CSE 222

Data Structures and Algorithms

8th Assignment

Report

Author

Mert Emir ŞEKER

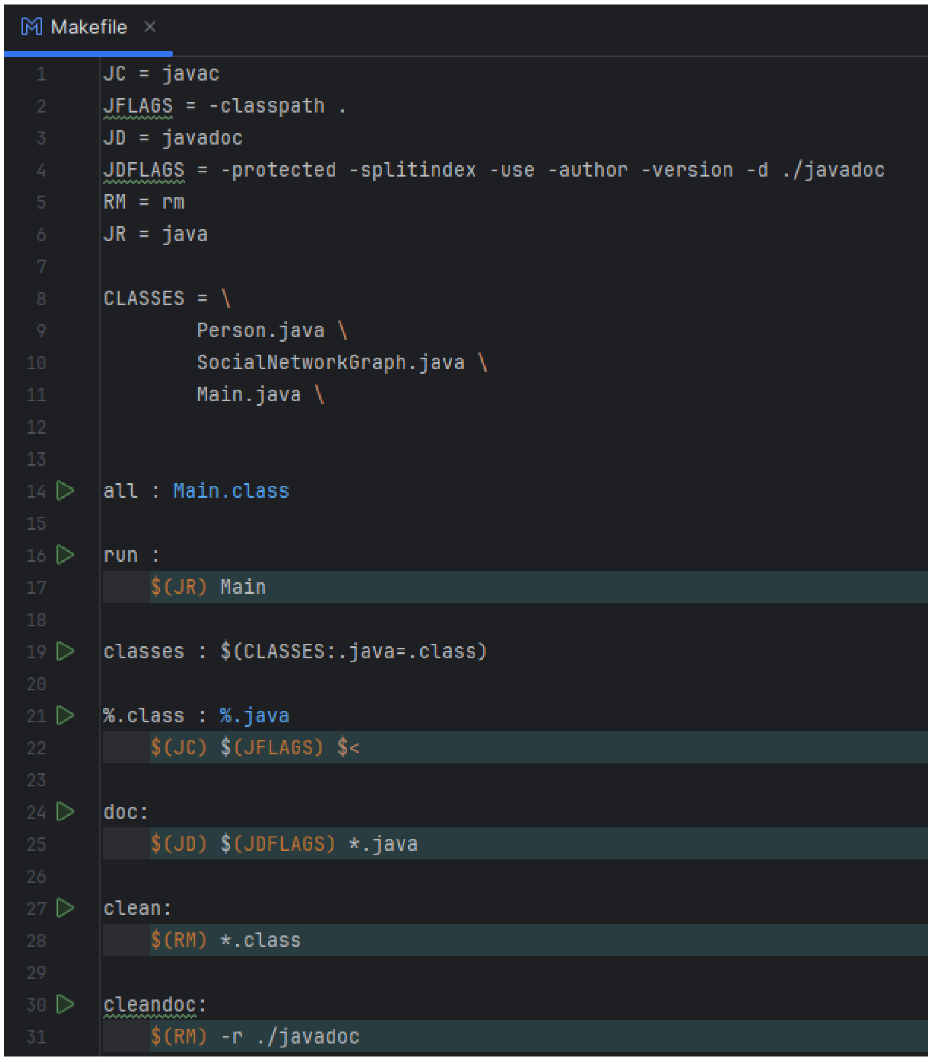
200104004085

Date

28.05.2024

Table Of Contents

1. Makefile:


Makefile

```
JC = javac
JFLAGS = -classpath .
JD = javadoc
JDFLAGS = -protected -splitindex -use -author -version -d ./javadoc
RM = rm
JR = java

CLASSES = \
        Person.java \
        SocialNetworkGraph.java \
        Main.java \


all : Main.class

run :
	$(JR) Main

classes : $(CLASSES:.java=.class)

%.class : %.java
	$(JC) $(JFLAGS) $<

doc:
	$(JD) $(JDFLAGS) *.java

clean:
	$(RM) *.class

cleandoc:
	$(RM) -r ./javadoc
```

2. Makefile Commands:

all: Compiles the Main.java file.

run: Runs the main program.

classes: Compiles the entire project.

%.class : %.java: Compiles individual Java files.

doc: Generates Javadoc documentation.

clean: Deletes compiled class files.

cleandoc: Deletes Javadoc documentation

3. How to run the code?

To run the program, you can write **make** or **make all** in terminal and you can write **make run**. If you want to create Javadoc you can write **make doc**.

4. Code Explanation:

4.1. Person.java:

The Person class acts as a template for users in a social network by encapsulating important details like name, age, interests, and the timestamp that indicates when the instance was created. This class has a constructor that sets the timestamp to the current date and time and initializes the name, age, and interests by copying the supplied list to guarantee encapsulation. For every attribute, it has thorough getter and setter methods that enable restricted access and modification capabilities. To make it easier to display and debug, the class extends the toString function to provide a comprehensive string representation of the user's name, age, and interests. In order to compare equality between two Person objects with the same name, the equals method is overridden. This ensures that two Person objects with the same name are regarded as equal, which is essential for accurately managing persons inside collections. In order to ensure appropriate functioning in hash-based collections and compatibility with the equals method, the hashCode method is modified to construct a hash code generated from the name. The class facilitates the depiction of each person's varied interests by maintaining hobbies as a list. Therefore, the Person class is an essential component of the social network's data structure as it is designed to guarantee that people can be uniquely identified by their names and offers reliable methods for modifying and retrieving their properties.

4.2. SocialNetworkGraph.java:

Using a graph data structure, the SocialNetworkGraph class simulates a social network in which people (Person objects) are represented as nodes and friendships as edges connecting these nodes. To facilitate rapid search and manipulation of friendships, the class maintains two maps: one indexed by the names of the participants, and another that associates each participant with a list of their friends. It offers ways to join and leave the network, form and break friendships, and examine the structure of the network through a number of functions like breadth-first search (BFS) to find the shortest path between two people, counting the number of clusters (connected components), and friend-suggesting based on mutual friends and common interests. Thus, this class provides all of the essential functions required to create, administer, and evaluate a social network, utilizing graph algorithms and data structures to facilitate these tasks.

4.2.1. public void addPerson(String name, int age, List<String> hobbies) :

Adding a new member to the social network is the responsibility of the addPerson function. Name, age, and interests are the three factors it requires. The process begins by determining if a person with the specified name already exists in the persons map. In that case, it ends the procedure and publishes a message saying that the individual already exists. In the event that the name cannot be located, the name serves as the key to add the new Person object to the people map together with the supplied information. It also initializes the friendships map with an empty list of friends for this new individual. Lastly, it publishes a confirmation message with the individual's information and the timestamp associated with their network addition. By using this method, every individual in the social network may be individually identified by their name.

4.2.2. public void removePerson(String name) : The removePerson method's purpose is to eliminate an individual by name from the social network. It initially uses the getPersonByName method to acquire the Person object linked to the supplied name. The method just returns without taking any further action if the person does not exist (i.e., the name cannot be located). The individual gets deleted from both the friendships and persons maps if they are located. Furthermore, it eliminates the buddy from each list it iteratively goes through the friendships map, guaranteeing that all references to this individual are eliminated. Ultimately, a confirmation message stating that the user has been deleted is printed. By using this method, the individual and all of their connections are guaranteed to be fully erased from the social network.

4.2.3. public void addFriendship(String name1, String name2): The addFriendship method establishes a friendship between two people in the social network. It takes two parameters, name1 and name2, which are the names of the two individuals to be connected. The method first retrieves the Person objects corresponding to these names using the getPersonByName method. If either person is not found, the method returns without making any changes. If both individuals are found, it adds each person to the other's list of friends in the friendships map, effectively creating a bidirectional friendship. Finally, it prints a message confirming that the friendship has been established between the two individuals. This method ensures that friendships are consistently and correctly represented in the social network.

4.2.4. public void removeFriendship(String name1, String name2): The removeFriendship method is used to dissolve a friendship between two people in the social network. It takes two parameters, name1 and name2, which represent the names of the two individuals whose friendship is to be removed. The method first retrieves the Person objects corresponding to these names using the getPersonByName method. If either person is not found, the method returns without performing any action. If both individuals are found, it removes each person from the other's list of friends in the friendships map, thereby removing the bidirectional friendship. Finally, it prints a message confirming that the friendship between the two individuals has been removed. This method ensures that the social network's data structure accurately reflects the removal of the friendship.

4.2.5. public void findShortestPath(String startName, String endName): The findShortestPath method aims to find and print the shortest path between two people in the social network using the Breadth-First Search (BFS) algorithm. It takes two parameters, startName and endName, which are the names of the starting and ending individuals. The method retrieves the corresponding Person objects using getPersonByName. If either person is not found, the method returns without any action. It initializes a queue for BFS, a map to keep track of previous nodes in the path (prev), and a set to keep track of visited nodes. The BFS starts with the start person, marking them as visited and adding them to the queue. As it processes each person, if the current person equals the end person, the method calls printPath to print the path from start to end and then returns. If the current person is not the end person, it continues to explore the unvisited neighbors, adding them to the queue, marking them as visited, and recording their previous node. If the queue is exhausted without finding the end person, it prints a message indicating that no path was found between the two individuals. This method effectively finds the shortest path between two people in the network, if it exists.

4.2.6. private void printPath(Person start, Person end, Map<Person, Person> prev): The printPath method is a helper function used to reconstruct and print the shortest path between two people in the social network. It takes three parameters: the start person, the end person, and a map prev that contains the previous node for each node in the path, as determined by the BFS algorithm. The method initializes an empty list path to store the sequence of people in the path. It starts from the end person and traces back through the prev map until it reaches the start person, adding each person to the path list. After constructing the path in reverse order, it reverses the list to get the correct order from start to end. It then constructs a string representation of the path, with each person's name separated by " -> ". Finally, it prints the constructed path string. This method effectively visualizes the shortest path found between the two individuals.

4.2.7. public void countClusters(): The social network's clusters, or related components, are found and counted using the countClusters method. An integer clusterCount is initialized to tally the number of clusters detected, and an empty set visited is initialized to maintain track of the individuals who have already been visited throughout the search. Every member of the network is iterated over by the technique. It uses the helper method bfs to execute a breadth-first search (BFS) beginning from each individual if they have not been visited. All individuals in the same cluster are added to a list cluster and marked as visited by this BFS. Following the completion of a cluster's BFS, it outputs the names of all members of the current cluster and increases the clusterCount. The total number of clusters detected is printed once every person has been processed. This technique efficiently locates and shows every linked element in the social network, offering a glimpse into the organization of the network.

4.2.8. private void bfs(Person start, Set<Person> visited, List<Person> cluster): The bfs method performs a breadth-first search (BFS) to explore and identify all people within the same cluster as the starting person. It takes three parameters: the start person from which the BFS begins, a visited set to keep track of the people that have been visited, and a cluster list to store the people belonging to the current cluster. The method initializes a queue and adds the start person to it, marking them as visited. It then enters a loop that continues until the queue is empty. In each iteration, the method dequeues the current person, adds them to the cluster list, and explores their neighbors (friends) from the friendships map. For each neighbor that has not yet been visited, it adds the neighbor to the queue and marks them as visited. This process ensures that all connected people (those reachable from the start person) are visited and added to the cluster list. The method effectively identifies all members of the same cluster, supporting the countClusters method in determining the connected components of the social network.

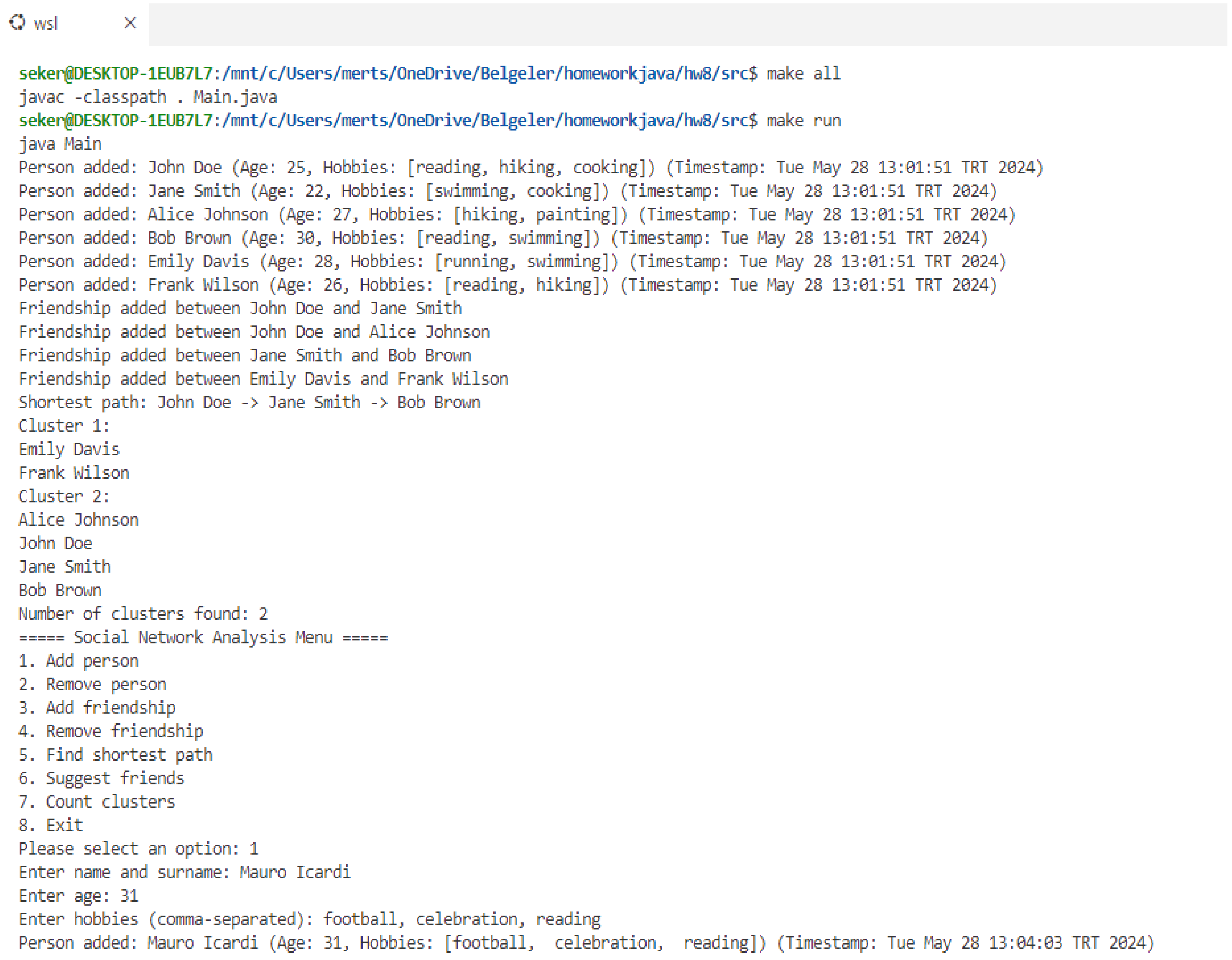
4.2.9. public void suggestFriends(String name, int maxSuggestions): The suggestFriends method generates friend suggestions for a given person based on the number of mutual friends and common hobbies. It takes two parameters: name, which is the name of the person for whom suggestions are being generated, and maxSuggestions, which is the maximum number of friend suggestions to return. The method starts by retrieving the Person object for the given name using getPersonByName. If the person is not found, the method returns without any action. Next, it initializes three maps: scores to store the calculated suggestion scores for each potential friend, mutualFriendsMap to track the number of mutual friends, and commonHobbiesMap to track the number of common hobbies. It also retrieves the list of the person's direct friends from the friendships map. The method then iterates over all people in the network. For each person, it skips those who are either the person themselves or already a direct friend. For the remaining people, it calculates the number of mutual friends by checking how many of the person's direct friends are also friends with the potential friend. It also calculates the number of common hobbies by comparing the hobbies of the person and the potential friend. A score is computed for each potential friend based on the formula mutualFriends + 0.5 * commonHobbies, and the results are stored in the scores, mutualFriendsMap, and commonHobbiesMap. Finally, the method sorts the entries in the scores map by the score in descending order, limits the results to maxSuggestions, and prints the suggested friends along with their scores, mutual friends, and common hobbies. This method effectively leverages the network structure and shared interests to provide personalized friend suggestions.

4.2.10. private Person getPersonByName(String name): The getPersonByName method is a private helper function that retrieves a Person object from the people map based on the given name. It takes one parameter, name, which is the name of the person to be retrieved. The method attempts to get the Person object associated with the provided name from the people map.If the person is found, it returns the Person object. If the person is not found (i.e., the name does not exist in the map), it prints a message indicating that the person with the specified name is not found and returns null. This method simplifies the process of looking up people in the network and provides a consistent way to handle cases where the person might not exist.

4.3. Main.java: The Main class demonstrates the functionality of the SocialNetworkGraph by providing a command-line interface for users to interact with the social network. The main method initializes a SocialNetworkGraph object and adds sample people and friendships for demonstration purposes. It showcases features such as adding and removing people, establishing and dissolving friendships, finding the shortest path between two people, counting clusters, and suggesting friends. The main method also includes an interactive loop that presents a menu with options for users to perform various operations on the social network. Users can add or remove people, add or remove friendships, find the shortest path between two individuals, get friend suggestions, and count the number of clusters. Each option prompts the user for necessary input and calls the corresponding method in the SocialNetworkGraph class to perform the action. The loop continues until the user selects the option to exit, at which point the program terminates. This class effectively serves as a user interface for testing and demonstrating the capabilities of the SocialNetworkGraph.

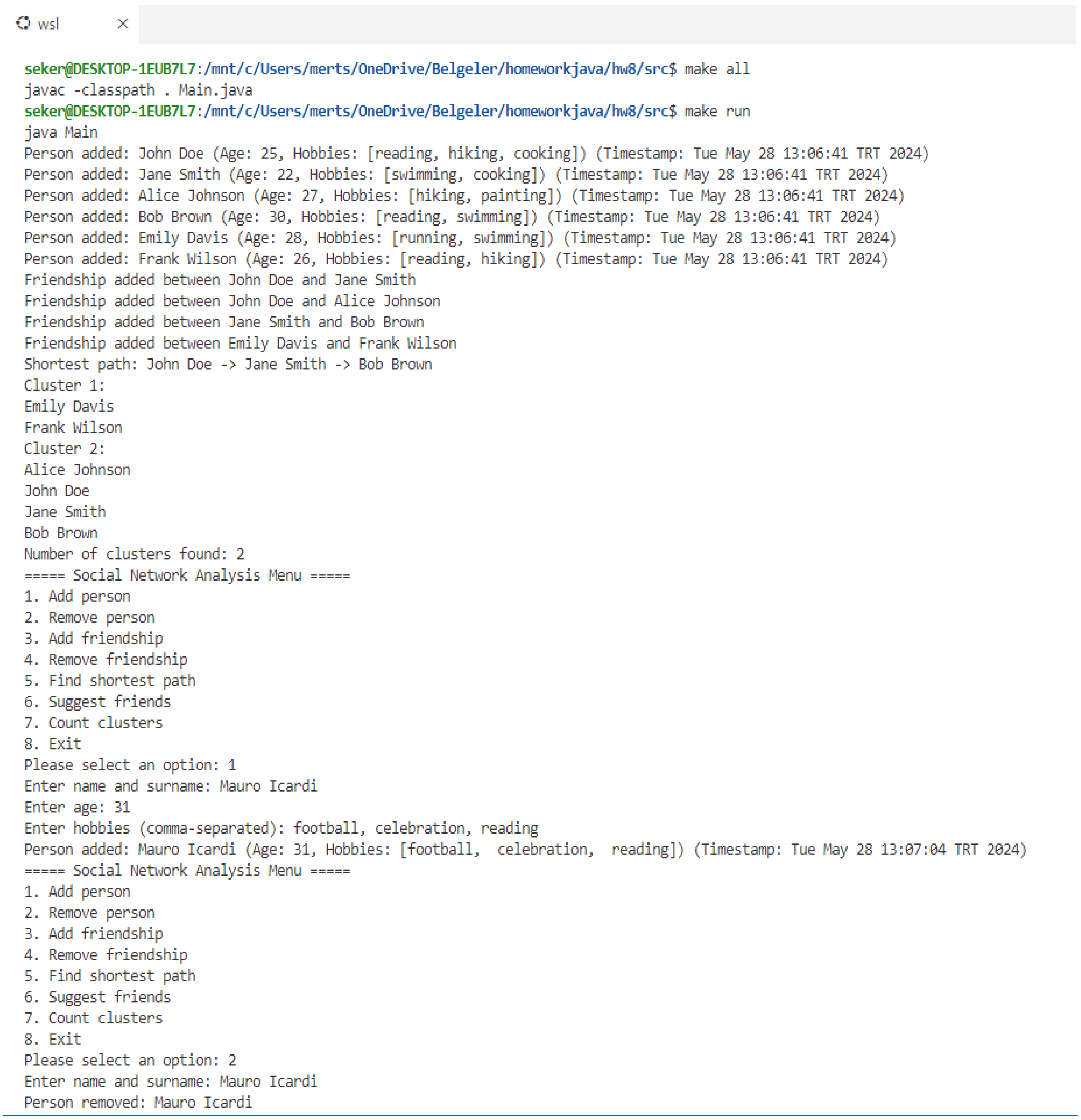
5. Example Outputs:

5.1. Add person:


wsl

```
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make all
javac -classpath . Main.java
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make run
java Main
Person added: John Doe (Age: 25, Hobbies: [reading, hiking, cooking]) (Timestamp: Tue May 28 13:01:51 TRT 2024)
Person added: Jane Smith (Age: 22, Hobbies: [swimming, cooking]) (Timestamp: Tue May 28 13:01:51 TRT 2024)
Person added: Alice Johnson (Age: 27, Hobbies: [hiking, painting]) (Timestamp: Tue May 28 13:01:51 TRT 2024)
Person added: Bob Brown (Age: 30, Hobbies: [reading, swimming]) (Timestamp: Tue May 28 13:01:51 TRT 2024)
Person added: Emily Davis (Age: 28, Hobbies: [running, swimming]) (Timestamp: Tue May 28 13:01:51 TRT 2024)
Person added: Frank Wilson (Age: 26, Hobbies: [reading, hiking]) (Timestamp: Tue May 28 13:01:51 TRT 2024)
Friendship added between John Doe and Jane Smith
Friendship added between John Doe and Alice Johnson
Friendship added between Jane Smith and Bob Brown
Friendship added between Emily Davis and Frank Wilson
Shortest path: John Doe -> Jane Smith -> Bob Brown
Cluster 1:
Emily Davis
Frank Wilson
Cluster 2:
Alice Johnson
John Doe
Jane Smith
Bob Brown
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: 1
Enter name and surname: Mauro Icardi
Enter age: 31
Enter hobbies (comma-separated): football, celebration, reading
Person added: Mauro Icardi (Age: 31, Hobbies: [football,  celebration,  reading]) (Timestamp: Tue May 28 13:04:03 TRT 2024)
```

5.2. Remove person:



wsl

```
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make all
javac -classpath . Main.java
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make run
java Main
Person added: John Doe (Age: 25, Hobbies: [reading, hiking, cooking]) (Timestamp: Tue May 28 13:06:41 TRT 2024)
Person added: Jane Smith (Age: 22, Hobbies: [swimming, cooking]) (Timestamp: Tue May 28 13:06:41 TRT 2024)
Person added: Alice Johnson (Age: 27, Hobbies: [hiking, painting]) (Timestamp: Tue May 28 13:06:41 TRT 2024)
Person added: Bob Brown (Age: 30, Hobbies: [reading, swimming]) (Timestamp: Tue May 28 13:06:41 TRT 2024)
Person added: Emily Davis (Age: 28, Hobbies: [running, swimming]) (Timestamp: Tue May 28 13:06:41 TRT 2024)
Person added: Frank Wilson (Age: 26, Hobbies: [reading, hiking]) (Timestamp: Tue May 28 13:06:41 TRT 2024)
Friendship added between John Doe and Jane Smith
Friendship added between John Doe and Alice Johnson
Friendship added between Jane Smith and Bob Brown
Friendship added between Emily Davis and Frank Wilson
Shortest path: John Doe -> Jane Smith -> Bob Brown
Cluster 1:
Emily Davis
Frank Wilson
Cluster 2:
Alice Johnson
John Doe
Jane Smith
Bob Brown
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: 1
Enter name and surname: Mauro Icardi
Enter age: 31
Enter hobbies (comma-separated): football, celebration, reading
Person added: Mauro Icardi (Age: 31, Hobbies: [football,  celebration,  reading]) (Timestamp: Tue May 28 13:07:04 TRT 2024)
===== 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 and surname: Mauro Icardi
Person removed: Mauro Icardi
```

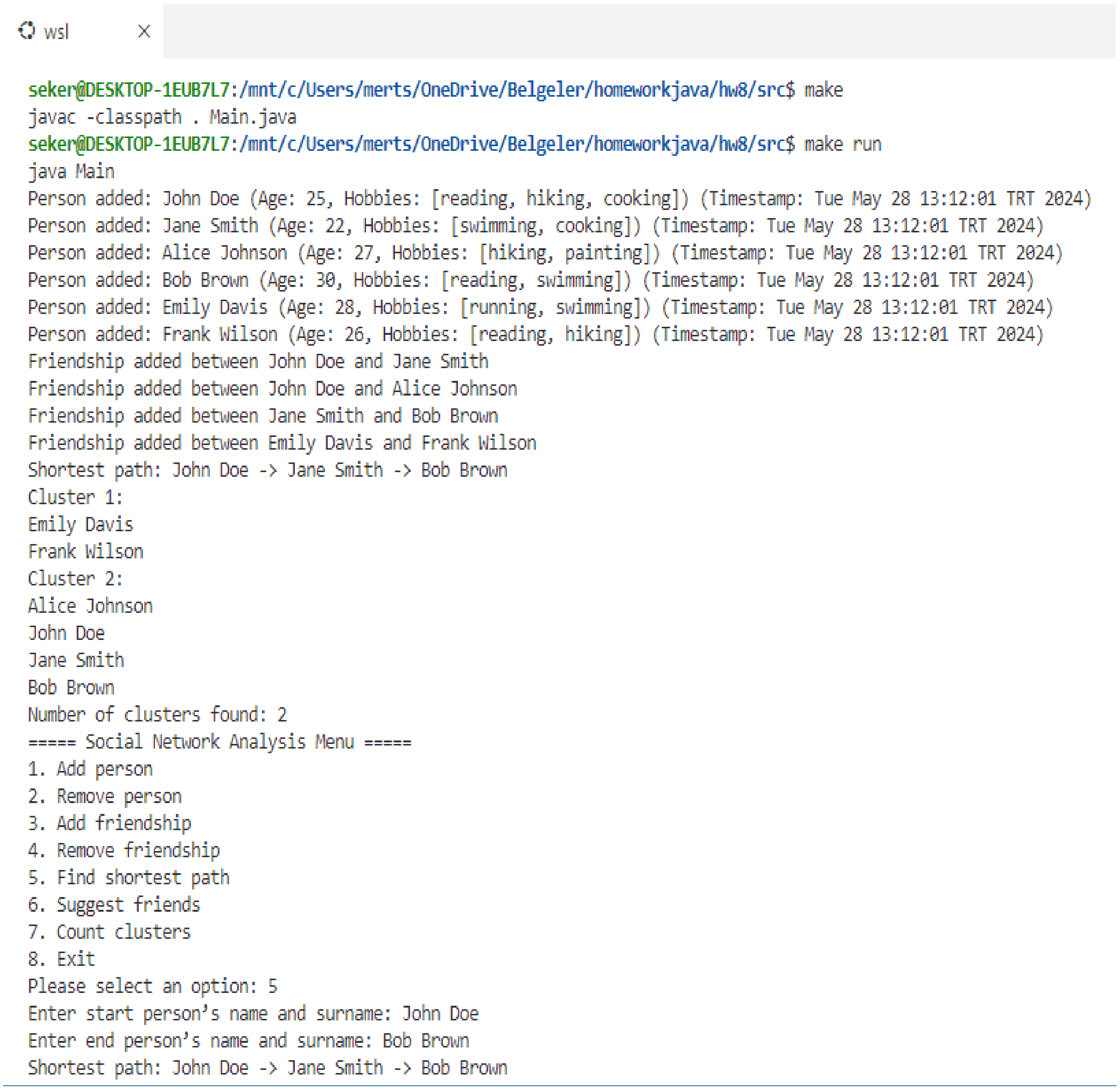
5.3. Add friendship:

```
===== 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 and surname: Mauro Icardi
Enter age: 31
Enter hobbies (comma-separated): football, celebration, reading
Person added: Mauro Icardi (Age: 31, Hobbies: [football,  celebration,  reading]) (Timestamp: Tue May 28 13:08:51 TRT 2024)
===== 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 and surname: Kerem Aktürkoğlu
Enter age: 25
Enter hobbies (comma-separated): football, celebration, writing
Person added: Kerem Aktürkoğlu (Age: 25, Hobbies: [football,  celebration,  writing]) (Timestamp: Tue May 28 13:09:37 TRT 2024)
===== 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: 3
Enter first person’s name and surname: Mauro Icardi
Enter second person’s name and surname: Kerem Aktürkoğlu
Friendship added between Mauro Icardi and Kerem Aktürkoğlu
```

5.4. Remove friendship:

```
===== 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 and surname: Mauro Icardi
Enter age: 31
Enter hobbies (comma-separated): football, celebration, reading
Person added: Mauro Icardi (Age: 31, Hobbies: [football,  celebration,  reading]) (Timestamp: Tue May 28 13:08:51 TRT 2024)
===== 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 and surname: Kerem Aktürkoğlu
Enter age: 25
Enter hobbies (comma-separated): football, celebration, writing
Person added: Kerem Aktürkoğlu (Age: 25, Hobbies: [football,  celebration,  writing]) (Timestamp: Tue May 28 13:09:37 TRT 2024)
===== 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: 3
Enter first person's name and surname: Mauro Icardi
Enter second person's name and surname: Kerem Aktürkoğlu
Friendship added between Mauro Icardi and Kerem Aktürkoğlu
===== 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 and surname: Kerem Aktürkoğlu
Enter second person's name and surname: Mauro Icardi
Friendship removed between Kerem Aktürkoğlu and Mauro Icardi
```

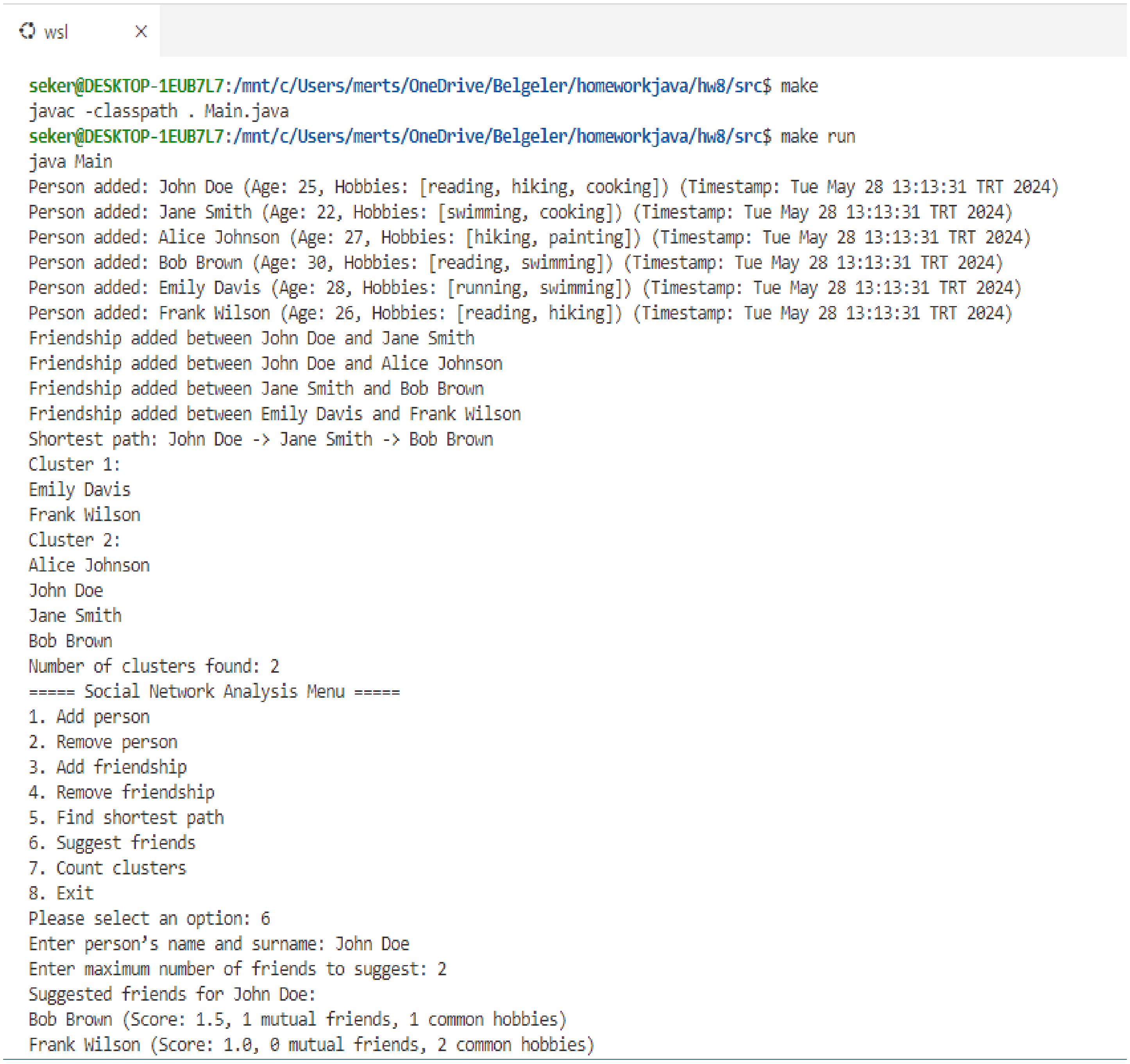
5.5. Find shortest path (using demo examples):



wsl

```
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make
javac -classpath . Main.java
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make run
java Main
Person added: John Doe (Age: 25, Hobbies: [reading, hiking, cooking]) (Timestamp: Tue May 28 13:12:01 TRT 2024)
Person added: Jane Smith (Age: 22, Hobbies: [swimming, cooking]) (Timestamp: Tue May 28 13:12:01 TRT 2024)
Person added: Alice Johnson (Age: 27, Hobbies: [hiking, painting]) (Timestamp: Tue May 28 13:12:01 TRT 2024)
Person added: Bob Brown (Age: 30, Hobbies: [reading, swimming]) (Timestamp: Tue May 28 13:12:01 TRT 2024)
Person added: Emily Davis (Age: 28, Hobbies: [running, swimming]) (Timestamp: Tue May 28 13:12:01 TRT 2024)
Person added: Frank Wilson (Age: 26, Hobbies: [reading, hiking]) (Timestamp: Tue May 28 13:12:01 TRT 2024)
Friendship added between John Doe and Jane Smith
Friendship added between John Doe and Alice Johnson
Friendship added between Jane Smith and Bob Brown
Friendship added between Emily Davis and Frank Wilson
Shortest path: John Doe -> Jane Smith -> Bob Brown
Cluster 1:
Emily Davis
Frank Wilson
Cluster 2:
Alice Johnson
John Doe
Jane Smith
Bob Brown
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: 5
Enter start person's name and surname: John Doe
Enter end person's name and surname: Bob Brown
Shortest path: John Doe -> Jane Smith -> Bob Brown
```

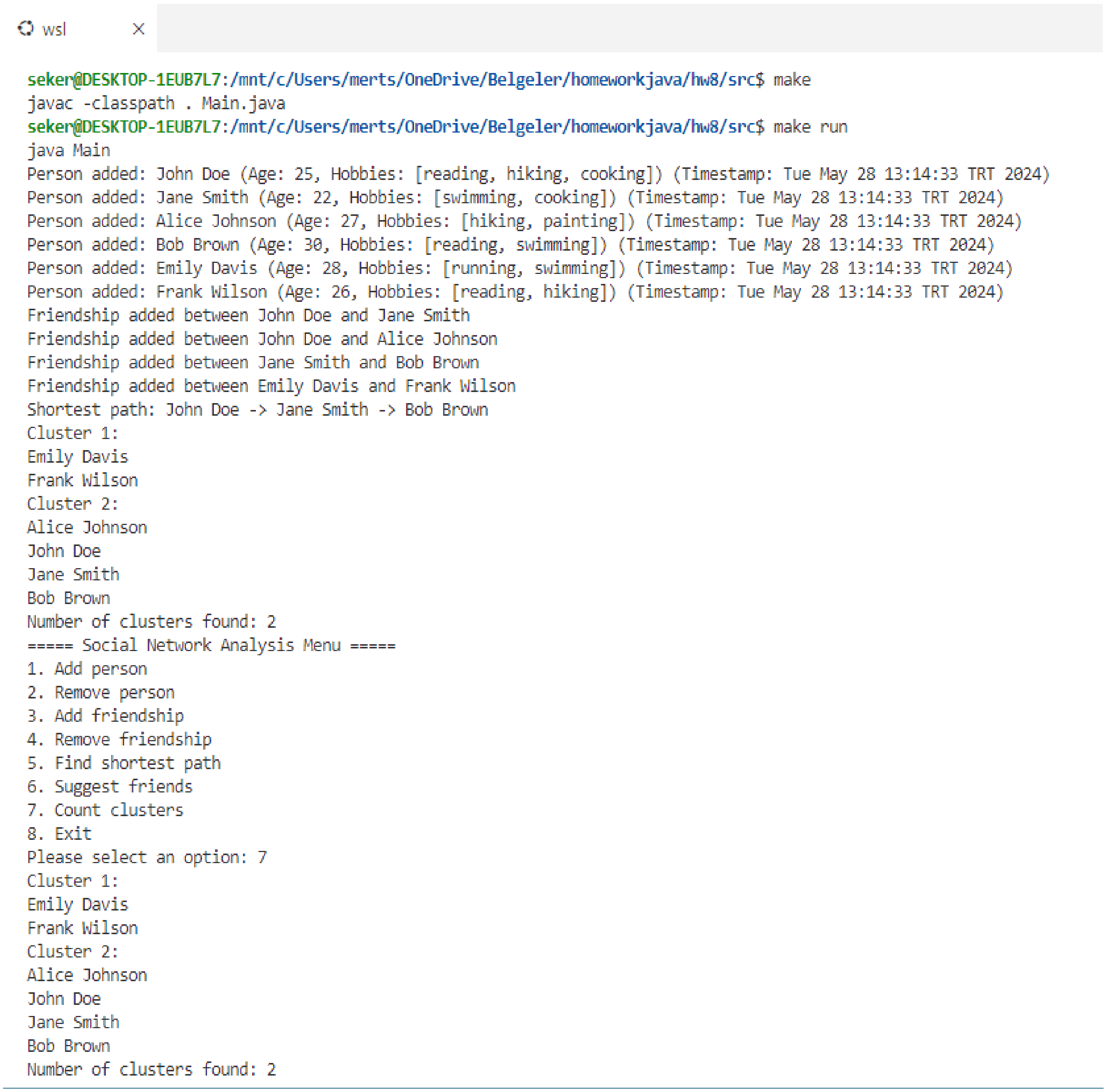
5.6. Suggest friends (using demo examples)::



wsl

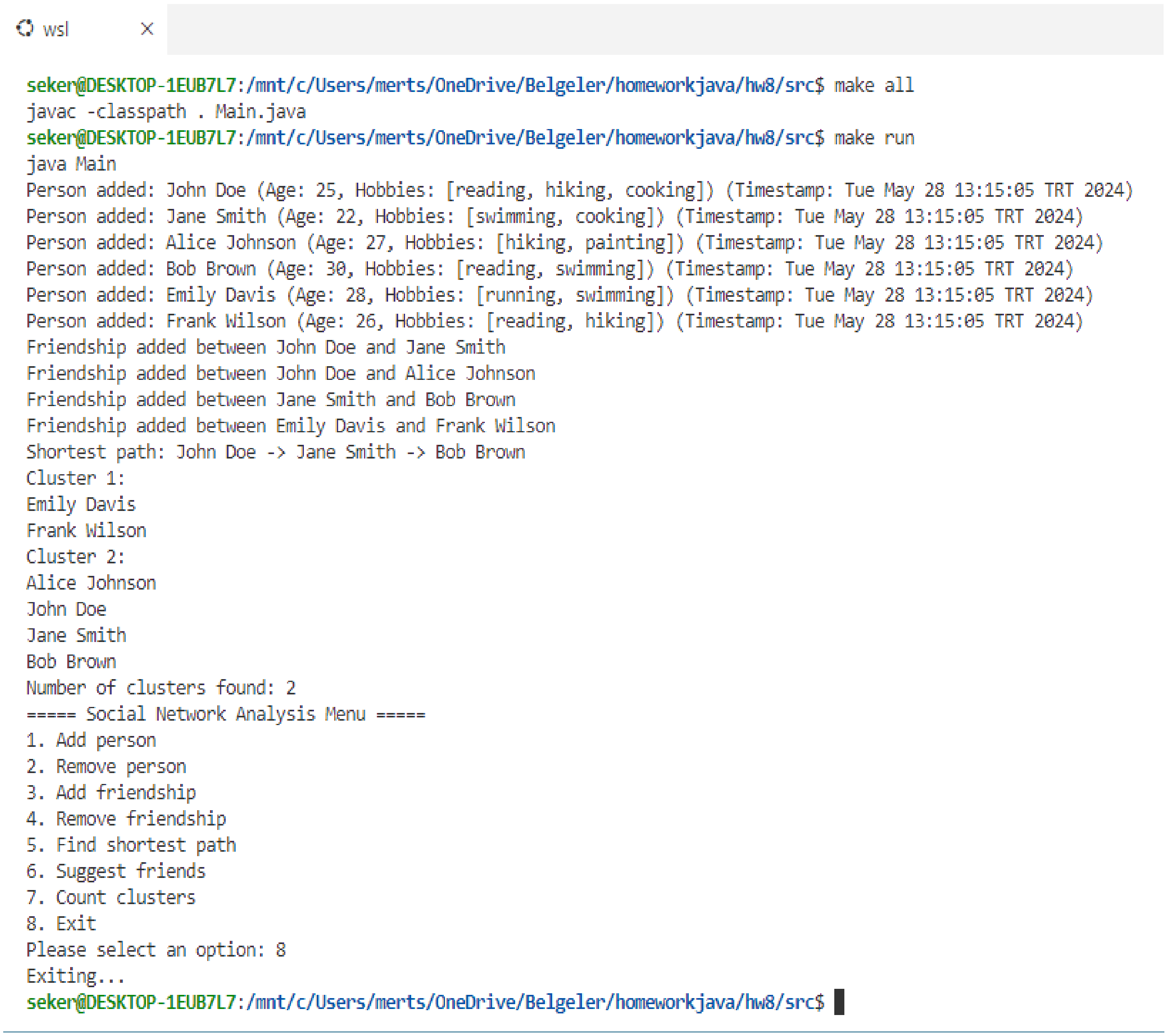
```
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make
javac -classpath . Main.java
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make run
java Main
Person added: John Doe (Age: 25, Hobbies: [reading, hiking, cooking]) (Timestamp: Tue May 28 13:13:31 TRT 2024)
Person added: Jane Smith (Age: 22, Hobbies: [swimming, cooking]) (Timestamp: Tue May 28 13:13:31 TRT 2024)
Person added: Alice Johnson (Age: 27, Hobbies: [hiking, painting]) (Timestamp: Tue May 28 13:13:31 TRT 2024)
Person added: Bob Brown (Age: 30, Hobbies: [reading, swimming]) (Timestamp: Tue May 28 13:13:31 TRT 2024)
Person added: Emily Davis (Age: 28, Hobbies: [running, swimming]) (Timestamp: Tue May 28 13:13:31 TRT 2024)
Person added: Frank Wilson (Age: 26, Hobbies: [reading, hiking]) (Timestamp: Tue May 28 13:13:31 TRT 2024)
Friendship added between John Doe and Jane Smith
Friendship added between John Doe and Alice Johnson
Friendship added between Jane Smith and Bob Brown
Friendship added between Emily Davis and Frank Wilson
Shortest path: John Doe -> Jane Smith -> Bob Brown
Cluster 1:
Emily Davis
Frank Wilson
Cluster 2:
Alice Johnson
John Doe
Jane Smith
Bob Brown
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: 6
Enter person's name and surname: John Doe
Enter maximum number of friends to suggest: 2
Suggested friends for John Doe:
Bob Brown (Score: 1.5, 1 mutual friends, 1 common hobbies)
Frank Wilson (Score: 1.0, 0 mutual friends, 2 common hobbies)
```

5.7. Count clusters (using demo examples)::



```
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make
javac -classpath . Main.java
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make run
java Main
Person added: John Doe (Age: 25, Hobbies: [reading, hiking, cooking]) (Timestamp: Tue May 28 13:14:33 TRT 2024)
Person added: Jane Smith (Age: 22, Hobbies: [swimming, cooking]) (Timestamp: Tue May 28 13:14:33 TRT 2024)
Person added: Alice Johnson (Age: 27, Hobbies: [hiking, painting]) (Timestamp: Tue May 28 13:14:33 TRT 2024)
Person added: Bob Brown (Age: 30, Hobbies: [reading, swimming]) (Timestamp: Tue May 28 13:14:33 TRT 2024)
Person added: Emily Davis (Age: 28, Hobbies: [running, swimming]) (Timestamp: Tue May 28 13:14:33 TRT 2024)
Person added: Frank Wilson (Age: 26, Hobbies: [reading, hiking]) (Timestamp: Tue May 28 13:14:33 TRT 2024)
Friendship added between John Doe and Jane Smith
Friendship added between John Doe and Alice Johnson
Friendship added between Jane Smith and Bob Brown
Friendship added between Emily Davis and Frank Wilson
Shortest path: John Doe -> Jane Smith -> Bob Brown
Cluster 1:
Emily Davis
Frank Wilson
Cluster 2:
Alice Johnson
John Doe
Jane Smith
Bob Brown
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: 7
Cluster 1:
Emily Davis
Frank Wilson
Cluster 2:
Alice Johnson
John Doe
Jane Smith
Bob Brown
Number of clusters found: 2
```

5.8. Exit:



wsl

```
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make all
javac -classpath . Main.java
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$ make run
java Main
Person added: John Doe (Age: 25, Hobbies: [reading, hiking, cooking]) (Timestamp: Tue May 28 13:15:05 TRT 2024)
Person added: Jane Smith (Age: 22, Hobbies: [swimming, cooking]) (Timestamp: Tue May 28 13:15:05 TRT 2024)
Person added: Alice Johnson (Age: 27, Hobbies: [hiking, painting]) (Timestamp: Tue May 28 13:15:05 TRT 2024)
Person added: Bob Brown (Age: 30, Hobbies: [reading, swimming]) (Timestamp: Tue May 28 13:15:05 TRT 2024)
Person added: Emily Davis (Age: 28, Hobbies: [running, swimming]) (Timestamp: Tue May 28 13:15:05 TRT 2024)
Person added: Frank Wilson (Age: 26, Hobbies: [reading, hiking]) (Timestamp: Tue May 28 13:15:05 TRT 2024)
Friendship added between John Doe and Jane Smith
Friendship added between John Doe and Alice Johnson
Friendship added between Jane Smith and Bob Brown
Friendship added between Emily Davis and Frank Wilson
Shortest path: John Doe -> Jane Smith -> Bob Brown
Cluster 1:
Emily Davis
Frank Wilson
Cluster 2:
Alice Johnson
John Doe
Jane Smith
Bob Brown
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...
seker@DESKTOP-1EUB7L7:/mnt/c/Users/merts/OneDrive/Belgeler/homeworkjava/hw8/src$
```