Lancaster University Leipzig

LZSCC.311 Coursework Stage 1 Specification

This coursework stage does not need to be submitted, but is test-marked in Week 4 with feedback. It is re-marked for real, along with your more advanced submission, in Week 7.

The goal of this stage is to build a simple client-server setup that invokes a method using Java RMI. You will develop both the client and the server.

Level 1: Invocation (9%)

Build an RMI server that offers this exact interface:

public AuctionItem getSpec (int itemId, int clientId) throws RemoteException;

This function call should return the details of an auctioned item that has the identifier *itemId*. The *clientId* can be assumed to be a username, and is used only in the later versions of the coursework. The details of the return type AuctionItem are for you to define, but below is a suggested structure.

int itemId
String itemTitle
String itemDescription

You may want to extend this structure to add other variables, such as *item condition (new / used)*, etc. – this may especially be relevant in the next stages of the coursework. You must also build a very simple client that invokes the above method on the server using RMI, and displays the return values to the user. *A basic command-line client is sufficient,* but must allow the user to enter the itemId details (this should not be hard-coded).

For your server, at this stage, it is sufficient to use a hard-coded set of auction items.

Tip: Do not forget to start the RMI registry when running your code.

Level 2: Basic Auction Logic (8%)

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 to 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 you should use in-memory Java data structures (e.g. Lists and Hash Tables) to store all auction data (using a persistent storage solution will not gain additional marks, and will make the following coursework stage more difficult).

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 consider your design carefully.

Mark Scheme

Level 1

- Client invocation and display — 3 marks
- Server interface — 3 marks
- Answer in-lab questions — 3 marks

Level 2

- Selling client: create listings with reserve, close listings, announce winner — 3 marks
- Buying client: browse, bid — 3 marks
- Management of listing and bidding data on server — 2 marks