SocioNet

https://github.com/jainvis/Advanced-Programming---Semester-2/tree/master/assignment2

Design:

- Data interface
    - To provide an inbuilt data
    - As it only contains hard-coded data, it of type abstract
- MiniNet Class()
    - This is a JAVAfx class
    - The source code for GUI is defined here
    - This is to be executed to run the application
- Abstract Driver Class()
    - This is the class where all the action happens
    - This is acting as a controller class for GUI
    - This class keeps the user-interacting class unaware of the data handling and storage
    - Here we have used HashMap<Integer, Profile>
        - To contain the information of profiles
        - Integer is being used to define the unique ID for each profile
        - Profile is an abstract class created for information
    - All the methods such as addProfile(), updateProfile(), addConnection() etc. are defined in this class
- Abstract Profile Class()
    - This class is created to define instance variables such as name, age, status, photo
    - This class also contains the instance variable ArrayList<String> friendlist to contain the names of friends
    - Also, while entering the status, we may pass "_" value but never a null value
    - As photo is also optional, we have taken the liberty to remove it from the GUI as part of a functionality that may be defined in future
    - This class implements all the interfaces created to handle relations such as friend, parent, child, colleague, classmate. But the same have been kept as abstract as relations will be handled in each sub-class
    - This is the super class for the following two classes:
        - Adult
        - Child
        - YoungChild
- Adult Class()
    - Based on the age > 16, the profiles will be stored with reference to this class
    - This class contains instance variable of type ArrayList<String> to contain relevant relations
    - Interfaces of relations; friend, child, colleague, classmate are implemented with relevant handling
    - Parent relation has been kept empty as no actions are required here
- Child Class()
    - Based on age>2 and age <16, the profiles will be stored with reference to this class
    - This class also contains instance variables of type ArrayList<String> to contain relations
    - Interfaces of relations; friend, parent, classmate are implemented with relevant handling code
    - Colleague and Child relations have been kept empty as a child cannot have these relations
- YoungChild Class()
    - Based on age<2, the profiles will be stored with reference to this class

    - This class also contains instance variables of type ArrayList<String> to contain relations
    - Interfaces of relations; parent is implemented with relevant handling code
    - As a young child cannot have any other relations, rest of the relations are kept empty
- <u>Customised Exceptions</u>
    - Besides JAVA exceptions such as NumberFormatException, IOException etc. certain customised exception have been designed
    - MiniNetException()
        - Super-class of customised exceptions which extend JAVA class Exception
    - NoAvailableException()
        - Sub-class of MiniNetException
        - This is thrown when trying to make a couple when at least one member is not an adult
    - NoParentException()
        - Sub-class of MiniNetException
        - This exception is used in multiple situations
        - When trying to add a parent when parents already exists in profile
        - When trying to add a parent who is not an adult
        - When trying to add a profile (age < 16) and parents name are not given, as they are mandatory for child and young child profiles
        - When trying to add a person as parent and the profile does not exist on our network
        - When trying to delete a profile and dependents exists
    - NoProfileException()
        - Sub-Class of MiniNetException
        - When trying to add a profile as a connection and it does not exist in the network
    - NoRelationException()
        - Sub-Class of MiniNetException
        - While checking a connection between two people and no direct connection is found
    - NoSuchAgeException()
        - Sub-Class of MiniNetException
        - When trying to add a profile with negative age or age greater than 150
    - NotToBeChildException()
        - Sub-Class of MiniNetException
        - When trying to add an Adult profile as a child
    - NotToBeClassmatesException()
        - Sub-Class of MiniNetException
        - Trying to add a young child in a classmate relation
    - NotToBeCoupledException()
        - Sub-Class of MiniNetException
        - Trying to add a profile as a couple when a spouse already exists
    - NotToBeFriendsException()
        - Sub-Class of MiniNetException
        - When trying to add an adult and a child or young child as friends
        - When trying to add two children or two young children as friends and the difference in their age is more than 3 years
- Interfaces
    - Friend, Parent, Children, Classmate and Colleague
        - These interfaces contain methods to be implemented according to type of profile
        - Example: addFriend(), deleteFriend(), setFriend(), getFriend()
- Connections Class()
    - This class has been added specially to contain name and relations of a profile

- This is used in GUI to display the relations of a profile in TableView

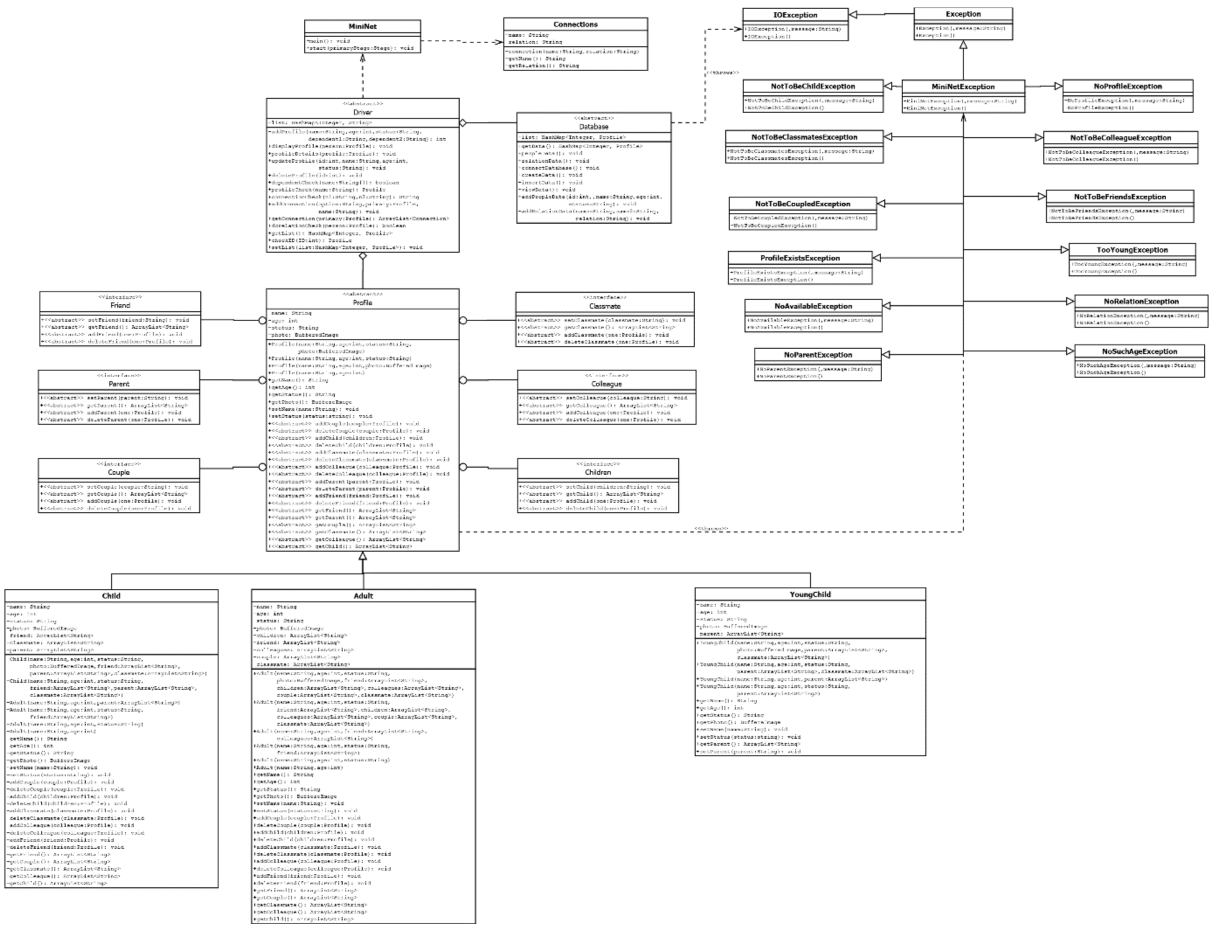
Assumptions:

While designing the application, we have taken a few assumptions:

- No child will have a single parent
- A child or Young Child profile cannot exist without parents
- A user can enter a string for variables such as name, status in capital or small letter, but we will store all the information in small letters
- All the people in our database will have unique names as our search will return the first name it encounters in the database while displaying profile
- The unique ID will be used to access any profile and we have taken 9999 as the maximum number of users in our application for now
- We have started the data with profile having unique ID, 1001 to 1008
- Although we have given an instance variable for photo, but for simplicity, we are not using the same on our network and thus displaying a message box to inform the user that the functionality is under development and soon be added.
- No instance variable of Profile, Adult, Child and Young Child is passed null
- A profile cannot be deleted if it has dependents

CLASS DESIGN **DIAGRAM** ON **NEXT PAGE**

CLASS DIAGRAM



Design Assumptions:

- Profile class is throwing all the exceptions, sub-classes of MiniNetException. Therefore, one arrow is shown in the design
- Although the child classes of Profile; Adult, Child and YoungChild have all the abstract methods but relations that are not to be implemented in a class are kept empty. Hence, the same are not shown in respective class operations. Example: methods of Parent interface are not shown in Adult class

PLEASE CONTINUE TO NEXT PAGE

DIFFERENCE BETWEEN ASSIGNMENT 1 & 2

ASSIGNMENT 1	**ASSIGNMENT 2**
1. Java console was used for user interaction	**1.** GUI (JAVAfx) is used for user interaction
2. Driver Class was doing all the user actions and running network	**2.** Driver Class is only used to control user actions and acts as a manager
3. Relations were handled only as a variable type	**3.** Interfaces are used to implement different relations for each profile type
4. Exception handling was limited as only JAVA exception were used	**4.** Customised Exceptions are used to define MiniNet Exceptions
5. Limited try-catch clauses were used, and if-else statement were used for error handling	**5.** Multiple try-catch statements are used to catch user-defined exceptions. Where appropriate, exception was thrown again.
6. Exceptions were all thrown and caught in Driver Class	**6.** Exception are thrown in Profile, Adult, Child, YoungChild and Driver Classes. Exception are handled in GUI

Discuss possible issues and/or comment on your knowledge gain and your learning experience through this group work

- **Natalie Sy**
  - Learned to code GUI using fxml and with scenebuilder, but didn't use these in the assignment
  - Learned the importance of using MVC (Model-View-Controller) pattern in GUI development
  - Learned file handling, but was only able to implement read data from people textfile and store in Hashmap
  - Learned database handling, but wasn't able to implement it successfully in the assignment
- **Vishesh Jain**
  - Learned GUI through JAVAfx
  - Learned the concept inheritance, polymorphism and abstract classes
  - Learned the concept of interfaces development and implementation
  - Learned .txt file handling through BufferReader and FileReader
  - Learned the concept of Object Oriented Programming