Explain how your design will be able to store the profile information:

- There are 3 main sub classes of users: Adults, Young adults and Infants. All these classes extends the User abstract class which helps store shared profile information such as name, age, friendships, profile picture and status.
- Infant class extends Young Adults as they both need 2 parents in order to be connected to the network.
- All relations including guardian, dependant, friend and coparent is stored as ENUMS in RelationType class. Each user has a list of Relationships which contain a user and a Relation type associated with the user. This helps facilitate further enhancements such as user friend request status. Each relationship could potentially store user friend request status, and once both users confirm the request, they can be friends. Furthermore, this allows the guardians of a dependant to set parental controls and restrictions on the dependant. RelationTypes can be used to give escalated privileges over other users.

Explain how your class hierarchy will facilitate the network management

- Adding a new user: New users are created using the Factory design pattern. UserFactory class creates users depending on the age of the user. The AddNewUser menu takes care of the UI and the associated actions that relate to creating a new user.
- Adding a new friend: Each subclass of User overrides the addRelation abstract method with the associated constraints. If the constraints are met, a new Relationship class is added to the users relationships list. AddRelation method also ensures that the reciprocal relation is added to the newly added user.
- Deleting a friend: Each subclass of User overrides the deleteRelation abstract method with the associated contraints. Infants and young adults cant delete guardian relations and adults cant delete coparent or dependant relation. DeleteRelation method also ensures that the reciprocal relation is deleted on the other users relations.
- Deleting a user: When deleting a user, we stream through the users relations and ask them to delete the relationship with the user to be deleted. If the to be deleted user is a guardian to another user, the dependant would also have to be deleted as a dependant requires 2 adults to be present in the social network.

Explain the process by which your program can maintain the networks and find connections more efficiently:

- While friends are stored inside the user class, all users in the social network is accessed through a UserStore interface.
- UserService implements UserStore interface thus enabling us to use the adapter design pattern in order to change the retrieval of users in the future.
- UserService is a singleton class which can only be instantiated once, so that all class that attempt to access the list of users in the network has access to the same list. As the users are currently stored in an ArrayList, it could potentially cause Multi-threading

issues but that can be mitigated by instantiating the ArrayList by using Collections.sychronisedList(Arraylist).

- Finding connections can be straightforward as all we would need to do is to get user’s friend’s friends as suggested connections to the current user.
- Ideally, a graph data structure would be best suited for social networks as it helps maintain users and its connections in the most efficient manner. Nevertheless, the current implementation is extensible by allowing database relations to be mapped directly to it. Each user has a list of relationships which map a relation type to a user object reference (which would be user id in database).

Github Project URL: https://github.com/spacetj/assignment1