Readme:

Main class for starting the Trading System project resides inside /src folder named ***TradingMain.java*** , you can use default (14.0.2) JDK to build and run. As you run the main class, you would see prompts asking you to login that resides in class StartMenuSystem.java, and you will be direct to 3 different login/signup menu given your input to command line and will see different aspect in terms of user functionality of the program. Given time constraint unfortunately I was not able to implement all features required in phase1&2, but I will explain my current status and plan to implement them given more time at the end of the document.

I would like to guide through the UML diagram and adding notes to my implementation:

1. Sign in
    a. you will be asked to enter username, password and type of account to access the full functionality for a client user (main interface implemented in class ClientUserSystem.java) or admin user(main interface implemented in class AdminSystem.java). Note that all user input information in this project will be case sensitive and space sensitive.
    Before proceeding, I suggest to create your own account via #2. But I have provided a list of username and passwords that you can log into in the UserAccounts.txt file.
    b. after login with a Client User account, you will see the following prompt:
        i. Look for things to trade (I will explain the detail for Tradying System later)
        ii. Browse Pending Transactions : browse current user's pending transactions (confirmed appointment), need to be confirmed by both users to proceed to a completed transaction (the latter part not complete yet).
        iii. View Recently Traded Items: view 3 recently traded items
        iv. View Most Frequent Trade Partners: 1 most frequent trade partner
        v. Browse Pending Appointments: browsing user initiated appointment requests, need to be confirmed by the other user to become a pending transaction.
        vi. Browse Inventory : browsing all users inventories (items for lending).
        vii. View your wish list
        viii. View your lending list: view your confirmed items (by admin) for lending
        ix. Request to add an item: need to be approved by an admin to become part of lending list
        x. Request admin to unfreeze account: not implemented yet, but should be simply maintaining a request to unfreeze ArrayList. Admin already have the ability to unfreeze any account.
        xi. View your Threshold limits: Phase 2. Not finished completely.

xii. Recommended Items to Lend: Phase 2. Upon entering item id you are interested in, the system will return the item's owner's wishlist for your reference. If the owner does not have a wishlist, the system will return a message. Ideally we will calculate the frequency of items in wishlist or other algorithms to give a "cold-start" solution for this functionality, but a simple message indicating the user does not have any wishlist would be informative enough.
xiii. Log Out

c. after login with a Admin User account (regardless of initial or subsequent admin user), you will see the following prompt.
   i. View Pending Users: view users in status pending (same functionality as status frozen, but need to be confirmed by admin)
   ii. View Pending Items: view items that are requested by users to add to their lending list
   iii. Review Pending Transactions: view all pending transaction (nothing admin can do here though)
   iv. Change Thresholds: Phase 2 change threshold value, will discuss how I approach this in more details later
   v. View all client users: pull all client user information
   vi. Unfreeze User
   vii. Freeze User
   viii. Display All Admins: pull all admin user information
   ix. View User Threshold: view threshold values for users
   x. Add New Admin User: will only work if its a initial admin user
   xi. Undo Pending Appointments: Phase 2. Delete pending appointments initiated by a user.
   xii. Undo Pending Items: Phase 2. Delete pending items requested by a user
   xiii. Log Out

2. Sing up
   a. Will prompt messages that ask you to enter username and password. username cannot be the same of any existing user.
3. Look around (Visitor Mode for Phase2)
   a. No sign in required, will directly enter a new prompt menu for visitors. have the following functionality/options:
      i. Browse Inventory
      ii. View All registered Client Users
      iii. View Threshold Values
      iv. Return to Main Menu
   b. all functionality above correspond to signed-in users.

Below is a more detailed explanation in the trading system implementation:

On entering the trading menu mentioned above in client user interface menu, you will see the following prompt:
1. Add an item to wishlist: add items to wishlist
2. Request to Trade: request a two-way trade
3. Request to Borrow: request a one-way trade
4. Confirm an Appointment: confirm pending appointment and will move the appointment to transactionticket with same information but different id. after both users involved have confirmed.
5. Return to Client menu

Note that again, due to time constraint, I was not able to finish fix the bugs for temporary transactions so all trading was performed in a permanent manner. However, the temporary transactions will be able to extend from the permanent transaction once a date time function is added to the trading class and the system will record current time based on the latest transaction/appointment. In other words, I plan to update system clock each time an event with time happens and update the following:

1. whether a user status should be set to pending (or frozen after confirmed by admin)
2. whether a pending transaction should be canceled
3. whether an item from temporary trade should be returned to its original owner.

Detailed implementation documentation for features implemented for Phase2:

1. Admin ability to undo actions: I was able to implement 2 undo actions, one for delete pending items requested by user. Another is to delete pending appointments initiated by a client user. This is done through calling functions in AdminSystem.java which removes pendingAppointList and PendingItemList maintained in client user entity.
2. Automatically Suggest Items to lend: This is done through the user menu in ClienUserSystem.java which pulls all inventory information for users to view and select items. Once an item is selected, the system roots the user with that item and provides his/her wish list. If no items in the wishlist the system will output corresponding information instead.
3. Visitor Account: visitor account is done through simply adding a visitor account and new visitor interface menu with less functionality. Once a look around request triggers, the system will automatically log in the visitor account. This design is to ensure we can easily extend the functionality a visitor can do through new implementation or an admin account.
4. Threshold Value Adjustment: this functionality is partially done, main functionality is maintained in the admin interface. A ThresholdManager.java account was created and there are four different categories of threshold that can be adjusted or added: user-specific threshold (e.g. number of incomplete transactions before getting freeze) and universal threshold (e.g. number of lending items before a user can borrow). Currently there is some inconsistency in using the threshold hashmap and did not have the chance to use “add” feature. But this could also be easily extended in the future.

5. New account status: A new account status VIP was created for awarding users who have more than 3 items in the lending list. VIP status have all features as a usual client user but can also be seen by other users while initiating appointments etc. I also planned to give the VIP user more privilege in terms of threshold (user-specific threshold) in future for example.Also, the visitor account itself has an “invisible” status that differs from other user accounts such that it can only be viewed by admin users.

Potential Bugs:
Phase 1 code uses .ser format files to save serialized information but did not specify the class `serialVersionUID` so it is not guaranteed it will work on every single compiler and might cause unexpected exceptions and crashes some partial functionality. However given time constraints I could not fix this issue so if any problem prompts unexpectedly any steps of your reproducing, please contact me and I would like to do a live demo / video recording on the program if you’re fine with it.
Reference:
https://stackoverflow.com/questions/10378855/java-io-invalidclassexception-local-class-incompatible

Next Steps:
1. Will finish implementing and fixing bugs for trading and temporary transactions via implementing a system clock.
2. Complete view 3 most recent transactions and most frequent trading partners functionality.
3. Will finish implementing the threshold value by using a hashmap provided in the threshold class.
4. will integrate other functionalities and bugs. potentially change the .ser persistent file system to a more extensible file system such as using Gson package to serialize key-value pairs to json object.