Final Project Documentation

Name: Anuj Jain
EID: aaj2447

Optional Features:

- Set a minimum starting price greater than 0
- Set a high limit or buy now price for each item
- Every customer is able to see the bid history of each item such as the bidder's username and their bid
- Items have descriptions available for users to view
- Users can search through items in the auction
- Background music when auction GUI is open
- Nice GUI
- Using the Observer class and Observer interface
- Salting and hashing of passwords using SHA-256, encryption, and decryption with RSA
- JSON database for items and users

User Guide:

Open up the client jar and a login window will pop up. You can enter the auction as a guest user, or create an account if you would like. If you already have an account, enter your username and password and log in. After you log in, the window will be closed, and the auction window will open up. On the left, you will see a table of all the items that are at the auction that you can bid on. Click on an item that you would like to bid on. You can also use the search field at the bottom to find the item you are looking for. Once you have selected an item, you can see the full information in the middle pane. To pull up the history of all bids on the item, click the "Get History" button on the right side of the window. This will update the list of bids on the right. To place your bid, enter a bid amount in the bid field and click bid. You will see if your bid was successful as the information in the table on the left will update automatically as well as the middle pane. You will see if a new bid comes in as the table on the left will automatically update and notify you of new bids and item status.

Programmer's Perspective:

Inside my Server class which extends Observable, I create a new instance of a server first. I then populate the lists of auction items and users with the JSON files that I have, users.json and items.json. I do this using Gson to convert the JSON files to the correct User and AuctionItem objects. I then set up networking. In this setup, I connect to port 4243 and create a new

ClentHandler and add the connected client as an observer. I have a ClientHandler class from which I can create new instances. ClientHandler has the methods in which the server can send a message to a client and receive messages from a client.

The way that the client and server are able to communicate is through strings sent across in JSON format. These strings have identifiers that correspond to the attributes in my Message class, so I am able to use Gson to create a more usable message object once the string is received. I have a processRequest method on both Client and Server which processes the type of request received and takes the necessary action.

Once the server is up and running, clients are now able to connect to it. When someone starts up the client, it connects to the server and is added as an observer. I use JavaFX to create the login window that the user first sees in which they can enter as a guest, create a new account, or log in. If a user creates an account, their username and password are sent to the server, and the server salts and hashes the password with SHA-256. I add the new instance of the user to the user list and store the salt and hash so that the password can be checked if the user tries to log in.

I also have encryption and decryption methods in both ClientSide and SeverSide using RSA for messages sent and received.

In general, to make my GUI look nice, I nested GridPanes within each other to really be able to place elements where I want them.

Once the user logs in, the login window closes, and a new window with the auction opens. On the left side, I have a TableView that I populate with items from the server by requesting them through a message. A user is able to search through this table with the search field below it using a FilteredList. I have an ActionListener on the selection of items that causes the middle pane to update with the item, item price, description, highest bidder, etc. In this middle pane, the user can also enter a bid on a selected item.

In the background, I am using MediaPlayer as part of JavaFX to play elevator music when the auction window is open.

References:

https://www.javainterviewpoint.com/java-salted-password-hashing/

https://www.tutorialspoint.com/java_cryptography/java_cryptography_encrypting_data.htm

https://edencoding.com/search-bar-dynamic-filtering/