SCC311 Coursework Part 2 Specification

Due: Friday Week 6 through Moodle
Marking: Your assigned Lab session Week 7
Total mark allocation: 35%

You are asked to implement a distributed auctioning system using Java RMI. This auctioning system should consist of a central auctioning server and two separate client programs.

Level 3: Auction Logic (17%)

Implement server logic to handle creating and managing listings, and bidding on listed items. This auctioning system should consist of an auctioning server and two separate clients:

The **first (seller) client** program should enable the creation of a new auction for an item offered for sale. The seller should be able to provide a starting price, a short description of the item, and a minimum acceptable price (reserve price). Creating an auction will return a unique auction ID (as an integer). Subsequently, the seller should be able to close the auction using the same client program by quoting the auction ID. When the seller closes an auction, the client should indicate who the winner is along with their details (see below), or else indicate that the reserve has not been reached.

The **second (buyer) client** program should enable potential buyers to bid for auctioned items. Firstly, the program should enable the browsing of active auctions with their current highest bid (but not the reserve price, which should remain secret). The client program should then enable a buyer to bid for a selected item, by entering the buyer's details: name and email address.

The **auctioning server** should deal with requests from different instances of the two client programs, and maintain the state of ongoing auctions. Note that we do *not* expect you to use a database system (e.g. MySQL) to store the auction data. Instead, memory java containers (e.g. Lists and Hash Tables) should be used.

Remember to apply principles of objective oriented programming, such as identity separation and encapsulation. We'll be marking on how elegant, clean and efficient your code is, so do your best.

Level 4: Access Control (10%)

The system implementation so far is insecure; it has no mechanism to mitigate the following cases:
- tampering with bids (one buyer modifying or stopping another buyer's bid),
- closing the bidding by a user who did not create the auction,
- accepting a bid that is the same or lower than the current bid,

Implement logic to ensure the right level of access for different users to protect against the above attacks. Depending on how you designed your system, you might need to add functionality to register user accounts.

Level 5: Authenticated Auctioning (8%)

Modify your system to provide cryptographic authentication. In particular, you should use a suitable challenge-response protocol based on either symmetric or asymmetric cryptosystems (be prepared to defend your choice).

Similar to Level 2, password-based authentication is not accepted. You can assume that registered users have their associated keys; e.g. user IDs and pre-generated keys can be loaded from a file.

To make things easier, secure communication *after* authentication is not a requirement for this task. In other words, client-server communication after initial authentication does not need to be encrypted as it was in Level 2.

Marking Scheme

Level 3	
Selling client: create listings with reserve, close listings, announce winner	6 marks
Buying client: browse, bid	4 marks
Management of listing and bidding data on server	4 marks
Code elegance	3 marks
Level 4	
Test case 1	5 marks
Test case 2	5 marks
Level 5	
5-stage challenge-response protocol, both parties verifying each other	max 8 marks
3-stage challenge-response protocol, only one party verifying the other	max 5 marks