Advanced Programming Assignment1

Student No: s3673099 Student Name: YILEI XU Contribution: 50%
Student No: s3666335 Student Name: CIFANG ZHANG Contribution: 50%

Through this assignment, we have learnt more about the characteristic of Java, which is an Object-oriented programming language. In the previous Java Programming Fundamental subject, we have learnt some basic knowledge about Java programming. And based on the fundamental knowledge we have got, when we were doing the Assignment of Advanced Java, we started to realise the importance of Object-oriented programming design.

The answer to the questions:

Q1. Explain how your design will be able to store the profile information.

A1: Each person is an object of its relevant class, in the class constructor, it has four parameters, which are: name, age, pic and status. The information will store in the object when we create the person. Each person also has a composition relationship with the RelationshipStore class, each object in this class has two instance variables, which are releventPerson and relationshipType(friends/couple/parents), there is an ArrayList in Person class, when the subclass implement it, it can store those relationship object in the ArrayList for each person. In the display method, it can print the profile of the person and the relationship the person has.

Q2. Explain how your class hierarchy will facilitate the network management

A2: The class design is like this:

Person is an abstract class, its subclass Adult/Children/Baby extends the abstract class.
FriendRelation and ParentRelation are two interfaces, their function is to do the actions about friends or the actions about parent.
RelationshipStore is a class for store the relationship of a person.
DriverClass is the class which has most of the methods to achieve the function.

MiniNet is the class which has the main method, it will run the project.

The subclass Adult/Children/Baby can have their object, which can reuse the constructor of Person class, and those subclasses override the method in Person class. It can reduce repeat code, and make the code more readable.

Because we need to operate the friends and parents many times, so use interface, and let the concrete class to implement them can be more efficient, and also reduce the repeat code. Furthermore, if we need to add more class to use those function, the interface is easy to be implemented, and we can implement them as many as we need, so it is easier to maintain the system.

The RelationshipStore and Person class has composition relationship, Person has RelationshipStore, to make the RelationshipStore independent instead of just create an ArrayList in the Person class is because it is easy to maintain, for example, we can use the object to call the method in this class to get the relationship.

The DriverClass has many methods, to avoid redundancy, we create some reusable method in other classes, so that in the DriverClass we can just call them straight away. It makes the code more readable.

When we were doing the coding, we found that if we want to add friend, we need to check if the person is inside the network, and check if it is an adult (or for children, if it is a children object), then check if this object is already in the person's friend list. Those conditions make the code longer and not readable. We redesigned the code, make the same code to become a method, put them in the abstract class or interface, reuse them in the subclass, and make the constraint that user can only add friends from the Adult (or Children) list. The implement of polymorphism makes the code shorter and readable, and also easy to maintain.

MiniNet is always clear and short, to achieve the encapsulated purpose of OO design.

Q3. Explain the process by which your program can maintain the networks and find connections more efficiently.

A3: To maintain the networks we use different classes to separate different type of person, the abstract class Person can make sure subclass reuse the constructor and

method, and separate the subclass can make sure they can do different implement, and keep the privacy and security.

The using of interface and composition (RelationshipStore) makes the network easy to maintain. If there are some updates in this network, it is easy to reuse them.

In order to find connections more efficiently, we use the RelationshipStore to store the relevant person and the relationship type, in the Person class we create the ArrayList to store all the relationship of the person, and in the subclasses, we create the object of those people. Since we have the list of relationship, it is easy to use a method to get the relevant person, and check relationship type, if the type is correct, there is a method to get the person and then we can do any actions to it.

Users can create the connections between people when they add a person, also, they can edit friend relationship by select the person by the name.

Users can add friend by selecting the menu, the person is added must inside the network and be the same type person. We design two people can only have single one relationship, which means when they are a couple or parents and children relationship, they cannot become friends. When two friends become a couple, we change their relationship type from friends to couple.

Users can remove friends from the person's friends list, this action will not delete this friend from the network, it only changed the person's relationship, and also change the relationship list of that friend.

Users can delete a person, the relationship of the person will be deleted too. If the person has children, the system will ask the user again to confirm the delete action, then the children will be deleted together with the person.

Those methods and the design can make the connection easily and maintain the connection easily.