TECHNICAL REPORT

ADTS:

- **Hashtable:** We are using a hash table that will use a user's username as the key and their index in the all vector as their value. We are using this data structure because it will make it easier when searching for a specific username.

- **Vector:** We are using multiple vectors that will store all the users associated to the client's Account. We need the different vectors because each will store a different "group" of users (followers, following, all, client not following back, not following client back). These vectors will be used for when we are displaying the information in our GUI.

CLASSES:

- **AboutPanel:** The starting panel in the GUI that will explain to the client how to use the program. This is also the panel that will ask the client to enter their username and password to their Instagram account.

    - **ButtonListener:** A listener for when the login button is clicked so that the program will know to search for the account in the database and get its information.

    - **PasswordListener:** A listener for when a password has been entered so that the program will know update the password of the account.

    - **UsernameListener:** A listener for when a username has been entered so that the program will know to update the username of the account.

- **Account:** This is the main class of the program. It stores all the information about the client's followers/following users, along with whether the program has access to the account information.

- **FollowersPanel:** The panel in the GUI that will display the followers associated with the client's Account. It will display all the followers that the client's Account does not follow back in a separate list.

    - **ButtonListener:** A listener for when the refresh button is clicked so that the information (labels and lists) know to refresh themselves.

- **FollowingPanel:** The panel in the GUI that will display the users the client's Account follows. It will display all the users that do not follow back the client's Account in a separate list.

    - **ButtonListener:** A listener for when the refresh button is clicked so that the information (labels and lists) know to refresh themselves.

TECHNICAL REPORT

- **InstagramGUI:** The class that will create the frame for the GUI that will allow the client to interact with the program.
- **SearchPanel:** The panel in the GUI that will let the client search a specific username to get their account's relationship to that username.
    - **SearchListener:** A listener for when a username has been entered so that the program will know to search for that username and return the results of the search.
- **TabbedPanel:** The panel that will hold all the smaller panels (About, Followers, Following, Search) in a tabbed format.
- **Testing:** The class that tests all the methods that create the data structures in Account.
- **User:** The class that will create a "user" object to represent all the users that are associated with the client's Instagram account. This object will hold the user's username, name, and relationship status.
- **WrongAccountInfoException:** The exception that is thrown when incorrect account information (username and password) are inputted for an Account.

MAIN METHODS:

Account:

- **getInfo( )** : Void method that gets all the account information for Account by using the helper method readDatabase( ). It will also catch the two exceptions that readDatabase may throw and print out specific messages for each exception.
- **readDatabase( )** : Void helper method that looks for the Account in the database text file (database.txt) containing all the accounts if the Account has access to the account information. This method will throw FileNotFoundException if database.txt cannot be found and WrongAccountInfoException if the Account doesn't have access to the account information. If the Account has access to its account information, it will use the helper method readFiles(String[ ] info) to gather information on the account. If the Account does not have access to its account information, it will first use the helper method empty( ) to empty out all the data structures and then throw the WrongAccountInfoException.
- **readFiles(String[ ] info)** : Void helper method that takes in an array of Strings, each String contains information about an user associated with the client's Account. It will then iterate over all the users and use the helper method readUsers(String userInfo) to create a User

object as needed and add that object to the hash Hashtable, all Vector, and either the following or followers Vector. After all the different users associated with the account has been read, it will then use the helper method addToVectors( ) to add the various User objects to the cnfb (client not following back) and nfcb (not following client back) Vectors as necessary.

- **readUser(String userInfo)** : Void helper method that takes in a String of a user's information in the format of "username, name, status" and then will create a User object for it if that username doesn't already exist in the hash Hashtable. After creating the User object, it will add it to the all Vector and either the following or followers Vector. If the User object already exists in the hash Hashtable and the all Vector, then we will reset the User's status to 0, update that information in the all Vector, and add the User to the mutuals Vector.

- **addToVectors( )** : Void helper method that iterates through all of the users in the all Vector, and adds them to the nfcb Vector if they are not following the client back, or adds them to the cnfb Vector if the client is not following them back. No action is taken if they are mutual followers.

- **calculatePercent( )** : Void method that calculates the percent of the client's followers that are not following the client back, and stores it in an instance variable nfcbPercent. Also calculates the percent of the client's followers that the client is not following back and stores it in the instance variable cnfbPercent.

- **search(String searchUser)** : Will search for the searchUser username by looking up its value in the hash Hashtable. If the searchUser exists in hash, then we will use the value given back as and look up that user's relationship to Account and return a String describing that relationship. If the searchUser does not exist in hash, then we will return a String saying that the user and Account are not associated with each other.

- **empty( )** : Void helper method that empties out all the data structures (hash, all, following, followers, mutuals, nfcb, cnfb).

User:

- **returnStatus( )** : Returns the status of the user in a String that is a sentence. This is necessary for the GUI implementation of our program.