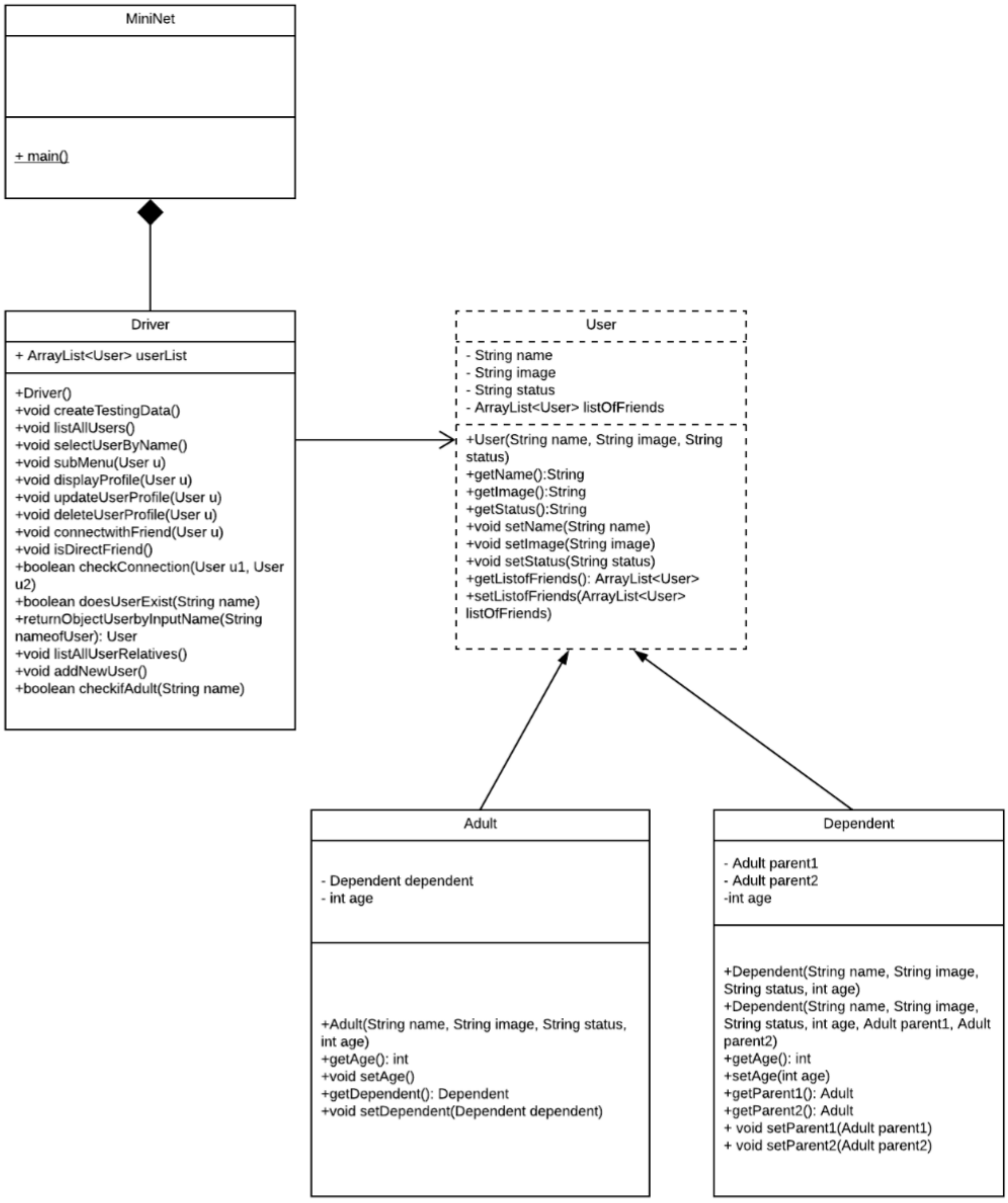
**Student Number: s3394108**

# 1.Class Diagram


MiniNet
+ main()
Driver
+ ArrayList<User> userList
+Driver()
+void createTestingData()
+void listAllUsers()
+void selectUserByName()
+void subMenu(User u)
+void displayProfile(User u)
+void updateUserProfile(User u)
+void deleteUserProfile(User u)
+void connectwithFriend(User u)
+void isDirectFriend()
+boolean checkConnection(User u1, User u2)
+boolean doesUserExist(String name)
+returnObjectUserbyInputName(String nameofUser): User
+void listAllUserRelatives()
+void addNewUser()
+boolean checkifAdult(String name)
User
- String name
- String image
- String status
- ArrayList<User> listOfFriends
+User(String name, String image, String status)
+getName():String
+getImage():String
+getStatus():String
+void setName(String name)
+void setImage(String image)
+void setStatus(String status)
+getListofFriends(): ArrayList<User>
+setListofFriends(ArrayList<User> listOfFriends)
Adult
- Dependent dependent
- int age
+Adult(String name, String image, String status, int age)
+getAge(): int
+void setAge()
+getDependent(): Dependent
+void setDependent(Dependent dependent)
Dependent
- Adult parent1
- Adult parent2
-int age
+Dependent(String name, String image, String status, int age)
+Dependent(String name, String image, String status, int age, Adult parent1, Adult parent2)
+getAge(): int
+setAge(int age)
+getParent1(): Adult
+getParent2(): Adult
+ void setParent1(Adult parent1)
+ void setParent2(Adult parent2)

## 2. Rationale behind the design

With the above design, it can be clearly seen that all of the user profile information (*name, status, image and list of friends*) will be stored under variables of objects which its type is User(they are instance of User class). These User objects then will be stored in an ArrayList name **userList.** This **userList** can be updated by using method add() or remove. The objects in the **userList** can be retrieved by running a for-loop through the **userList.** Hence their variables could be easily retrieved or modified by different accessors and mutators (getName(), getStatus(), getImage(), etc.)

There is a special variable for each User object which is called **listOfFriends.** The type of this variable is ArrayList, which allows this variable to store references that points to the other User objects (who are their friends). This ArrayList **listOfFriends** can be update by using the method add(). This listOfFriends also be retrieved and modified by accessors and mutators (getListoffFriends() or setListofFriends())

It is noticed that the User class in this design is only an abstract class. Two classes which are Adult and Dependent are two subclasses which inherited all the available methods as well as having their own constructors that receive slightly different number of parameters. For example, the constructor for Adult class will require a parameter: int *age* and the constructor for the Dependent class will require three more additional parameters: int *age*, Adult *parent1*, Adult *parent2.*

The design of this program is based on an MVC model which the controller will be the Driver class where all the user interaction occurs as well as it utilise other classes to manage the social network. As Driver class, the class itself contains many different methods, both void methods that execute the function based on user choice (such as *listAllUser(), selectUserbyName(), displayProfile(), updateUserProfile(), deleteUserProfile()* etc.) as well as methods that support the compiling process (such as *createTestingData(), submenu(), doesUserExist(), checkConnection(), checkIfAdult()).*

This Driver which contains Menu that prompts the user to input their choice as well as provide data for user profile. Based on what input data is, the methods of Driver class will behave accordingly. For example, based on the input for variable *age, the addNewUser() method will* decide to create objects that is instance of Adult or Dependent.

Working together, all the methods such as *doesUserExist(String name), returnObjectbyInputName(String nameofUser)* or *checkconnection(User u1, User u2),* allows the program to clarify and find connections between User easily.

## 3. GitHub

https://github.com/rmit-s3394108-Thuy-Pham/AP_assignment1/releases/tag/v0.1