CSE 222 HOMEWORK 8: Graphs

Gebze Technical University Computer Engineering Department

Submission Deadline: 29th May 2024 23:59

ÇAĞRI YILDIZ – 1901042630

Part 1: Detailed Explanation of the Program

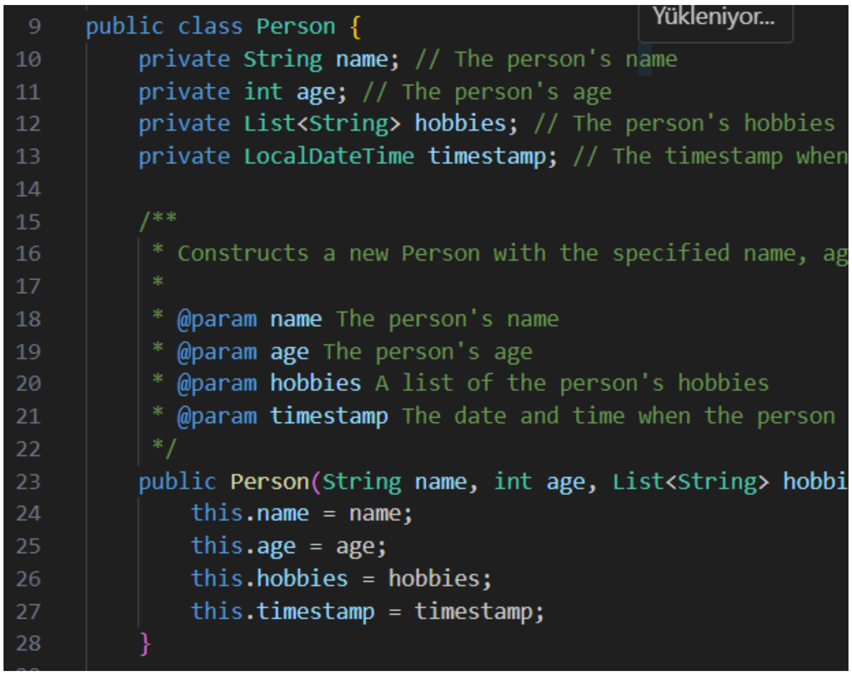
Overview

This project involves implementing a social network analysis system using graph data structures and algorithms. The goal is to simulate a social network and perform extensive analyses on the network, such as suggesting friends and counting clusters.

Class Hierarchy and Core Features

1. **Person Class**

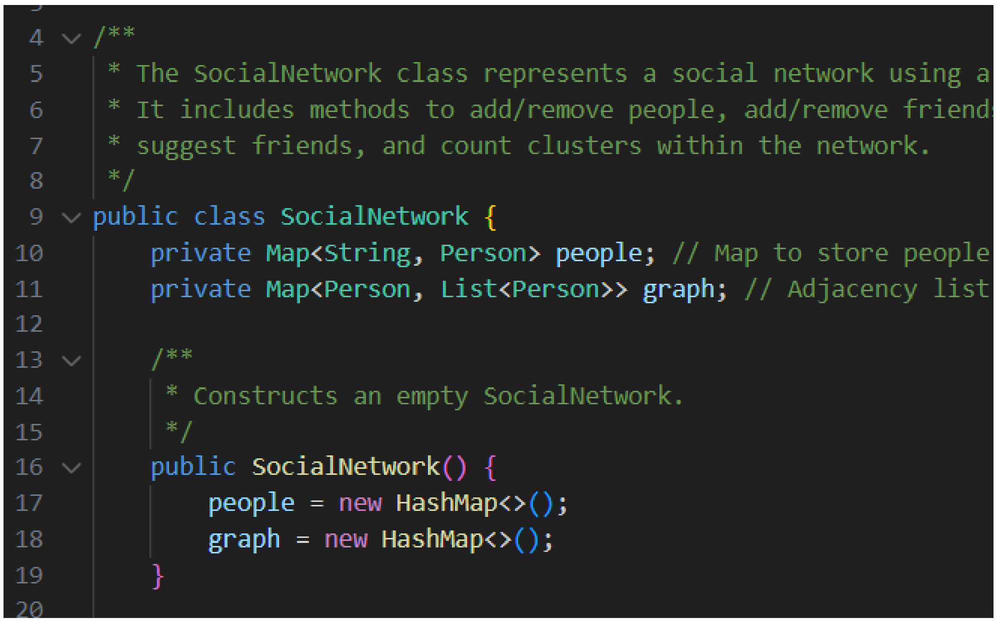
- **Attributes**: Name, Age, Hobbies, Timestamp.
- **Core Methods:**
    - Constructor: Initializes a person with the given attributes.
    - Getters: Retrieve individual attributes.
    - toString(): Provides a string representation of the person.



```java
public class Person {
    private String name; // The person's name
    private int age; // The person's age
    private List<String> hobbies; // The person's hobbies
    private LocalDateTime timestamp; // The timestamp when

    /**
     * Constructs a new Person with the specified name, ag
     *
     * @param name The person's name
     * @param age The person's age
     * @param hobbies A list of the person's hobbies
     * @param timestamp The date and time when the person
     */
    public Person(String name, int age, List<String> hobbi
        this.name = name;
        this.age = age;
        this.hobbies = hobbies;
        this.timestamp = timestamp;
    }
```

2. **SocialNetwork Class**

- **Attributes:** People (Map<String, Person>), Graph (Map<Person, List<Person>>).
- **Core Methods:**
    - addPerson: Adds a person to the network.
    - removePerson: Removes a person and all associated friendships.
    - addFriendship: Adds a friendship between two people.
    - removeFriendship: Removes a friendship between two people.
    - findShortestPath: Uses Breadth-First Search (BFS) to find the shortest path between two people.
    - suggestFriends: Suggests friends based on mutual friends and common hobbies.
    - countClusters: Counts the number of clusters using a BFS-based algorithm.



```java
/**
 * The SocialNetwork class represents a social network using a
 * It includes methods to add/remove people, add/remove friend
 * suggest friends, and count clusters within the network.
 */
public class SocialNetwork {
    private Map<String, Person> people; // Map to store people
    private Map<Person, List<Person>> graph; // Adjacency list

    /**
     * Constructs an empty SocialNetwork.
     */
    public SocialNetwork() {
        people = new HashMap<>();
        graph = new HashMap<>();
    }
```

Design Decisions

- **Graph Representation:** Used adjacency lists to efficiently represent friendships.
- **Data Structures**: Chose HashMap for people to allow quick lookups and ArrayList for friends to maintain an adjacency list.
- **Handling Duplicate Names:** Used a combination of name and timestamp as a unique identifier to handle duplicates.

Challenges and Solutions

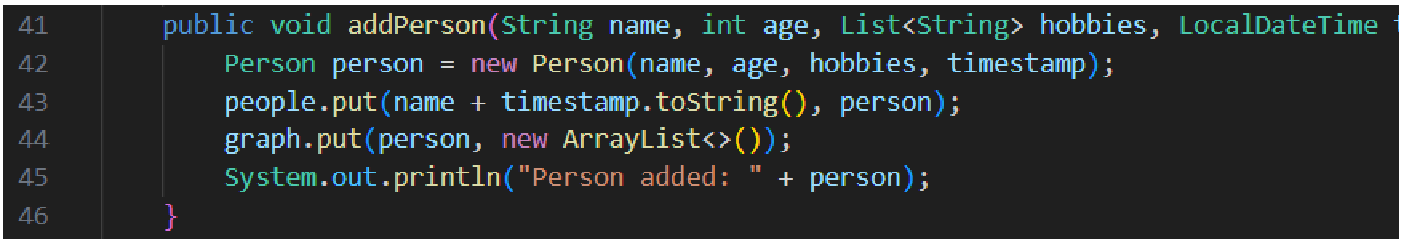
- **Timestamp Handling:** Adjusted timestamps to exclude nanoseconds for simplicity.
- **Error Handling:** Implemented comprehensive error handling to manage invalid inputs and ensure smooth program execution.

Part 2: Compilation and Execution

- **Compilation** => Command: make
- **Running the Program** => Command: make run
- **Generating Javadoc** => Command: make javadoc
- **Testing the code** => Command: make test

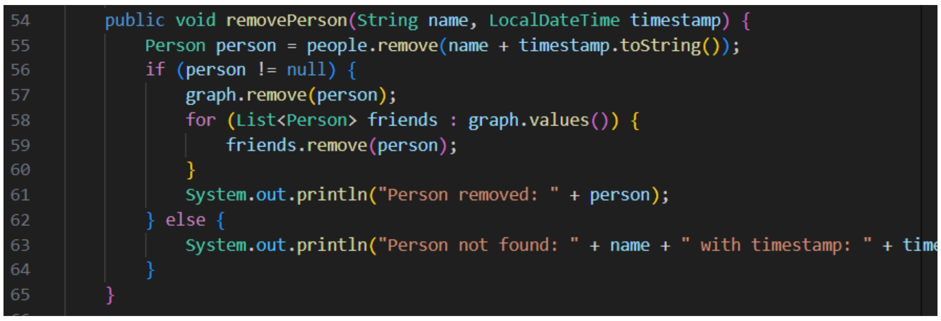
Part 3: Program Flow

1. **Adding a Person**
    - Method: addPerson
    - Description: Adds a new person to the social network with the current timestamp.



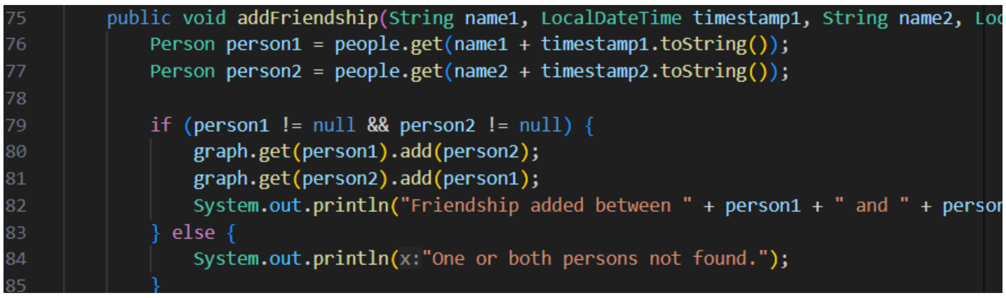
```java
    public void addPerson(String name, int age, List<String> hobbies, LocalDateTime t
        Person person = new Person(name, age, hobbies, timestamp);
        people.put(name + timestamp.toString(), person);
        graph.put(person, new ArrayList<>());
        System.out.println("Person added: " + person);
    }
```

2. **Removing a Person**
    - Method: removePerson
    - Description: Removes a person from the social network based on their name and timestamp, and removes all associated friendships.



```java
    public void removePerson(String name, LocalDateTime timestamp) {
        Person person = people.remove(name + timestamp.toString());
        if (person != null) {
            graph.remove(person);
            for (List<Person> friends : graph.values()) {
                friends.remove(person);
            }
            System.out.println("Person removed: " + person);
        } else {
            System.out.println("Person not found: " + name + " with timestamp: " + tim
        }
    }
```

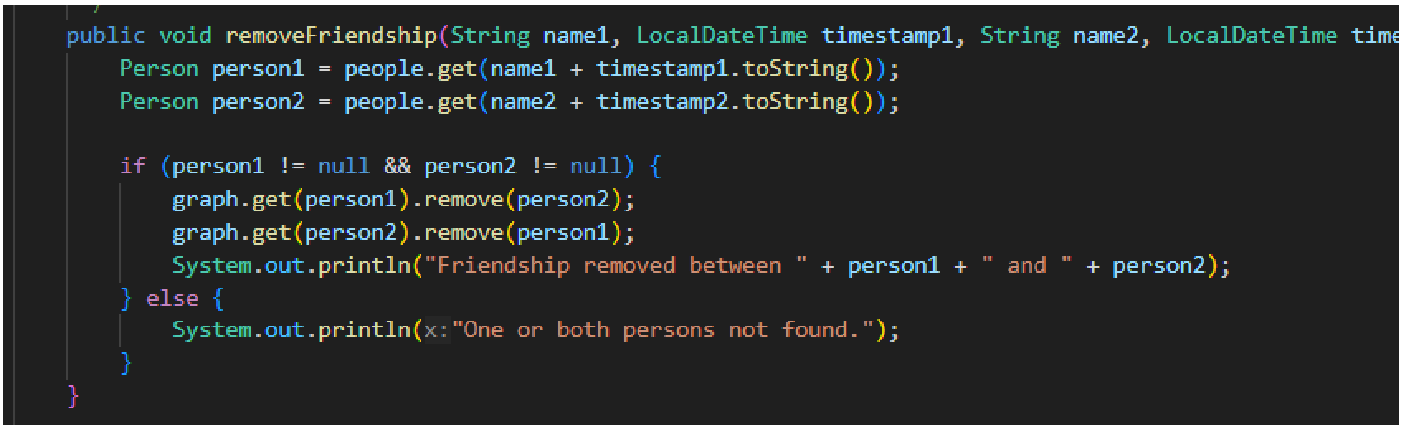
3. **Adding a Friendship**
    - Method: addFriendship
    - Description: Adds a friendship between two people in the network based on their names and timestamps.



```java
    public void addFriendship(String name1, LocalDateTime timestamp1, String name2, Lo
        Person person1 = people.get(name1 + timestamp1.toString());
        Person person2 = people.get(name2 + timestamp2.toString());

        if (person1 != null && person2 != null) {
            graph.get(person1).add(person2);
            graph.get(person2).add(person1);
            System.out.println("Friendship added between " + person1 + " and " + perso
        } else {
            System.out.println(x:"One or both persons not found.");
        }
```

4. Removing a Friendship

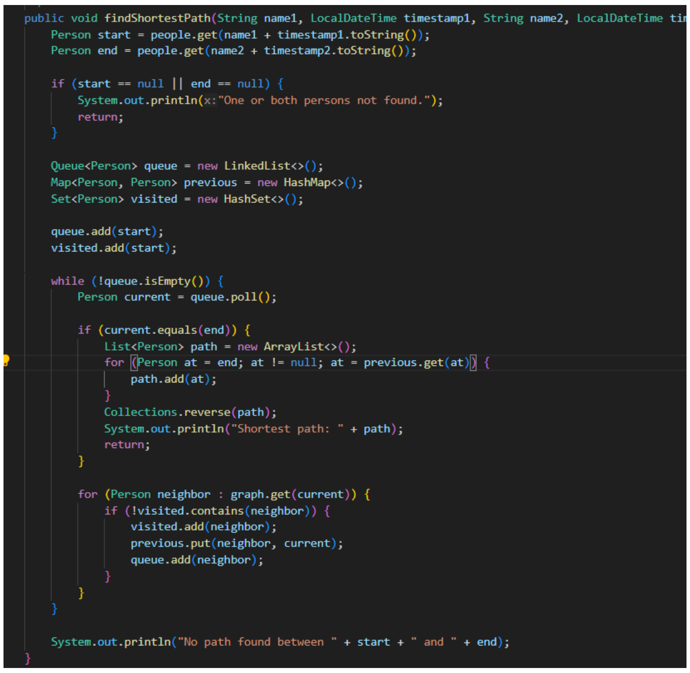
- Method: removeFriendship
- Description: Removes a friendship between two people in the network based on their names and timestamps.



```java
public void removeFriendship(String name1, LocalDateTime timestamp1, String name2, LocalDateTime time
    Person person1 = people.get(name1 + timestamp1.toString());
    Person person2 = people.get(name2 + timestamp2.toString());

    if (person1 != null && person2 != null) {
        graph.get(person1).remove(person2);
        graph.get(person2).remove(person1);
        System.out.println("Friendship removed between " + person1 + " and " + person2);
    } else {
        System.out.println(x:"One or both persons not found.");
    }
}
```

5. Finding Shortest Path

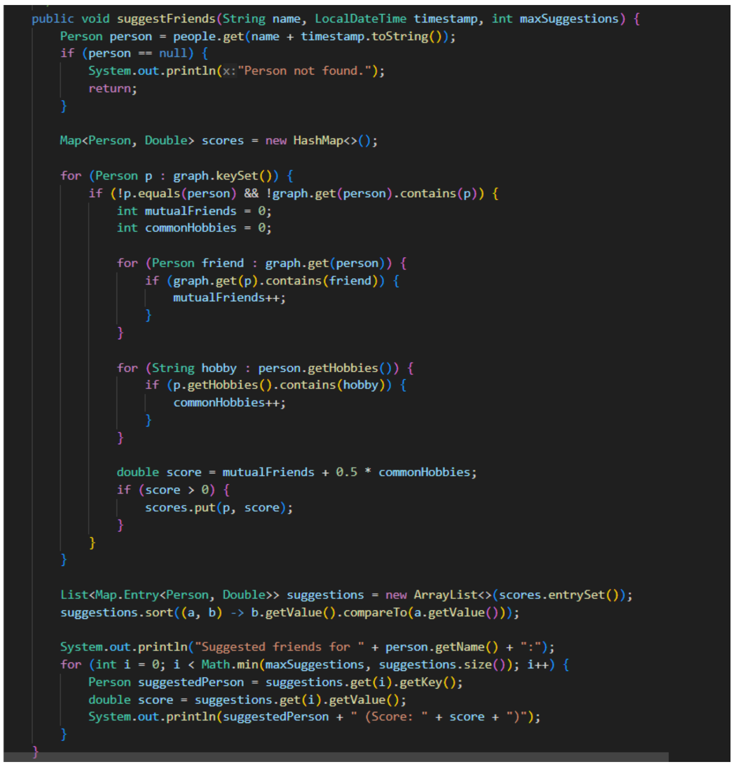
- Method: findShortestPath
- Description: Uses Breadth-First Search (BFS) to determine the shortest path between two people in the network.



```java
public void findShortestPath(String name1, LocalDateTime timestamp1, String name2, LocalDateTime ti
    Person start = people.get(name1 + timestamp1.toString());
    Person end = people.get(name2 + timestamp2.toString());

    if (start == null || end == null) {
        System.out.println(x:"One or both persons not found.");
        return;
    }

    Queue<Person> queue = new LinkedList<>();
    Map<Person, Person> previous = new HashMap<>();
    Set<Person> visited = new HashSet<>();

    queue.add(start);
    visited.add(start);

    while (!queue.isEmpty()) {
        Person current = queue.poll();

        if (current.equals(end)) {
            List<Person> path = new ArrayList<>();
            for (Person at = end; at != null; at = previous.get(at)) {
                path.add(at);
            }
            Collections.reverse(path);
            System.out.println("Shortest path: " + path);
            return;
        }

        for (Person neighbor : graph.get(current)) {
            if (!visited.contains(neighbor)) {
                visited.add(neighbor);
                previous.put(neighbor, current);
                queue.add(neighbor);
            }
        }
    }

    System.out.println("No path found between " + start + " and " + end);
}
```

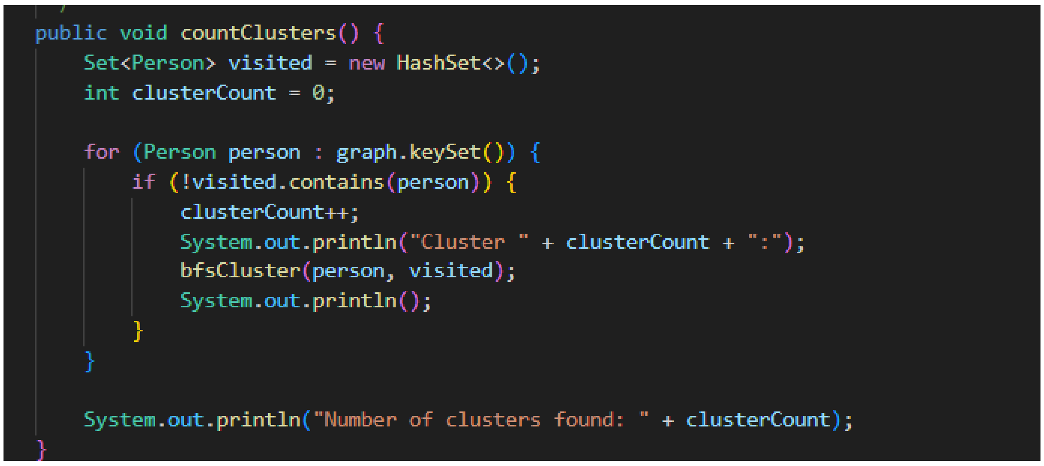
6. Suggesting Friends

- Method: suggestFriends
- Description: Suggests friends for a person based on mutual friends and common hobbies, calculated as: score = mutualFriends * 1 + commonHobbies * 0.5.



```java
public void suggestFriends(String name, LocalDateTime timestamp, int maxSuggestions) {
    Person person = people.get(name + timestamp.toString());
    if (person == null) {
        System.out.println(x:"Person not found.");
        return;
    }

    Map<Person, Double> scores = new HashMap<>();

    for (Person p : graph.keySet()) {
        if (!p.equals(person) && !graph.get(person).contains(p)) {
            int mutualFriends = 0;
            int commonHobbies = 0;

            for (Person friend : graph.get(person)) {
                if (graph.get(p).contains(friend)) {
                    mutualFriends++;
                }
            }

            for (String hobby : person.getHobbies()) {
                if (p.getHobbies().contains(hobby)) {
                    commonHobbies++;
                }
            }

            double score = mutualFriends + 0.5 * commonHobbies;
            if (score > 0) {
                scores.put(p, score);
            }
        }
    }

    List<Map.Entry<Person, Double>> suggestions = new ArrayList<>(scores.entrySet());
    suggestions.sort((a, b) -> b.getValue().compareTo(a.getValue()));

    System.out.println("Suggested friends for " + person.getName() + ":");
    for (int i = 0; i < Math.min(maxSuggestions, suggestions.size()); i++) {
        Person suggestedPerson = suggestions.get(i).getKey();
        double score = suggestions.get(i).getValue();
        System.out.println(suggestedPerson + " (Score: " + score + ")");
    }
}
```

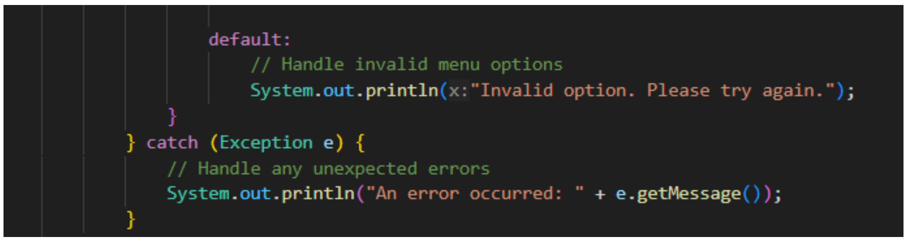
7. **Counting Clusters**
   - Method: countClusters
   - Description: Uses Breadth-First Search (BFS) to count and display the number of clusters in the network.



```java
public void countClusters() {
    Set<Person> visited = new HashSet<>();
    int clusterCount = 0;

    for (Person person : graph.keySet()) {
        if (!visited.contains(person)) {
            clusterCount++;
            System.out.println("Cluster " + clusterCount + ":");
            bfsCluster(person, visited);
            System.out.println();
        }
    }

    System.out.println("Number of clusters found: " + clusterCount);
}
```

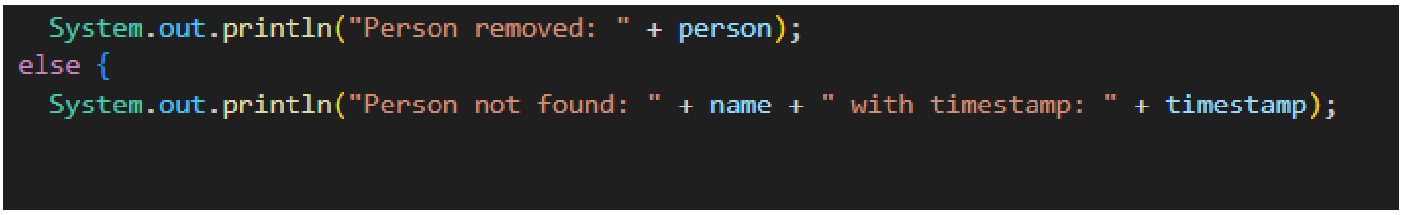
Part 4: Error Handling and Edge Cases

1. Invalid Inputs
   - Implemented error handling mechanisms to display informative error messages to the user.



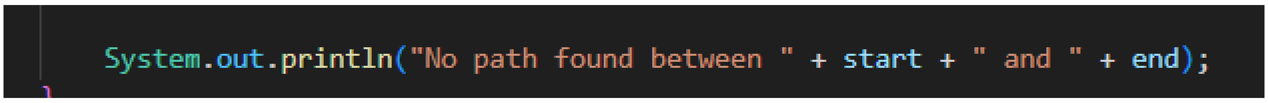
```java
                default:
                    // Handle invalid menu options
                    System.out.println(x:"Invalid option. Please try again.");
            }
        } catch (Exception e) {
            // Handle any unexpected errors
            System.out.println("An error occurred: " + e.getMessage());
        }
```

2. **Missing Person or Friendship**
   - If a person or friendship is not found during removal, a message is displayed indicating the issue.



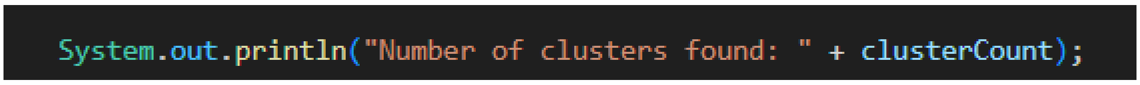
```java
  System.out.println("Person removed: " + person);
else {
  System.out.println("Person not found: " + name + " with timestamp: " + timestamp);
```

3. No Path Between Two People
   - When finding the shortest path, if no path exists, a message is displayed.



```java
    System.out.println("No path found between " + start + " and " + end);
```

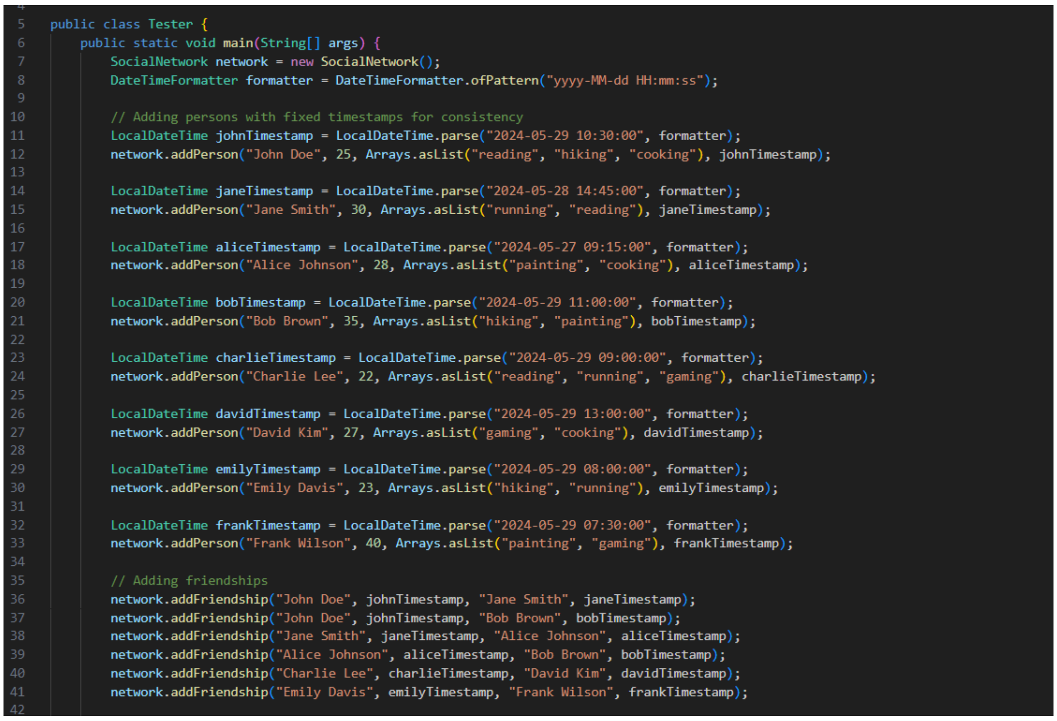
4. Empty Clusters
   - If an empty cluster is encountered while counting clusters, it is included in the count and documented.



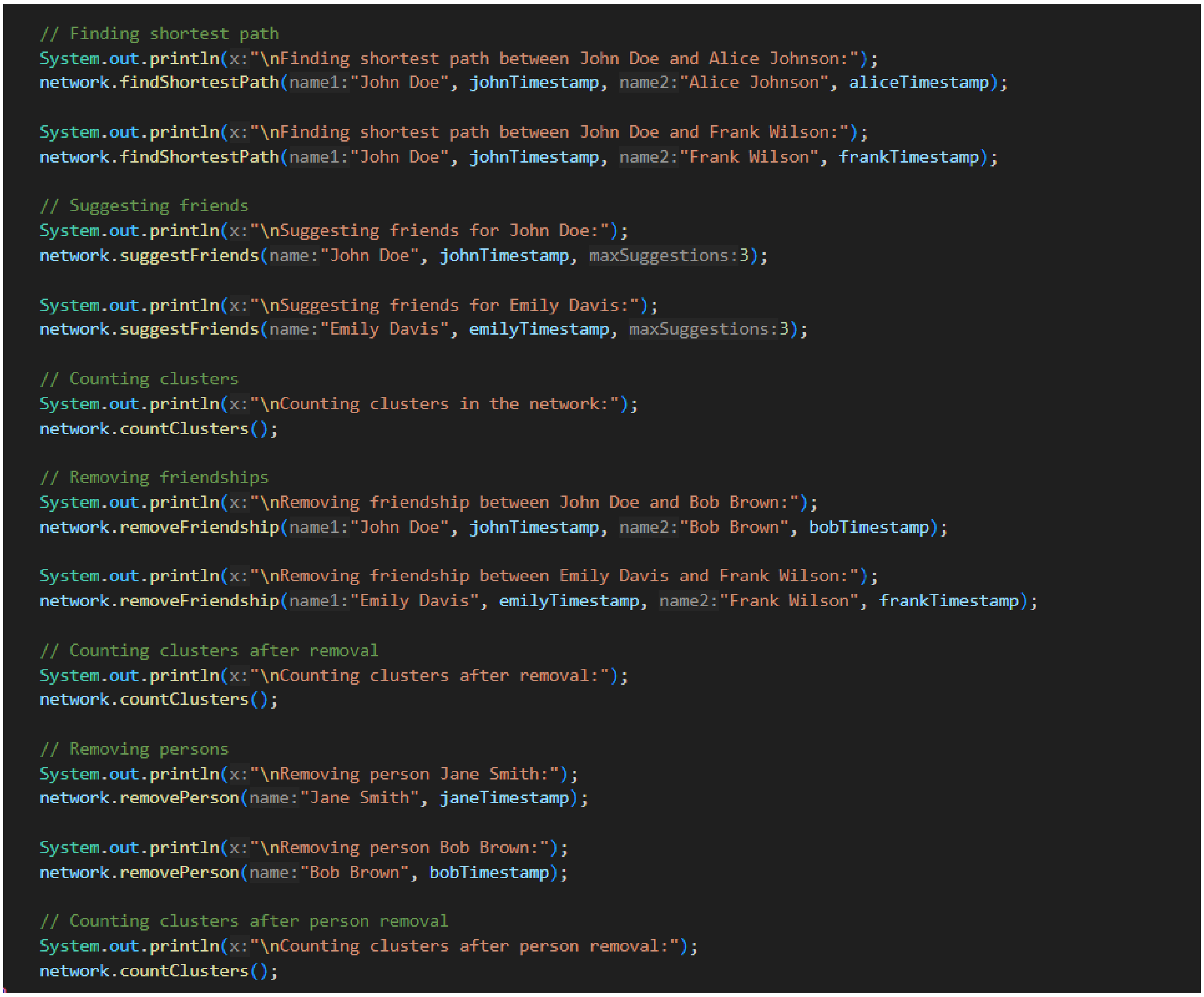
```java
  System.out.println("Number of clusters found: " + clusterCount);
```

Part 4: Example Outputs

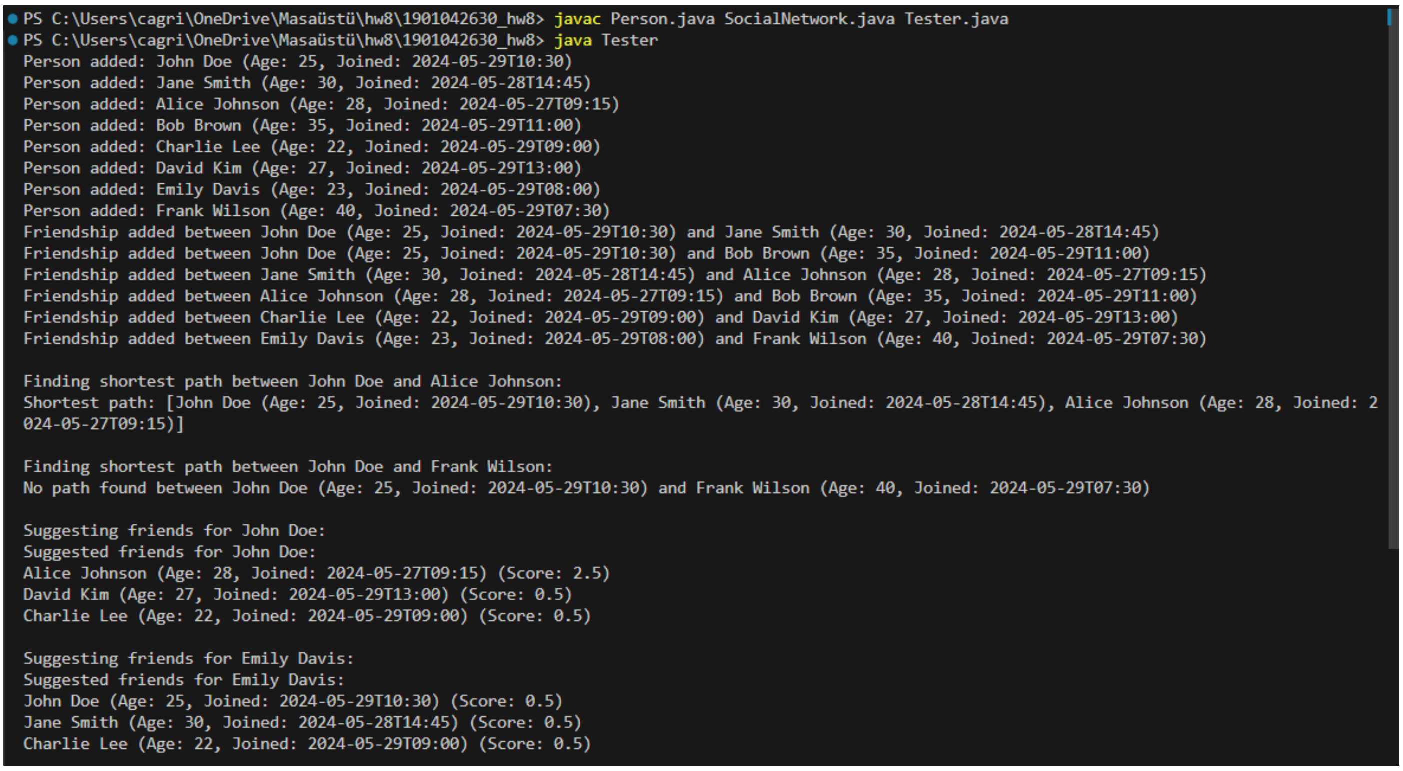
1. **Tester input & output:**



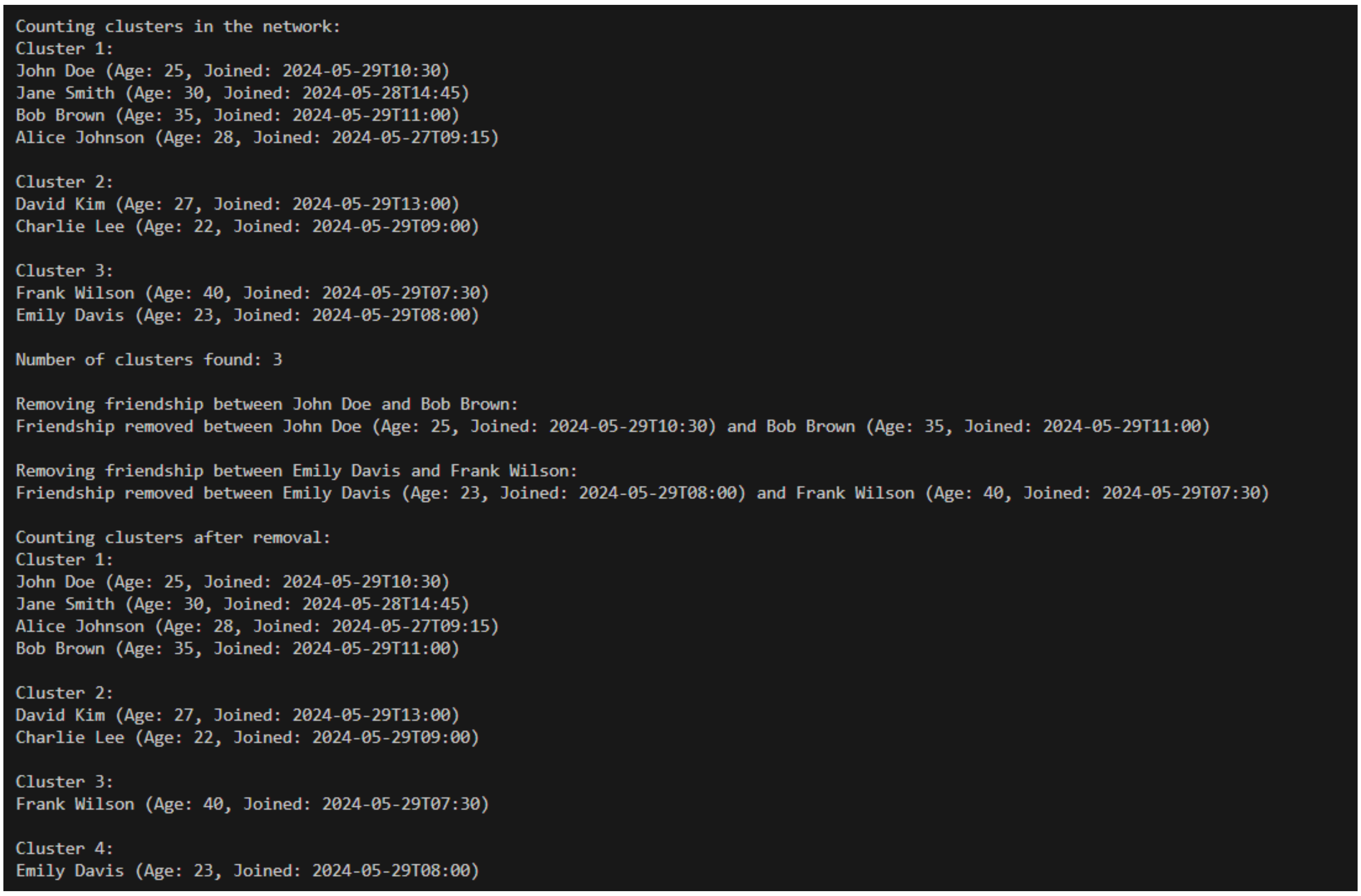
```java
public class Tester {
    public static void main(String[] args) {
        SocialNetwork network = new SocialNetwork();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

        // Adding persons with fixed timestamps for consistency
        LocalDateTime johnTimestamp = LocalDateTime.parse("2024-05-29 10:30:00", formatter);
        network.addPerson("John Doe", 25, Arrays.asList("reading", "hiking", "cooking"), johnTimestamp);

        LocalDateTime janeTimestamp = LocalDateTime.parse("2024-05-28 14:45:00", formatter);
        network.addPerson("Jane Smith", 30, Arrays.asList("running", "reading"), janeTimestamp);

        LocalDateTime aliceTimestamp = LocalDateTime.parse("2024-05-27 09:15:00", formatter);
        network.addPerson("Alice Johnson", 28, Arrays.asList("painting", "cooking"), aliceTimestamp);

        LocalDateTime bobTimestamp = LocalDateTime.parse("2024-05-29 11:00:00", formatter);
        network.addPerson("Bob Brown", 35, Arrays.asList("hiking", "painting"), bobTimestamp);

        LocalDateTime charlieTimestamp = LocalDateTime.parse("2024-05-29 09:00:00", formatter);
        network.addPerson("Charlie Lee", 22, Arrays.asList("reading", "running", "gaming"), charlieTimestamp);

        LocalDateTime davidTimestamp = LocalDateTime.parse("2024-05-29 13:00:00", formatter);
        network.addPerson("David Kim", 27, Arrays.asList("gaming", "cooking"), davidTimestamp);

        LocalDateTime emilyTimestamp = LocalDateTime.parse("2024-05-29 08:00:00", formatter);
        network.addPerson("Emily Davis", 23, Arrays.asList("hiking", "running"), emilyTimestamp);

        LocalDateTime frankTimestamp = LocalDateTime.parse("2024-05-29 07:30:00", formatter);
        network.addPerson("Frank Wilson", 40, Arrays.asList("painting", "gaming"), frankTimestamp);

        // Adding friendships
        network.addFriendship("John Doe", johnTimestamp, "Jane Smith", janeTimestamp);
        network.addFriendship("John Doe", johnTimestamp, "Bob Brown", bobTimestamp);
        network.addFriendship("Jane Smith", janeTimestamp, "Alice Johnson", aliceTimestamp);
        network.addFriendship("Alice Johnson", aliceTimestamp, "Bob Brown", bobTimestamp);
        network.addFriendship("Charlie Lee", charlieTimestamp, "David Kim", davidTimestamp);
        network.addFriendship("Emily Davis", emilyTimestamp, "Frank Wilson", frankTimestamp);
```

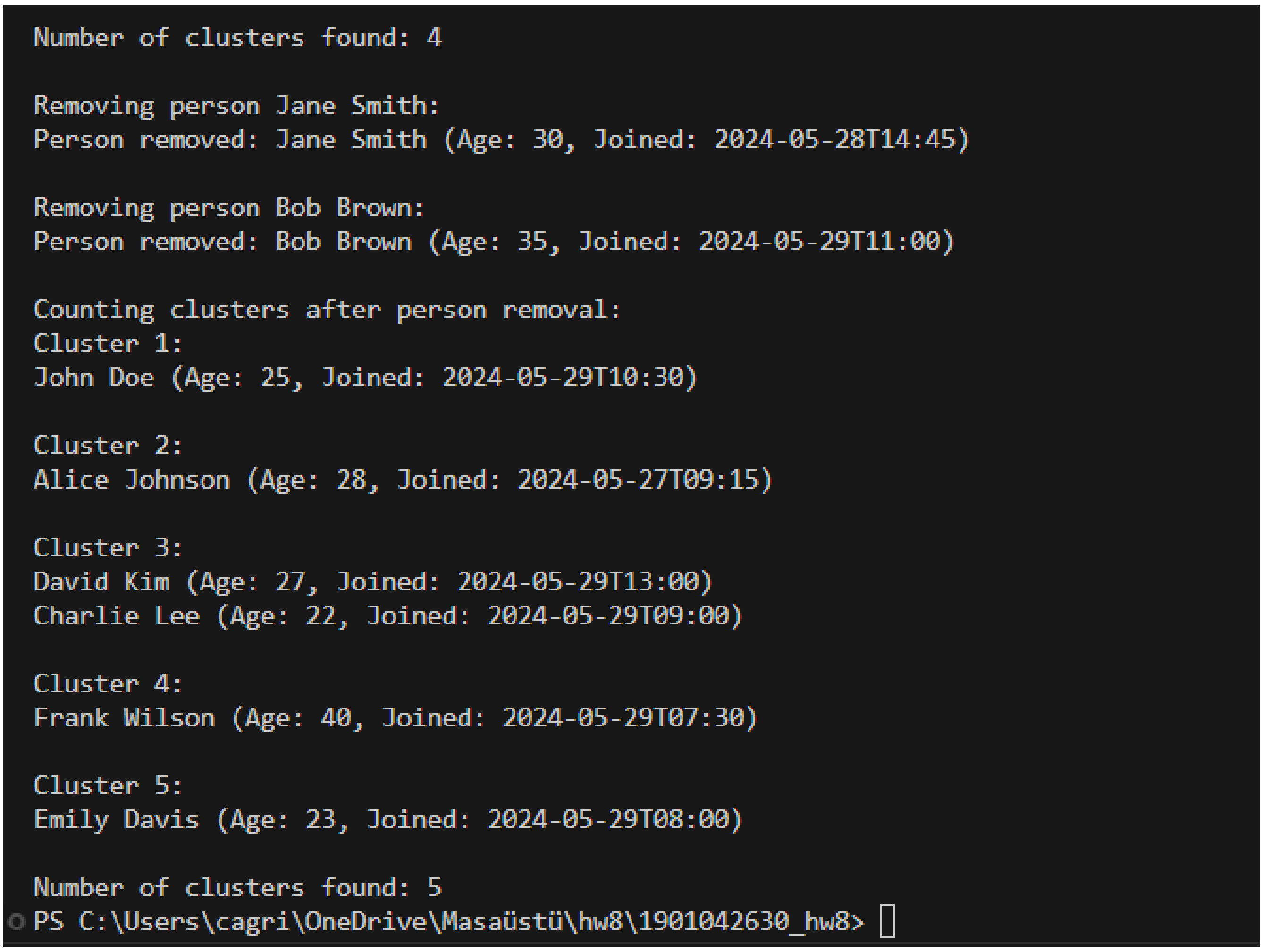


```java
    // Finding shortest path
    System.out.println(x:"\nFinding shortest path between John Doe and Alice Johnson:");
    network.findShortestPath(name1:"John Doe", johnTimestamp, name2:"Alice Johnson", aliceTimestamp);

    System.out.println(x:"\nFinding shortest path between John Doe and Frank Wilson:");
    network.findShortestPath(name1:"John Doe", johnTimestamp, name2:"Frank Wilson", frankTimestamp);

    // Suggesting friends
    System.out.println(x:"\nSuggesting friends for John Doe:");
    network.suggestFriends(name:"John Doe", johnTimestamp, maxSuggestions:3);

    System.out.println(x:"\nSuggesting friends for Emily Davis:");
    network.suggestFriends(name:"Emily Davis", emilyTimestamp, maxSuggestions:3);

    // Counting clusters
    System.out.println(x:"\nCounting clusters in the network:");
    network.countClusters();

    // Removing friendships
    System.out.println(x:"\nRemoving friendship between John Doe and Bob Brown:");
    network.removeFriendship(name1:"John Doe", johnTimestamp, name2:"Bob Brown", bobTimestamp);

    System.out.println(x:"\nRemoving friendship between Emily Davis and Frank Wilson:");
    network.removeFriendship(name1:"Emily Davis", emilyTimestamp, name2:"Frank Wilson", frankTimestamp);

    // Counting clusters after removal
    System.out.println(x:"\nCounting clusters after removal:");
    network.countClusters();

    // Removing persons
    System.out.println(x:"\nRemoving person Jane Smith:");
    network.removePerson(name:"Jane Smith", janeTimestamp);

    System.out.println(x:"\nRemoving person Bob Brown:");
    network.removePerson(name:"Bob Brown", bobTimestamp);

    // Counting clusters after person removal
    System.out.println(x:"\nCounting clusters after person removal:");
    network.countClusters();
```



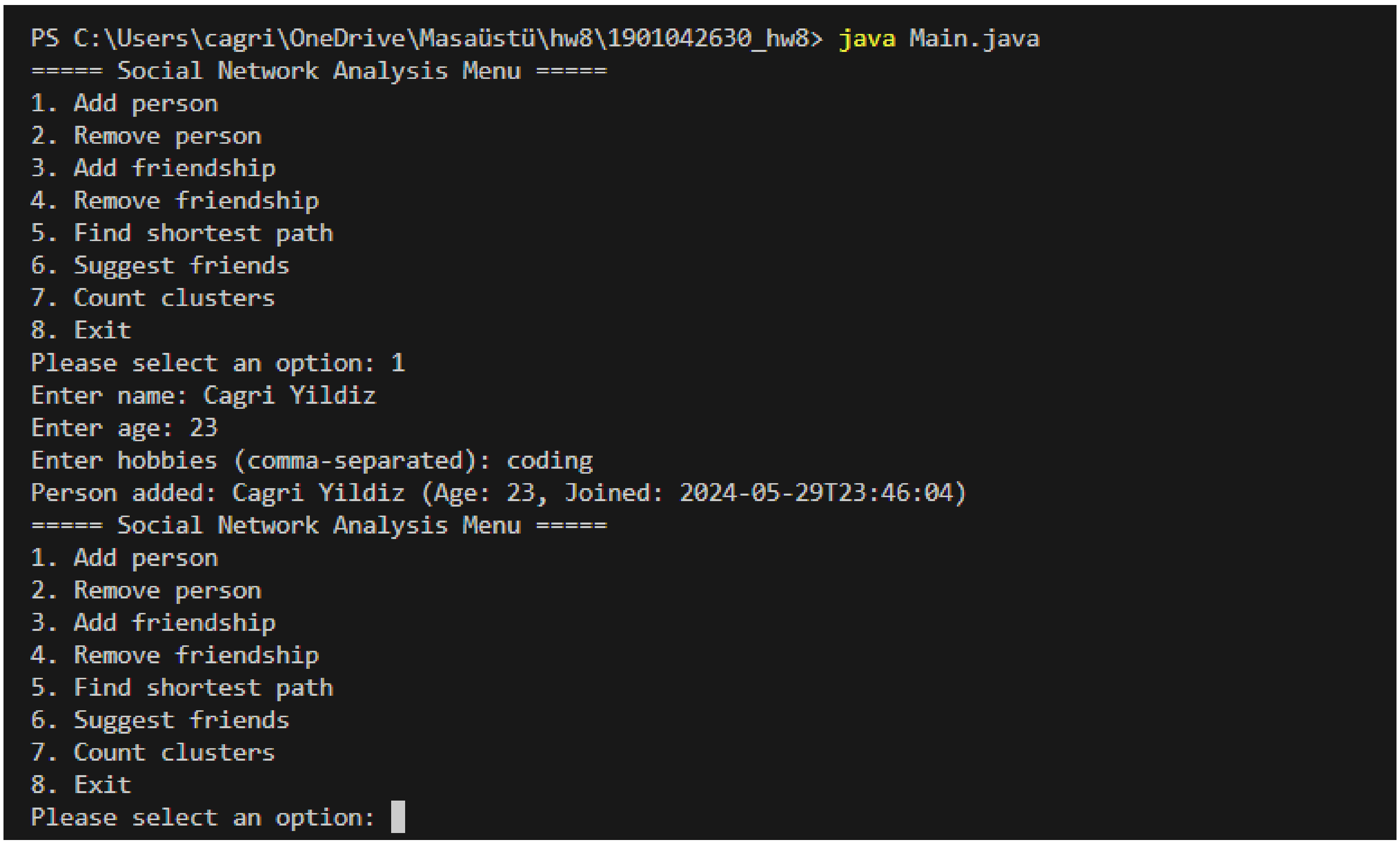
```
PS C:\Users\cagri\OneDrive\Masaüstü\hw8\1901042630_hw8> javac Person.java SocialNetwork.java Tester.java
PS C:\Users\cagri\OneDrive\Masaüstü\hw8\1901042630_hw8> java Tester
Person added: John Doe (Age: 25, Joined: 2024-05-29T10:30)
Person added: Jane Smith (Age: 30, Joined: 2024-05-28T14:45)
Person added: Alice Johnson (Age: 28, Joined: 2024-05-27T09:15)
Person added: Bob Brown (Age: 35, Joined: 2024-05-29T11:00)
Person added: Charlie Lee (Age: 22, Joined: 2024-05-29T09:00)
Person added: David Kim (Age: 27, Joined: 2024-05-29T13:00)
Person added: Emily Davis (Age: 23, Joined: 2024-05-29T08:00)
Person added: Frank Wilson (Age: 40, Joined: 2024-05-29T07:30)
Friendship added between John Doe (Age: 25, Joined: 2024-05-29T10:30) and Jane Smith (Age: 30, Joined: 2024-05-28T14:45)
Friendship added between John Doe (Age: 25, Joined: 2024-05-29T10:30) and Bob Brown (Age: 35, Joined: 2024-05-29T11:00)
Friendship added between Jane Smith (Age: 30, Joined: 2024-05-28T14:45) and Alice Johnson (Age: 28, Joined: 2024-05-27T09:15)
Friendship added between Alice Johnson (Age: 28, Joined: 2024-05-27T09:15) and Bob Brown (Age: 35, Joined: 2024-05-29T11:00)
Friendship added between Charlie Lee (Age: 22, Joined: 2024-05-29T09:00) and David Kim (Age: 27, Joined: 2024-05-29T13:00)
Friendship added between Emily Davis (Age: 23, Joined: 2024-05-29T08:00) and Frank Wilson (Age: 40, Joined: 2024-05-29T07:30)

Finding shortest path between John Doe and Alice Johnson:
Shortest path: [John Doe (Age: 25, Joined: 2024-05-29T10:30), Jane Smith (Age: 30, Joined: 2024-05-28T14:45), Alice Johnson (Age: 28, Joined: 2
024-05-27T09:15)]

Finding shortest path between John Doe and Frank Wilson:
No path found between John Doe (Age: 25, Joined: 2024-05-29T10:30) and Frank Wilson (Age: 40, Joined: 2024-05-29T07:30)

Suggesting friends for John Doe:
Suggested friends for John Doe:
Alice Johnson (Age: 28, Joined: 2024-05-27T09:15) (Score: 2.5)
David Kim (Age: 27, Joined: 2024-05-29T13:00) (Score: 0.5)
Charlie Lee (Age: 22, Joined: 2024-05-29T09:00) (Score: 0.5)

Suggesting friends for Emily Davis:
Suggested friends for Emily Davis:
John Doe (Age: 25, Joined: 2024-05-29T10:30) (Score: 0.5)
Jane Smith (Age: 30, Joined: 2024-05-28T14:45) (Score: 0.5)
Charlie Lee (Age: 22, Joined: 2024-05-29T09:00) (Score: 0.5)
```


```
Counting clusters in the network:
Cluster 1:
John Doe (Age: 25, Joined: 2024-05-29T10:30)
Jane Smith (Age: 30, Joined: 2024-05-28T14:45)
Bob Brown (Age: 35, Joined: 2024-05-29T11:00)
Alice Johnson (Age: 28, Joined: 2024-05-27T09:15)

Cluster 2:
David Kim (Age: 27, Joined: 2024-05-29T13:00)
Charlie Lee (Age: 22, Joined: 2024-05-29T09:00)

Cluster 3:
Frank Wilson (Age: 40, Joined: 2024-05-29T07:30)
Emily Davis (Age: 23, Joined: 2024-05-29T08:00)

Number of clusters found: 3

Removing friendship between John Doe and Bob Brown:
Friendship removed between John Doe (Age: 25, Joined: 2024-05-29T10:30) and Bob Brown (Age: 35, Joined: 2024-05-29T11:00)

Removing friendship between Emily Davis and Frank Wilson:
Friendship removed between Emily Davis (Age: 23, Joined: 2024-05-29T08:00) and Frank Wilson (Age: 40, Joined: 2024-05-29T07:30)

Counting clusters after removal:
Cluster 1:
John Doe (Age: 25, Joined: 2024-05-29T10:30)
Jane Smith (Age: 30, Joined: 2024-05-28T14:45)
Alice Johnson (Age: 28, Joined: 2024-05-27T09:15)
Bob Brown (Age: 35, Joined: 2024-05-29T11:00)

Cluster 2:
David Kim (Age: 27, Joined: 2024-05-29T13:00)
Charlie Lee (Age: 22, Joined: 2024-05-29T09:00)

Cluster 3:
Frank Wilson (Age: 40, Joined: 2024-05-29T07:30)

Cluster 4:
Emily Davis (Age: 23, Joined: 2024-05-29T08:00)
```

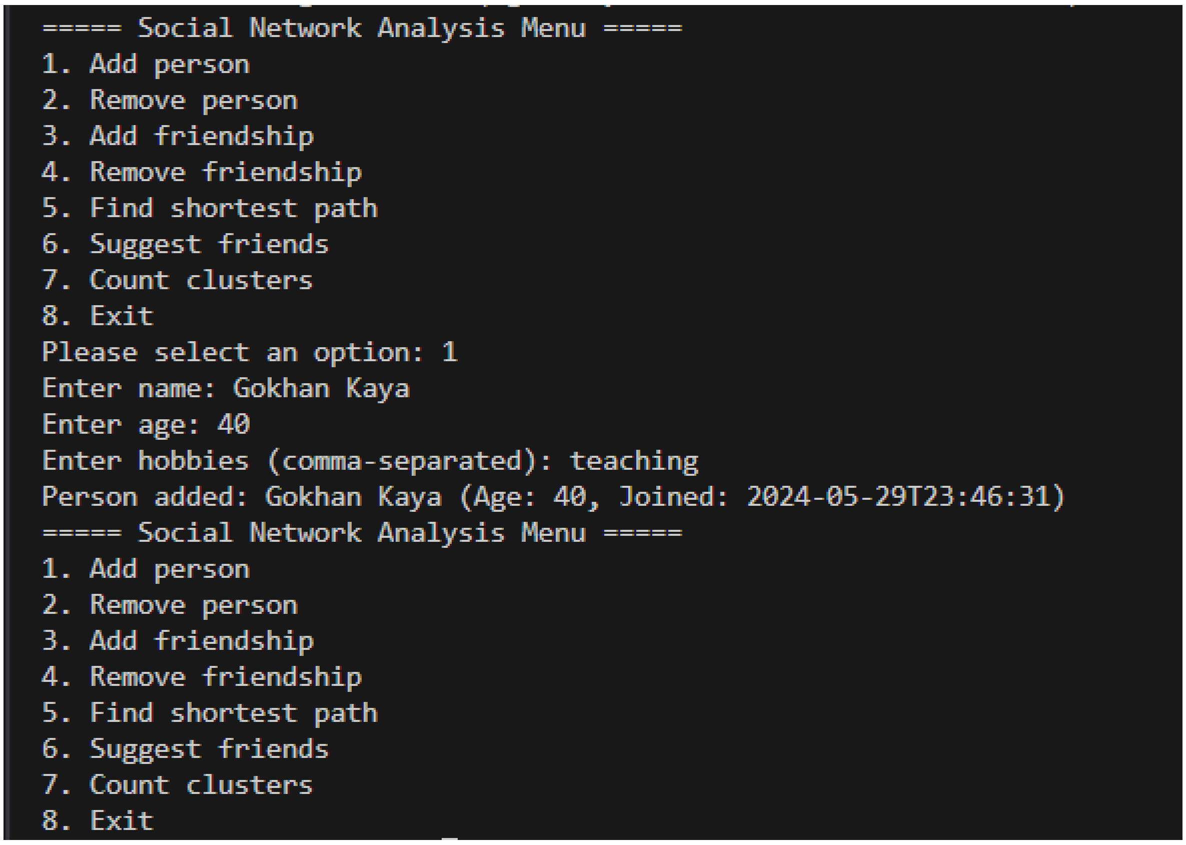

```
Number of clusters found: 4

Removing person Jane Smith:
Person removed: Jane Smith (Age: 30, Joined: 2024-05-28T14:45)

Removing person Bob Brown:
Person removed: Bob Brown (Age: 35, Joined: 2024-05-29T11:00)

Counting clusters after person removal:
Cluster 1:
John Doe (Age: 25, Joined: 2024-05-29T10:30)

Cluster 2:
Alice Johnson (Age: 28, Joined: 2024-05-27T09:15)

Cluster 3:
David Kim (Age: 27, Joined: 2024-05-29T13:00)
Charlie Lee (Age: 22, Joined: 2024-05-29T09:00)

Cluster 4:
Frank Wilson (Age: 40, Joined: 2024-05-29T07:30)

Cluster 5:
Emily Davis (Age: 23, Joined: 2024-05-29T08:00)

Number of clusters found: 5
PS C:\Users\cagri\OneDrive\Masaüstü\hw8\1901042630_hw8>
```

2. User Input & Outputs:

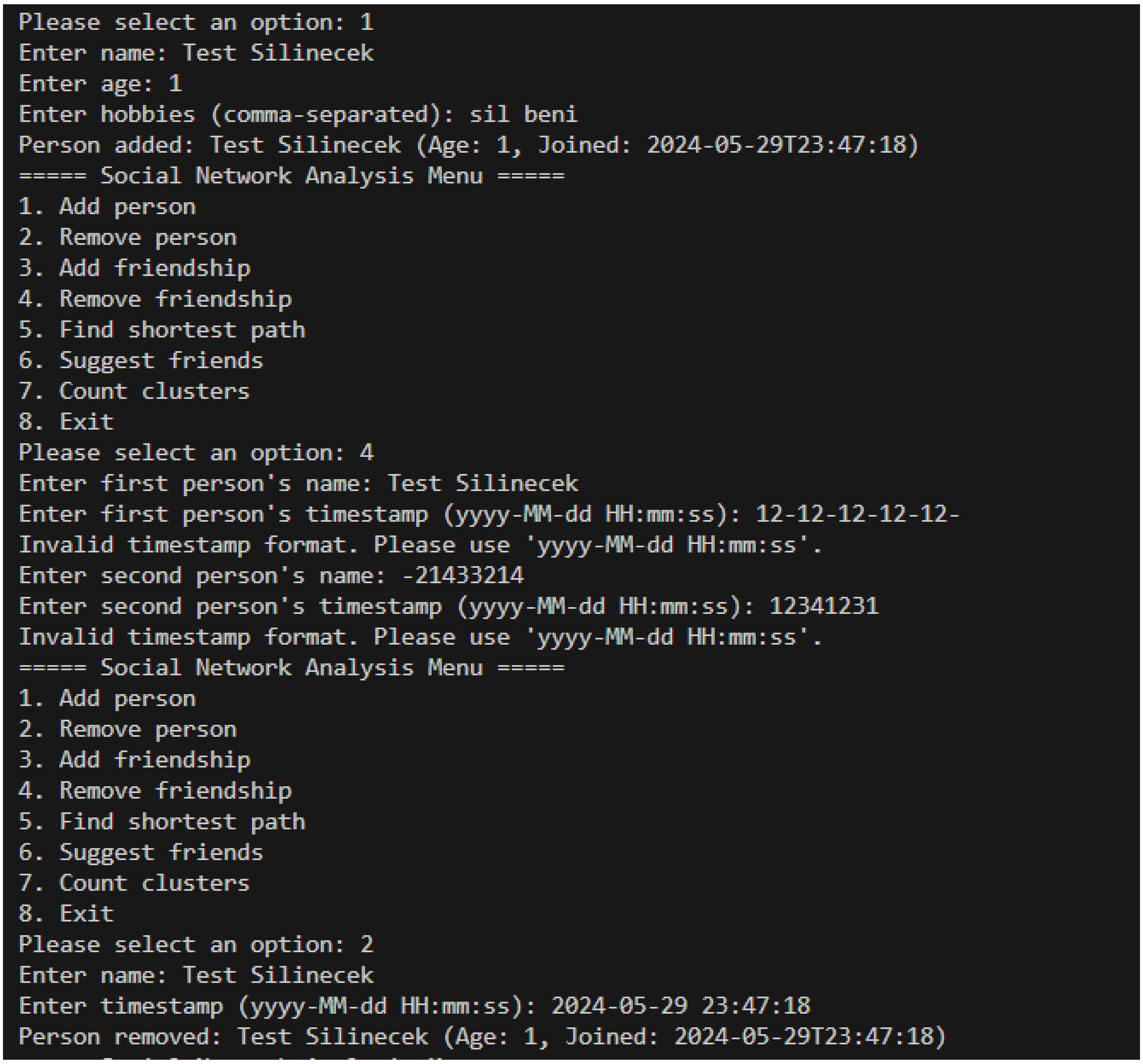
Note: When entering timestamp, you should enter it as 2024-05-29 23:51:41, not 2024-05-29T23:51:41.



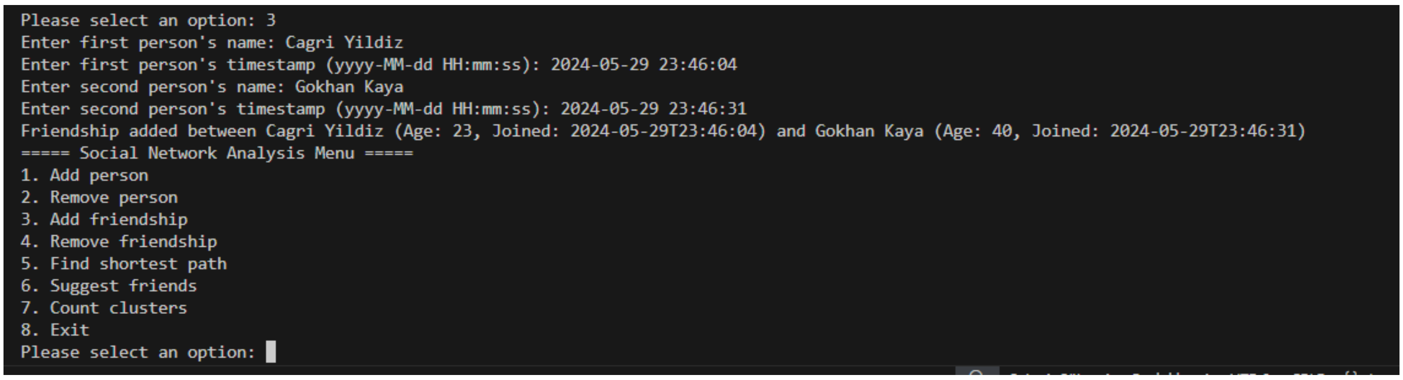
```
PS C:\Users\cagri\OneDrive\Masaüstü\hw8\1901042630_hw8> java Main.java
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 1
Enter name: Cagri Yildiz
Enter age: 23
Enter hobbies (comma-separated): coding
Person added: Cagri Yildiz (Age: 23, Joined: 2024-05-29T23:46:04)
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option:
```



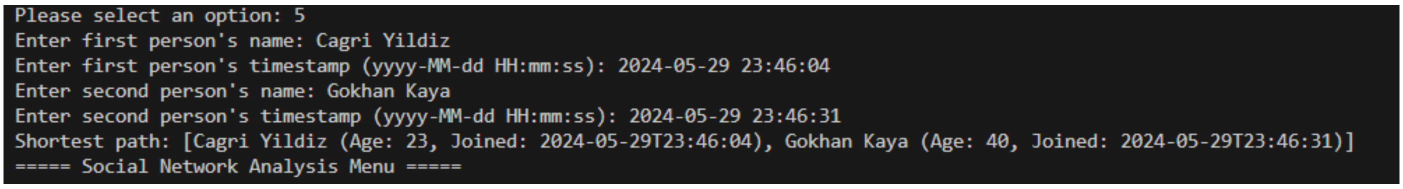
```
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 1
Enter name: Gokhan Kaya
Enter age: 40
Enter hobbies (comma-separated): teaching
Person added: Gokhan Kaya (Age: 40, Joined: 2024-05-29T23:46:31)
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
```



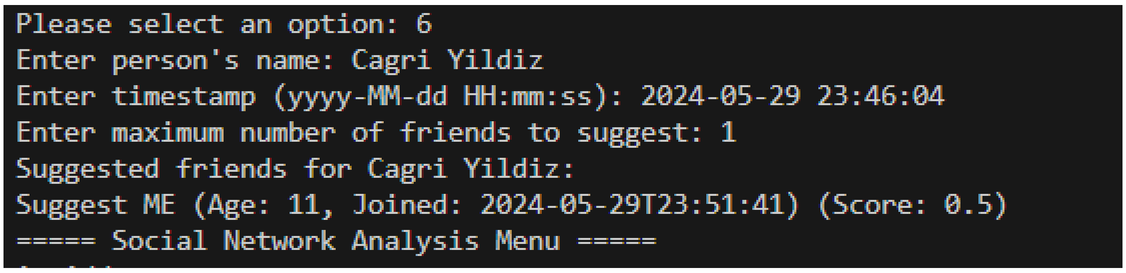
```
Please select an option: 1
Enter name: Test Silinecek
Enter age: 1
Enter hobbies (comma-separated): sil beni
Person added: Test Silinecek (Age: 1, Joined: 2024-05-29T23:47:18)
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 4
Enter first person's name: Test Silinecek
Enter first person's timestamp (yyyy-MM-dd HH:mm:ss): 12-12-12-12-12-
Invalid timestamp format. Please use 'yyyy-MM-dd HH:mm:ss'.
Enter second person's name: -21433214
Enter second person's timestamp (yyyy-MM-dd HH:mm:ss): 12341231
Invalid timestamp format. Please use 'yyyy-MM-dd HH:mm:ss'.
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 2
Enter name: Test Silinecek
Enter timestamp (yyyy-MM-dd HH:mm:ss): 2024-05-29 23:47:18
Person removed: Test Silinecek (Age: 1, Joined: 2024-05-29T23:47:18)
```



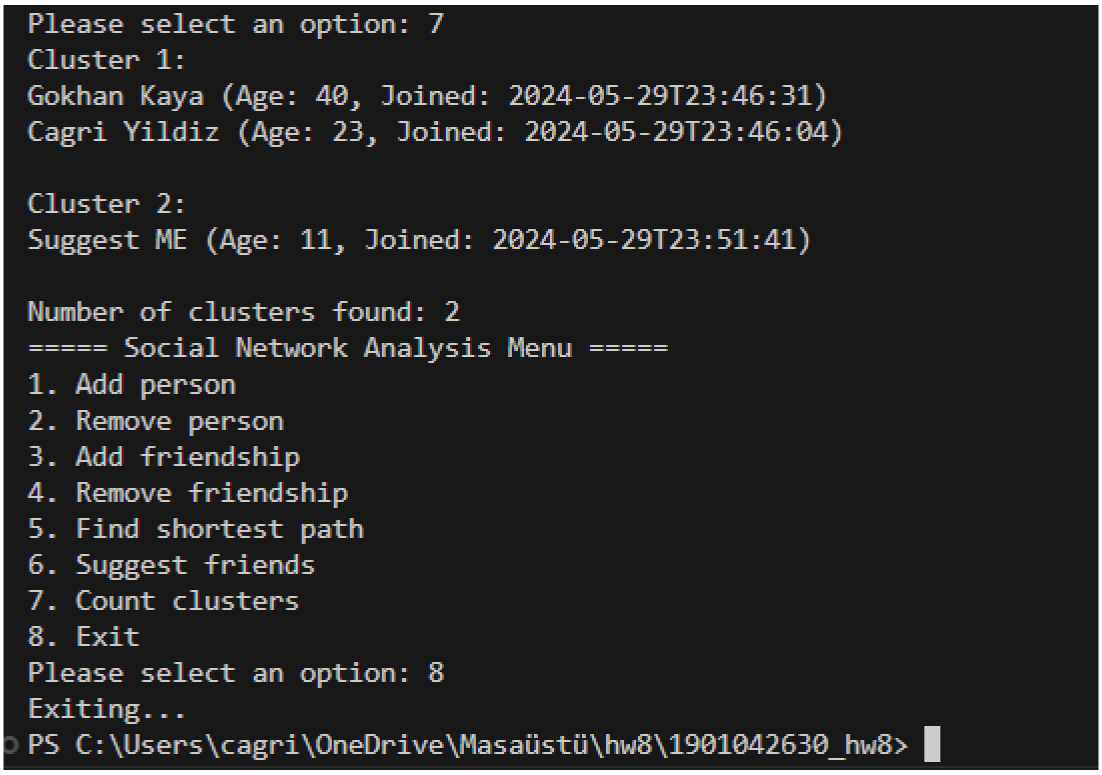
```
Please select an option: 3
Enter first person's name: Cagri Yildiz
Enter first person's timestamp (yyyy-MM-dd HH:mm:ss): 2024-05-29 23:46:04
Enter second person's name: Gokhan Kaya
Enter second person's timestamp (yyyy-MM-dd HH:mm:ss): 2024-05-29 23:46:31
Friendship added between Cagri Yildiz (Age: 23, Joined: 2024-05-29T23:46:04) and Gokhan Kaya (Age: 40, Joined: 2024-05-29T23:46:31)
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option:
```



```
Please select an option: 5
Enter first person's name: Cagri Yildiz
Enter first person's timestamp (yyyy-MM-dd HH:mm:ss): 2024-05-29 23:46:04
Enter second person's name: Gokhan Kaya
Enter second person's timestamp (yyyy-MM-dd HH:mm:ss): 2024-05-29 23:46:31
Shortest path: [Cagri Yildiz (Age: 23, Joined: 2024-05-29T23:46:04), Gokhan Kaya (Age: 40, Joined: 2024-05-29T23:46:31)]
===== Social Network Analysis Menu =====
```



```
Please select an option: 6
Enter person's name: Cagri Yildiz
Enter timestamp (yyyy-MM-dd HH:mm:ss): 2024-05-29 23:46:04
Enter maximum number of friends to suggest: 1
Suggested friends for Cagri Yildiz:
Suggest ME (Age: 11, Joined: 2024-05-29T23:51:41) (Score: 0.5)
===== Social Network Analysis Menu =====
```



```
Please select an option: 7
Cluster 1:
Gokhan Kaya (Age: 40, Joined: 2024-05-29T23:46:31)
Cagri Yildiz (Age: 23, Joined: 2024-05-29T23:46:04)

Cluster 2:
Suggest ME (Age: 11, Joined: 2024-05-29T23:51:41)

Number of clusters found: 2
===== Social Network Analysis Menu =====
1. Add person
2. Remove person
3. Add friendship
4. Remove friendship
5. Find shortest path
6. Suggest friends
7. Count clusters
8. Exit
Please select an option: 8
Exiting...
PS C:\Users\cagri\OneDrive\Masaüstü\hw8\1901042630_hw8>
```

THANKS FOR READING

ÇAĞRI YILDIZ

1901042630