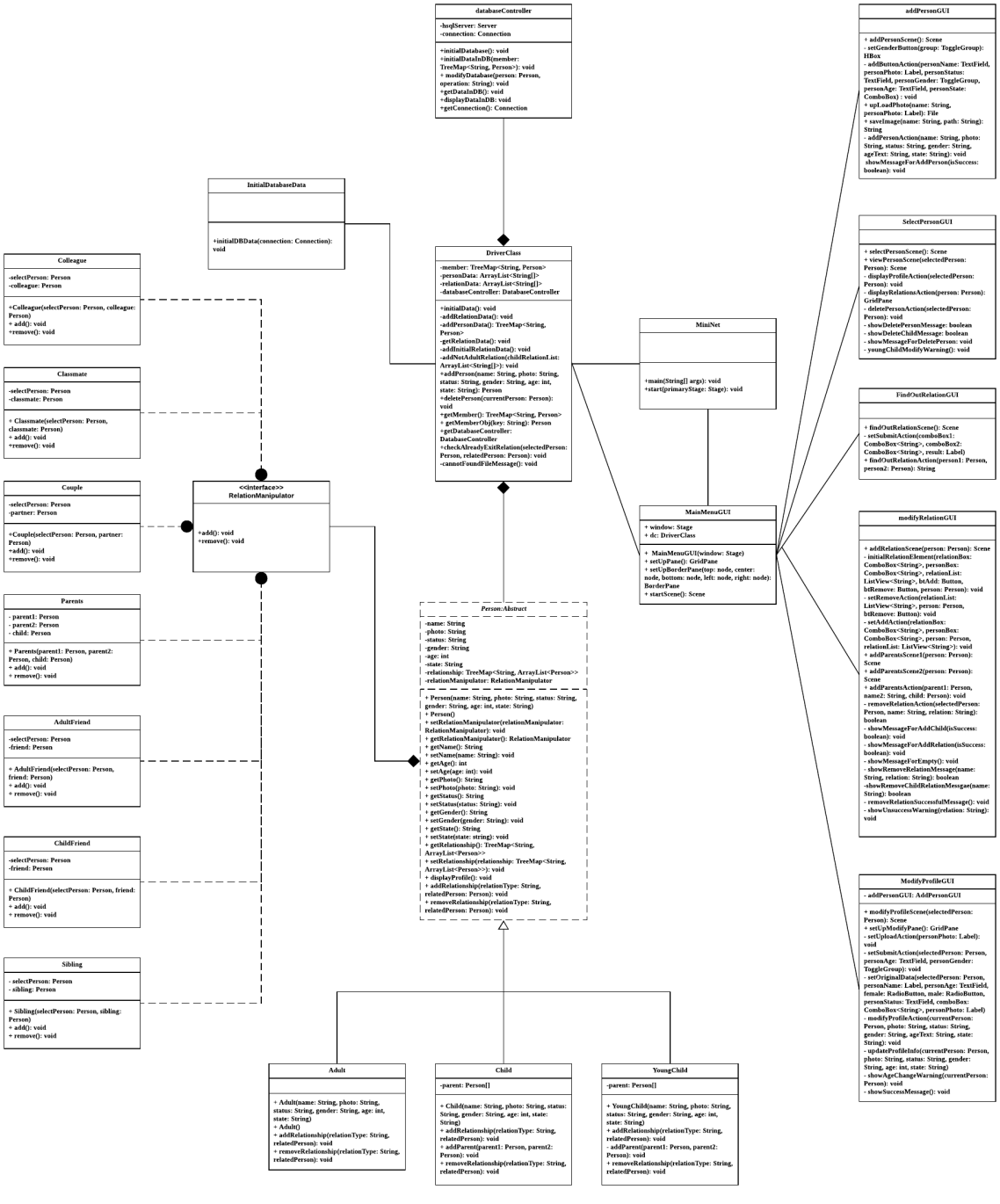
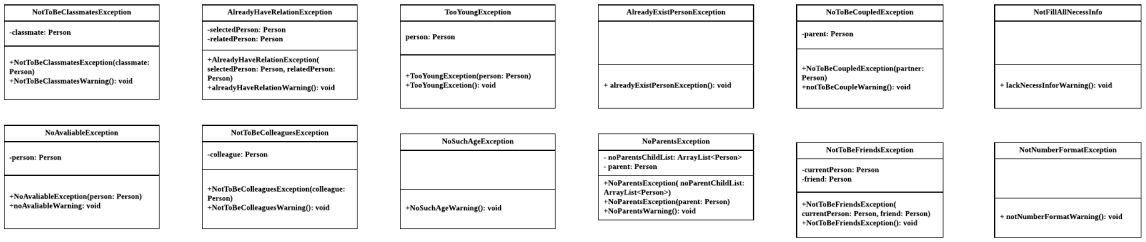
Student ID: s3666335 Name: CIFANG ZHANG Student ID: s3673099 Name: YILEI XU

MiniNet Class Diagram



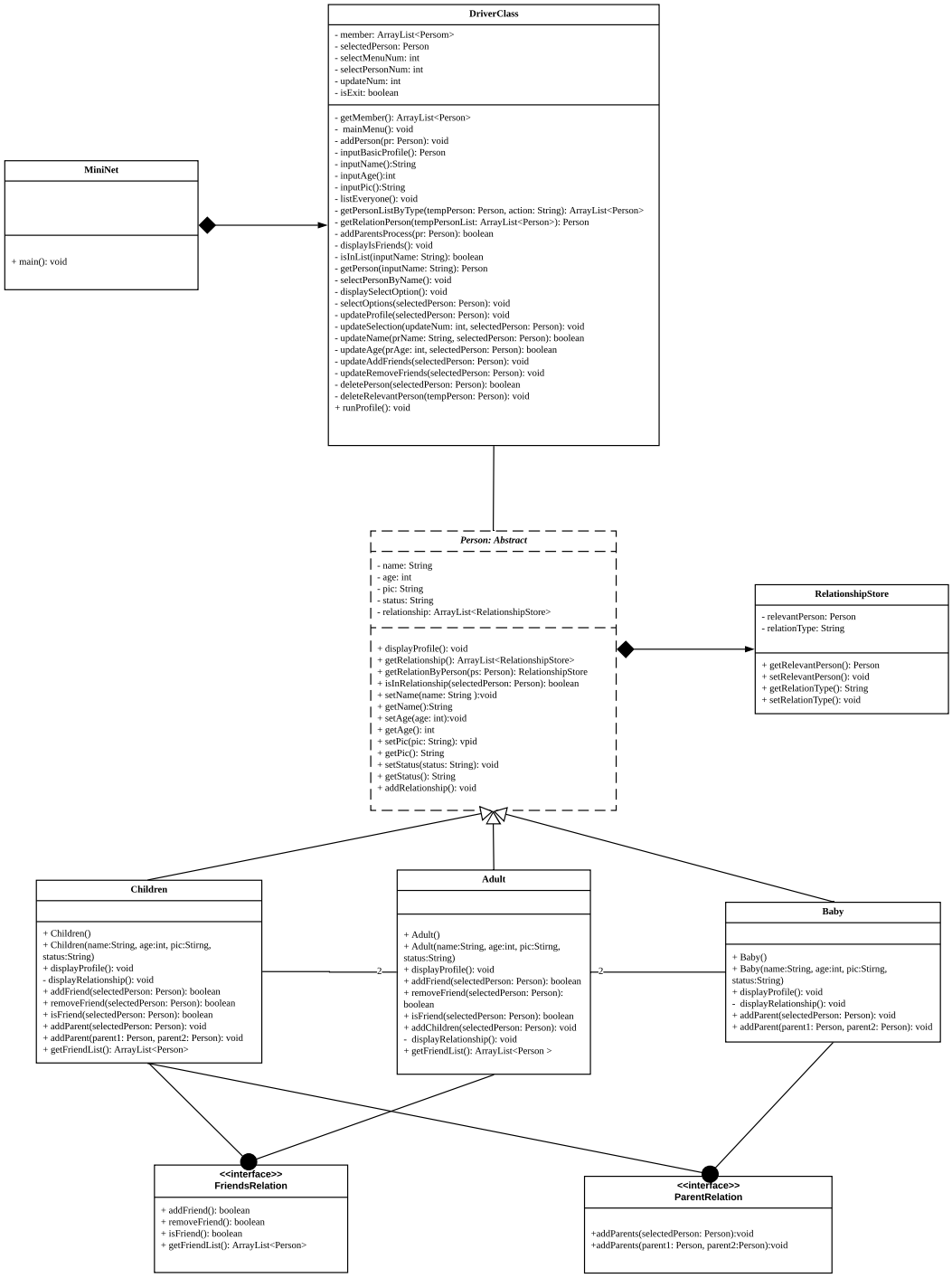
Exception Classes



Student ID: s3666335 Name: CIFANG ZHANG Student ID: s3673099 Name: YILEI XU

1. Explain the changes if you use a different design compared to your assignment 1

Class diagram of **Assignment 1**:



Compare to Assignment 1, our Assignment 2 class design has some changes as below:

- Create only one interface “RelationManipulator”. Interface “RelationManipulator” only have 2 methods - “add()”, “remove()”- which mean add relationship and remove relationship.
- Create relationship classes based on the assignment, such as “Parents”, “Couple”, “AdultFriend” and “ChildFriend”. Those classes implement the interface “RelationManipulator” and implements methods “add()” and “remove()” by different behaviours.
- Class “Adult”, “Child” and “YoungChild” extend abstract class “Person” and override the method “addRelationship(String relation, Person relatedPerson)” and “removeRelationship(String relation, Person relatedPerson)”. The parameter “relation” stands for the relationship type that the user wants to add/remove, and “relatedPerson” stands for the person who will be added/removed by the user.
- Add GUI classes, GUI classes are sorted by the scenes in the main menu.
- Use “static” variables (window: Stage, dc: DriverClass) and “static” method (setUpPane(), setUpBorderPane(), startScene()) in the MainMenuGUI class. This approach allows other GUI classes to access the methods and variables in the main GUI class. In addition, other GUI classes can access the methods in DriverClass by “dc: DriverClass” instance variable. dc: DriverClass should only be instanced once, otherwise the datas which stored in member: TreeMap<Person> would be initialised when we want to call the method through dc: DriverClass in other GUI classes. That is the reason why we make it static.
- Add Exception classes.
- Add database class “DatabaseController”. This class mainly response to connect with database and operate the data in the database (insert, update and delete data).

2. Explain how the new classes are organized

- The interface “RelationManipulator” contains only two methods (add() , remove()).
- Relationship classes (“Parents”, “AdultFriend”.etc) implement the interface “RelationManipulator”. And based on different type of relation, implements different operation to add/ remove relations in add()/remove().
- “Person” abstract class and interface “RelationManipulator” are composition relationship, in the “Person” class we declare “RelationManipulator” variable, then create getter and setter method to access or mutate the relation data.
- “Adult”, “Child” and “YoungChild” are subclasses of abstract class “Person”. Because different types of person have different relations, the different relationship object is created (AdultFriends, ChildFriends, Classmate, etc) and call the object’s own “add()”/ “remove()” method. This process is implemented in overriding methods “addRelationship(relation, selectedPerson)” and “removeRelationship(relation, selectedPerson)” of “Adult”, “Child” and “YoungChild” classes based on the passed parameters “relation”. In this case, different types of person can only call the overridding method to operate their relationship automatically.
- GUI classes include “MainMenu” and other sub-menu classes, the sub-menu classes associated with the “MainMenu” class by using its static variables and methods.
- The “MainMenu” class has composition relationship with the “DriverClass”, by creating a “DriverClass” object, the “MainMenu” class is able to access the method in “DriverClass”.
- Different Exception classes associate with different classes’ method based on what kind of exception they need to handle.
- The database class “DatabaseController” associated with the “DriverClass”.

3. Explain the process by which your program will interact with user and external data source to run a game.

- The application has some initial data of people and their relationship which is stored in the file (people.txt, relations.txt), and the database (people information).
- When the user runs the program, the data will be initialized by calling the initialData() method in the DriverClass. If people.txt can be found, the system would read the people information data from the people.txt file and initial the database. If the people.txt file cannot be found, the people information would be read from the database which is initialised by hard code. If people.txt cannot be found and database cannot be connected, it would initiate an empty people list and give an error message.
- All the images for people's profile photo or the background of GUI, are stored in the "image" folder. The images would be got by relative path when it is shown in the application. For example, a person who has a photo to show, the relative path would be the relative path plus the photo name and type of image (eg. "./image/person.jpg"). When the user adds a new person to the network, they also can upload an image from their computer. The application would copy the selected image into the "image" folder and rename it (the name would be person's name plus the image's type). The image name would also be stored in person's profile.
- The system has "add person" and "modify person" profile functions., when the user goes to those scenes, it shows the interface of add/ modify person profile, and then it will get the user's input data (person's name, age, gender, status, photo, state) and save them to the memory of the system and database.
- The relation data (from the relation.txt file or operating by the user, like add relation, remove relation) would only be stored in the memory.

4. How to run this application

Double click MiniNet.jar file which is under MinNet Folder.