Distributed Systems



# DT2284 Distributed Systems Assignment

## Maciej Piotrowski

## C13470112

## 17th of November 2016

# Introduction

This application allows for multiple clients to connect to server and bid for items which are currently placed on the auction. This application is implemented using Java sockets and Java multi-threading. It is a simulation of online auction. I’ve modified chat server given to me by the lecturer to implement this assignment.

***Client Specification***

- Connects to the server. The item currently being offered for sale and the current bid or a (or reserve price) are displayed.

- Enter the bid. The amount entered should be greater than the current highest bid.

- After a new bid is placed, the amount of the new bid must be displayed on the client’s window/console.

***Server Specification***

- Receive connections from multiple clients.

- After a client connects, notify the client which item is currently on sale and the highest bid (or reserve price).

- Specify the bid period. Max allowed 1 minute. When a new bid is raised, the bid period is reset back.

- When a new bid is placed, all clients are notified immediately. Clients should be notified about the time left for bidding (when appropriate).

- If the bid period elapses without a new bid, then the auction for this item closes. The successful bidder (if any) is chosen and all clients are notified.

- When an auction for one item finishes, another item auctioning should start. Minimum of 5 items should be auctioned, one after another. Only one item at a time.

- Any item not sold should be auctioned again (automatically).

***Extra Functionality***

-Sound during bidding process

-Informing auctioneers when last item on the auction is on the auction

-No one can bid anymore when auction is over

-Auction starts when there are two auctioneers connected to the server

To start the server command ***java -classpath . AuctionServer port*** must be entered or execute the Server.bat file (***java -classpath . AuctionServer 9898***).

To start the client command ***java -classpath . localhost port name*** must be entered or execute the ClientDamian.bat (***start java -classpath . Client localhost 9898 Damian***) and ClientMaciek.bat file (***start java -classpath . Client localhost 9898 Maciek***).

# Design Documentation

**AuctionServer**

The entire auction application functionality is implemented in this Java class. The server is binding a socket to the port number chosen and entered by the user. Server then calls the start function which adds a thread to the server. It also adds items to the array list of objects by calling addAuctionItems() function and adds items to the auction by getting the 1st object from array list and assigning the values to the local variables (itemNmae, itemPrice)(addItemsToAuction() function). When server is running, it waits and listen for the connection from the clients. The welcome(int ID) function displays appropriate message after clients connects to the server and it’s called from addThread(Socket socket) function which passed the ID of the client which just connected to the server. When 2 clients are connected to the server and there are items still for auction runTimer() function is called where the majority of auction logic is implemented and start\_auction is set to 1 so server can accept bids from the clients. If there are no auction items left message is send to the client who connected after the auction is over. When user enter bid the input is checked if its equals to “.bye” then client is removed from the server. When user enters bid the string message from the client is decoded into integer and compared with item price. If the bid entered by user is the same as a current bid or lower, then message is send to the client informing him about the situation. Otherwise item price will be changed into client bid. The client will be alerted that he placed the highest bid for the item and other users will be informed that the price of an item has changed. The timer is then cancelled and restarted again for another minute. The runTimer() function monitors the auction times. If bid was placed, then depending on the item quantity auctioneers receive appropriate message from the server. Users will get reminder after 30 seconds with one beep sound and message and after 15 seconds with two beep sounds and the message. When user wins the auction 3 beep sounds will be played and message will be send to the user and other users. Item will be removed from the auction and new one will be placed for the next biding. If no one bids for the item, then the item is removed from the array list and its added again at the end of the array. Users will still get notification after 30 and 15 seconds but without the sound.

**Client**

This class is responsible for handling user interaction with the server. Client connects to the server on the localhost, port the same to match the server and the user name. Depending on the time when client connects with the server, welcome message is send from the server to the client. The user can enter a bit for the item. If the bid is not an integer he will get message asking him to input valid bid with one exception. If user enters .bye into command window then it calls the stop function in the client class. It disconnects the client from the server closing user input and output stream. Every time user enters the bid the time will reset to one minute.

**ClientThread**

This class handles threads created for each client when they connect to the server. It allows for the threads to be set up and handle the closing and stopping of the threads. The client thread is mainly used for listening for incoming messages send from the server on the shared port and closing or getting the input stream.

**AuctionServerThread**

This class handles thread created for the server when server is started. The server creates output and input stream which is responsible for data going in and out from the server to the client.

**Items**

This class is used for adding items objects into array of objects which are used to be auctioned later. It contains getters and setters for the item name and the price.

# Declaration

I declare that this work, which is submitted as part of my coursework, is entirely my own, except where clearly and explicitly stated.