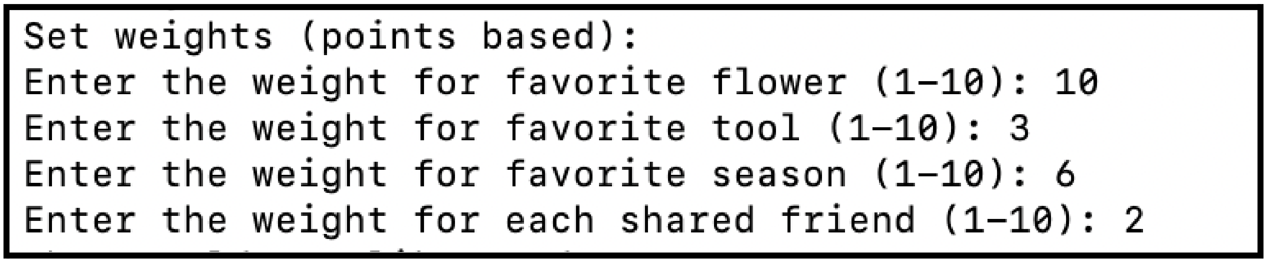
How to Run the Project:

1. Open your terminal or command prompt and navigate to the folder containing the PersonInfo.java file.
2. Execute the following commands in order:
   a. javac *.java
   b. java PersonInfo.java

Using the Program

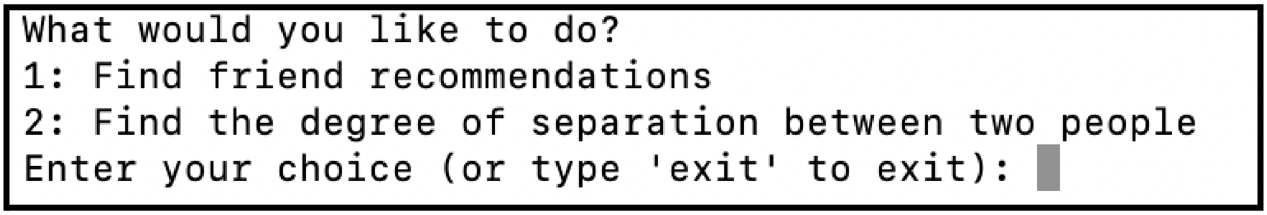
1. Setting Weights: Upon starting the program, the user will be prompted to assign weights for each interest found in the interests.csv file, as well as a weight for each common friend.



```
Set weights (points based):
Enter the weight for favorite flower (1-10): 10
Enter the weight for favorite tool (1-10): 3
Enter the weight for favorite season (1-10): 6
Enter the weight for each shared friend (1-10): 2
```

2. Main Menu: After setting weights, the user will enter a continuous loop where they can choose between two options:

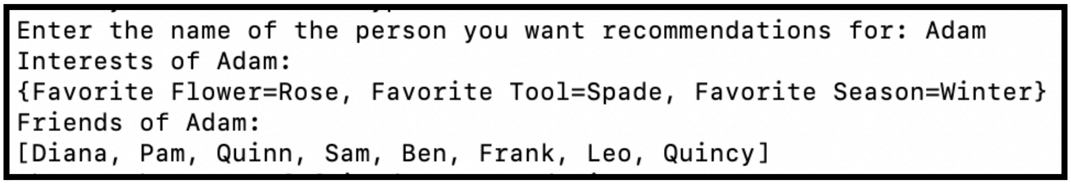
   - 1: Find friend recommendations
   - 2: Find the degree of separation between two people
   - Or, the user can type 'exit' to exit the program.



```
What would you like to do?
1: Find friend recommendations
2: Find the degree of separation between two people
Enter your choice (or type 'exit' to exit):
```

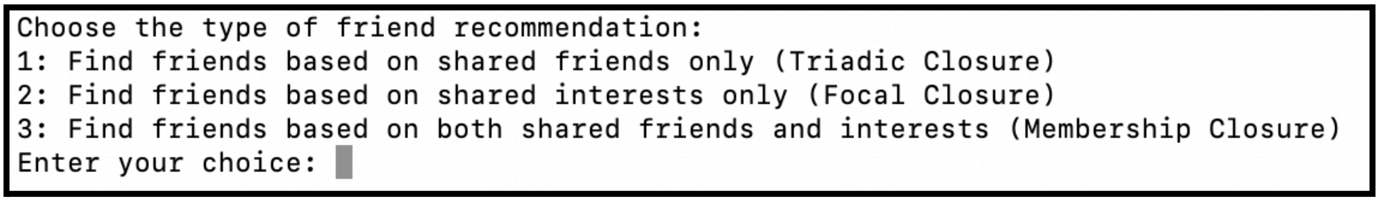
3. Finding Friend Recommendations

   - If the user selects option 1, they will be prompted to enter the name of the person they want recommendations for.
   - Once entered, the program will display the data it has about that person.



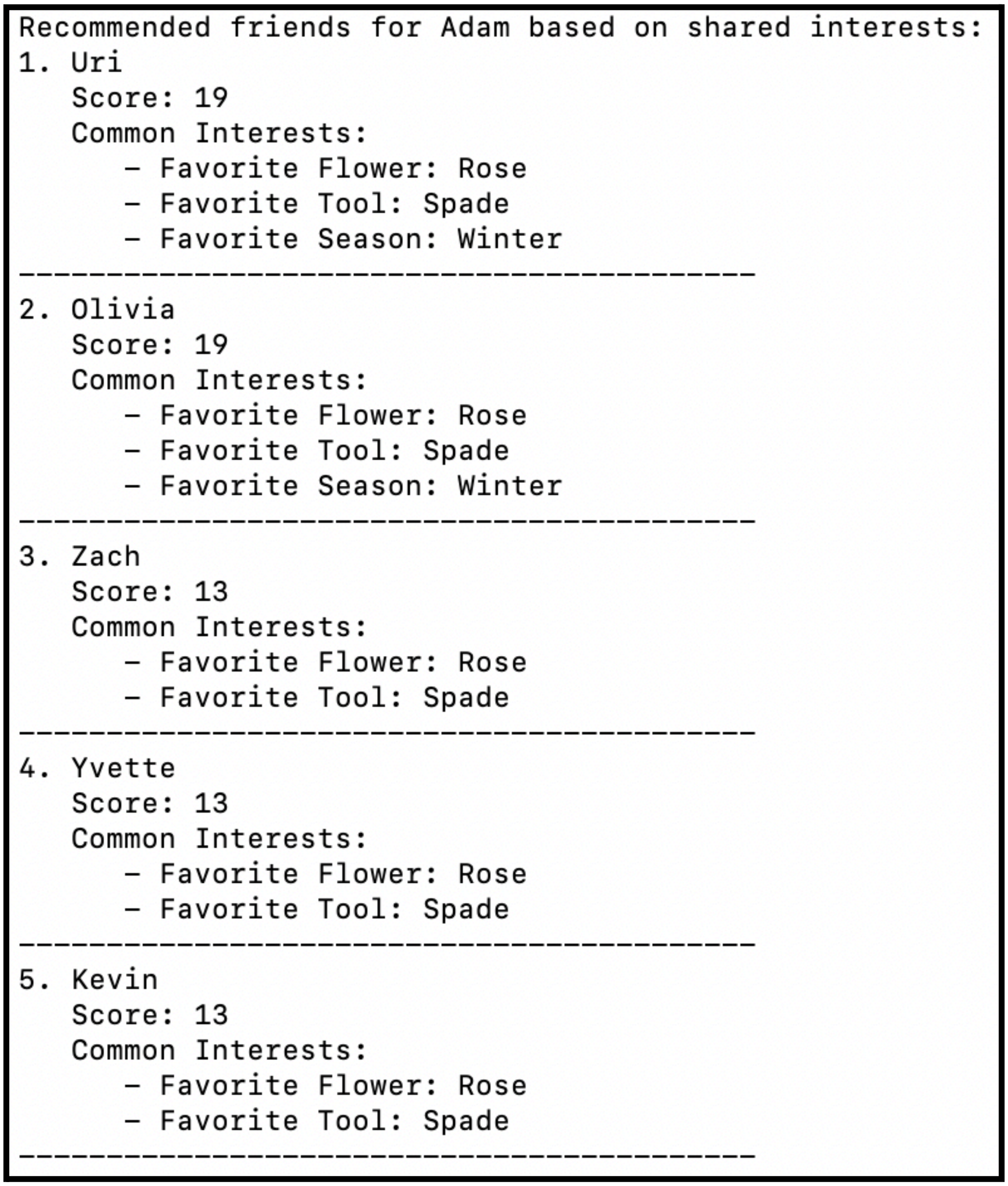
```
Enter the name of the person you want recommendations for: Adam
Interests of Adam:
{Favorite Flower=Rose, Favorite Tool=Spade, Favorite Season=Winter}
Friends of Adam:
[Diana, Pam, Quinn, Sam, Ben, Frank, Leo, Quincy]
```

- The user will then choose the type of recommendation they want (based on shared friends only, shared interests only, or both).



```
Choose the type of friend recommendation:
1: Find friends based on shared friends only (Triadic Closure)
2: Find friends based on shared interests only (Focal Closure)
3: Find friends based on both shared friends and interests (Membership Closure)
Enter your choice: 
```

- After making a selection, the program will display the top 5 recommendations based on the chosen criteria.
- For example, we want to find recommendations based on mutual interests, so we enter 2.

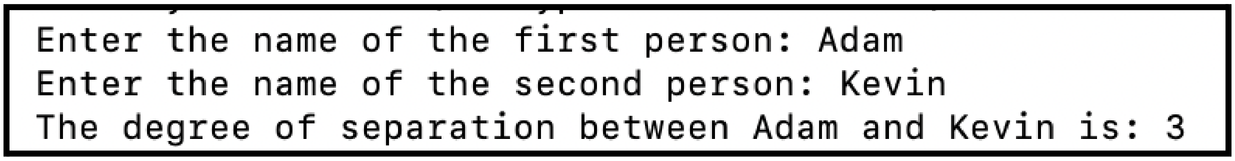


```
Recommended friends for Adam based on shared interests:
1. Uri
   Score: 19
   Common Interests:
      - Favorite Flower: Rose
      - Favorite Tool: Spade
      - Favorite Season: Winter
------------------------------------------
2. Olivia
   Score: 19
   Common Interests:
      - Favorite Flower: Rose
      - Favorite Tool: Spade
      - Favorite Season: Winter
------------------------------------------
3. Zach
   Score: 13
   Common Interests:
      - Favorite Flower: Rose
      - Favorite Tool: Spade
------------------------------------------
4. Yvette
   Score: 13
   Common Interests:
      - Favorite Flower: Rose
      - Favorite Tool: Spade
------------------------------------------
5. Kevin
   Score: 13
   Common Interests:
      - Favorite Flower: Rose
      - Favorite Tool: Spade
------------------------------------------
```

- The user will then return to the main menu, where they can select another option.

4. Finding Degree of Separation

    - If the user selects option 2, they will be prompted to enter the names of two people to find the degree of separation between them.
    - Once entered, the program will calculate and display the degree of separation between the two individuals.



```
Enter the name of the first person: Adam
Enter the name of the second person: Kevin
The degree of separation between Adam and Kevin is: 3
```

    - The user will then return to the main menu.

Files Used:

- The program is designed to work with any interests.csv file, as long as the columns are formatted as shown below:

Person, Interest1, Interest2, ..., InterestX

- The number of interests is flexible and can vary.
- Additionally, the friendships.csv file must be formatted as an adjacency matrix, as shown below:

```
Friend1, Friend2, ..., FriendX
Friend1, 0/1, 0/1, ..., 0/1
Friend2, 0/1, 0/1, ..., 0/1
...
FriendY, 0/1, 0/1, ..., 0/1
```

- In this matrix, the values 0 and 1 represent the absence or presence of a friendship, respectively.