code more readable.

Cluster Class:

The last class I created in addition to the demo code is the `Cluster` class. Like the previous classes, I used this to keep cluster data in one place and make my work easier, resulting in more readable code. The `Cluster` class contains a variable `List<Person> peopleInCluster`, which holds all the people in the same cluster. I use the public method `addPerson(Person p)` to add people to the list and the public method `getPeople()` to directly get the list of people in the cluster.

CLI Class:

This class is the Command Line Interface of our program. It contains the `menu()` method and the `caller()` method. The `menu` method prints the menu as specified by the assistant and returns the user's selection to the `caller` method. The `caller` method continues to call the `menu` method as long as the returned number is not 8. Based on the returned value, the `caller` method calls the necessary method from the `SocialNetwork` object, which is stored as a variable in the `CLI` class.

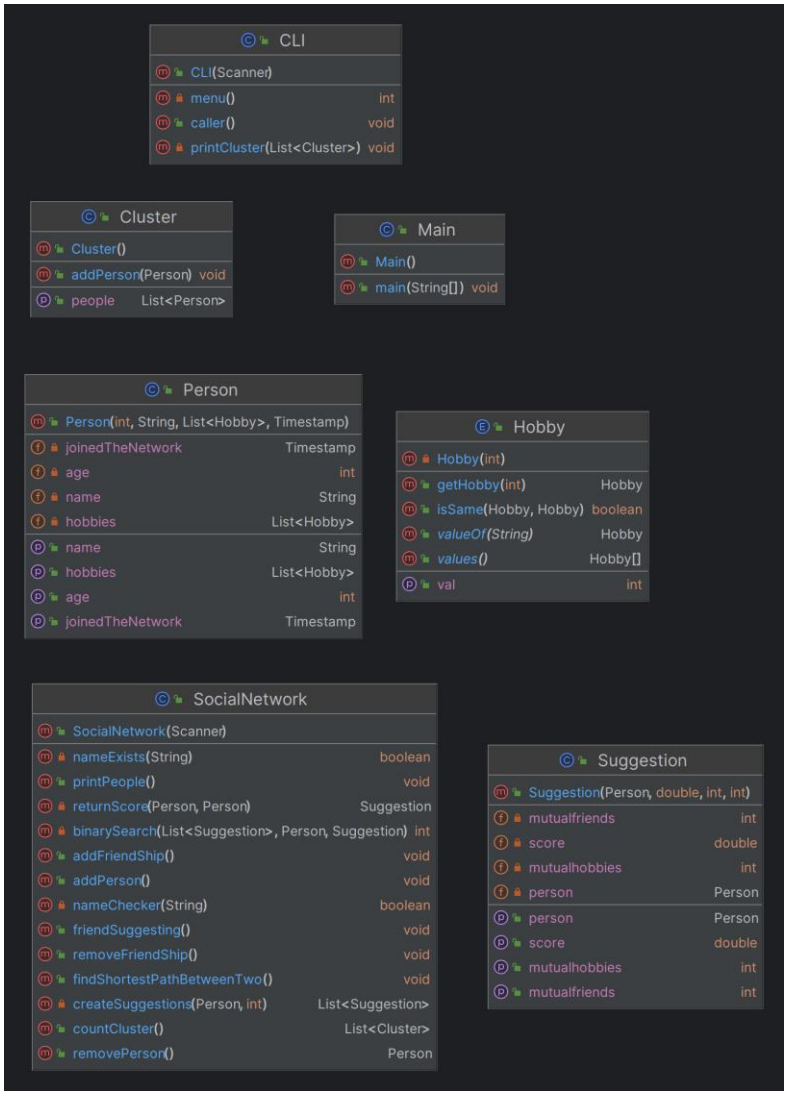
Person Class:

This `Person` class is not much different from the one provided in the demo code. As member variables, it stores the person's age, name, timestamp of joining the network, and hobbies in a `List<Hobby>`. Aside from the necessary getter and setter methods, there are no other methods in this class.

SocialNetwork Class:

The `SocialNetwork` class is the largest and main class of the program. All operations such as adding or removing a person, adding or removing friendships, finding and printing clusters, and suggesting friends are done through the methods in this class. I will explain the methods of this class on the next page, so I will skip the method descriptions for now.

Class Diagram of the Program:



Social Network Class's Methods:

addPerson(): After being called by the CLI, it collects information such as the name, age, and hobbies of the person the user wants to add. Then, it adds the created `Person` object to the `people` hashmap of the `SocialNetwork` class, with the person's name as the key and the person object itself as the value.

removePerson(): After being called by the CLI, it collects the name and the timestamp of when the person was added to the network. If such a person is found in `people`, it first removes the person from `people`, then takes the keyset from the `graph` and removes this person from all their friends' friend lists. Finally, it updates the graph.

addFriendship(): It collects the names and timestamp information of the two people who will be friends. If these person objects exist in the network and are not already friends, it adds each of them to the other's friend list and then updates the graph.

removeFriendship(): The person-finding part works almost the same as in `addFriendship`. However, it removes the two people from each other's friend lists instead and updates the graph with the new lists (where each person is a key and their friend list is the value).

findShortestPathBetweenTwo():

This method is used to find the shortest path between two people. First, it collects the names of the two people and the timestamps of when they joined the network from the user. After verifying this information, it uses the Breadth-First Search (BFS) algorithm to find the shortest path between the two people.

1. **Collecting Start and End Persons:**
    - The method asks the user for the names and timestamps of the two people.
    - After verifying this information, it creates `startPerson` and `endPerson` objects.
2. **BFS Algorithm:**
    - It creates a queue (`Queue<Person>`) and a parent map (`Map<Person, Person>`).
    - `startPerson` is added to the queue and marked with a null parent in the parent map.
    - The loop continues until the queue is empty:
        - The current person is dequeued.
        - If the current person is `endPerson`, the loop breaks.
        - The current person's friends are checked, and unvisited friends are added to the queue and the parent map.
3. **Extracting the Shortest Path:**
    - If `endPerson` is not in the parent map, it indicates that there is no path between the two people.
    - Otherwise, starting from `endPerson`, the method follows the parents to find the shortest path and adds it to a list.
    - The shortest path list is reversed, and the shortest path between the two people is printed.
4. **Error Handling:**
    - If an invalid timestamp format is provided or the specified person cannot be found, appropriate error messages are displayed.

This method provides a user-friendly interface and uses the BFS algorithm to find the shortest path between two people.

friendSuggesting():

This method suggests friends to a specified person based on common interests and mutual friends. It works by first gathering the user's name and timestamp of joining the

network to verify their identity. After validation, it prompts the user to input the number of friend suggestions they want.

The method uses several helper methods:

- **createSuggestions(Person personToBeSuggested, int suggestionCount):** This method generates a list of friend suggestions. It calculates a similarity score for each potential friend using the `returnScore` method, which assesses common hobbies and mutual friends. The suggestions are then sorted in descending order using a binary search.
- **binarySearch(List<Suggestion> list, Person personToBeSuggested, Suggestion sug):** This method inserts each suggestion into the correct position in the sorted list to maintain order.
- **returnScore(Person basePerson, Person toBeFriend):** This method calculates the similarity score based on common hobbies and mutual friends, returning a `Suggestion` object.

Finally, the `friendSuggesting` method prints the top friend suggestions, including the score, mutual friends, and common hobbies for each suggested friend. This approach ensures the most relevant friend recommendations are presented to the user.

countCluster():

This method calculates the clusters in the social network by performing a depth-first search (DFS) traversal. It iterates over all people in the network and identifies unvisited individuals to start a new cluster. For each unvisited person, it initiates a new cluster and explores their friends using a queue-based approach. The method adds all connected individuals to the cluster and marks them as visited to prevent duplicate processing. Finally, it adds the completed clusters to a list and returns the list of clusters found in the network. This approach ensures that each cluster contains all individuals connected through friendships.

OUTPUTS

1) Add Person:

```
Please select an option: 1
Adding a person to the social network please give the following details about person.
Name : Yahya
Age : 12
Hobby : SPORTS MUSIC READING TRAVELING GAMING COOKING PAINTING DANCING WRITING PHOTOGRAPHY GARDENING
Please give the hobbys  numbers starting from 1 with ',' seperated.
1,2,3,4,5
Person created succesfully.
Credentials:
Name : Yahya
Age : 12
Hobbies: [SPORTS, MUSIC, READING, TRAVELING, GAMING]
Created Time : 2024-05-29 20:06:16.032
```

```
Please select an option: 1
Adding a person to the social network please give the following details about person.
Name : Derya
Age : 21
Hobby : SPORTS MUSIC READING TRAVELING GAMING COOKING PAINTING DANCING WRITING PHOTOGRAPHY GARDENING FISHING HIKING CYCLING
Please give the hobbys  numbers starting from 1 with ',' seperated.
3,4,5,6,9,12
Person created succesfully.
Credentials:
Name : Derya
Age : 21
Hobbies: [READING, TRAVELING, GAMING, COOKING, WRITING]
Created Time : 2024-05-29 20:06:39.015
```

2) Remove Person:

```
Please select an option: 2
For removing a person from the social network please give the following details about person.
Name : Sam

Give the timestamp of when the person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:59.103
Sam removed from the social network succesfully.
```

3)Add Friendship

```
Please select an option: 3
For adding a friendship between two people please give the following details.
Name of the first person : Derya

Timestamp of when the first person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:39.015
Name of the second person : Yahya

Timestamp of when the second person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:16.032
Friendship added between Derya and Yahya succesfully.
```

```
Please select an option: 3
For adding a friendship between two people please give the following details.
Name of the first person : John

Timestamp of when the first person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:08:30.093
Name of the second person : Sam

Timestamp of when the second person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:59.103
Friendship added between John and Sam succesfully.
```

4)Remove Friendship

```
Please select an option: 4
For removing a friendship between two people please give the following details.
Name of the first person : Derya

Timestamp of when the first person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:39.015
Name of the second person : Yahya

Timestamp of when the second person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:16.032
Friendship removed between Derya and Yahya succesfully.
```

5)Find the Shortest Path

```
Please select an option: 5
For finding the shortest path between two people please give the following details.
Name of the first person : Sam

Timestamp of when the first person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:59.103
Name of the second person : Derya

Timestamp of when the second person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:39.015
No path exists between the two people.
```

This is correct there are no friendships between these two, it is seen in the coming ss's.

One more Shortest Path

```
Please select an option: 5
For finding the shortest path between two people please give the following details.
Name of the first person : Derek

Timestamp of when the first person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:08:54.468
Name of the second person : Sam

Timestamp of when the second person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:59.103
Shortest path between Derek and Sam : Derek -> John -> Sam
```

6)Suggest Friends

```
Please select an option: 6
For suggesting friends to a person please give the following details.
Name of the person : Yahya

Timestamp of when the second person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:06:16.032
How many friend suggestions do you want to get : 3
John (Score: 2,0, 0 mutual friends, 4 common hobbies)
Derya (Score: 1,5, 0 mutual friends, 3 common hobbies)
Derek (Score: 1,5, 0 mutual friends, 3 common hobbies)
```

```
Please select an option: 6
For suggesting friends to a person please give the following details.
Name of the person : John

Timestamp of when the second person joined the network(format is like this : yyyy-MM-dd HH:mm:ss.SSS) : 2024-05-29 20:08:30.093
How many friend suggestions do you want to get : 3
Derya (Score: 2,0, 0 mutual friends, 4 common hobbies)
Yahya (Score: 2,0, 0 mutual friends, 4 common hobbies)
Derek (Score: 1,5, 0 mutual friends, 3 common hobbies)
```

7)Count Clusters

```
Please select an option: 7
Number of clusters: 3
Cluster 1:
Derya
Yahya
Cluster 2:
John
Derek
Sam
Cluster 3:
Murat
```