

Distributed Systems Assignment 1

By James Byrne C11393906 - 19/11/2015

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

# Instructions

Compiling the application

* Navigate to auction-assign
* Execute the compile.sh or compile.bat file depending on your platform
  + E.g. “sh compile.sh” on a unix system
* The application should be fully compiled

Instructions for the server

There are two ways of starting the application

* Navigate to auction-assign
* Execute the startServer.sh or startServer.bat file
* OR
* Navigate to auction-assign/src/server
* Execute the following command
  + java AuctionServer
* The server will now run and give updates on the status of its clients

Instructions for the client

* Navigate to auction-assign
* Execute the startCliet.sh or startClient.bat file
* Navigate to auction-assign/src/client
* Execute the following command
  + java AuctionClient
* Follow the instructions given through the terminal

# Application Design

## AuctionServer.java

The AuctionServer class receives a connection request from a client which is then accepted. It will then spawn an instance of the ClientHandler class and ItemHandler classes. It will then run the ClientHandler passing both the ServerSocket (client) and the instance of ItemHandler. Every client will be passed the same instance of ItemHandler.

Functions :

main() :

* Take a client connection
* Create a new instance of ItemHandler
* Spawn a ClientHandler and pass a reference to the ItemHandler

## ClientHandler.java

The ClientHandler class manages all of the clients interactions with the server. It implements the runnable class and runs as its own thread.

Functions :

run()

* Establish input and output streams
* Give the client the current item and bid
* Wait for user input
* If user inputs “get bid” return the current bid
* If user inputs a number
  + If the users number is greater than the current bid accept bid and set new current bid
  + If the users number is less than the current bid return the appropriate error message
* Wait until the “QUIT” command is received from the user

## ItemHandler.java

The ItemHandler class manages the items that are up for auction and their properties. Every client uses the same instance of ItemHandler. This keeps every client up to date on the same object.

Functions :

ItemHandler() : constructor

bid(String newBid):

* Takes in the request for a bid

getCurrentItem() :

* Returns the item currently up for auction

getCurrentBid() :

* Returns the current bid on the item currently for sale

## ItemFactory.java

The ItemFactory creates the data structure which the ItemHandler works upon. This is, an ArrayList containing HashMaps. The HashMaps represent each “item” and its properties, name and bid.

Functions :

ItemFactory() :

* Default constructor

newAuction() :

* Returns an ArrayList of HashMaps containing all of the items to be auctioned and their properties

## AuctionClient.java

The AuctionClient class starts the client connection and spawns a thread (CleintHandlerThread) to deal with user input and output.

Functions :

main () :

* Establish a connection with the server
* Spawn an instance of the ClientHandlerThread class passing the host and PORT number

## ClientThreadHandler.java

The ClientThreadHandler class manages the users interactions with the server.

Functions :

ClientThreadhandler() :

* Default constructor takes in the host and PORT number

run() :

* Open input and output streams
* Give the user instructions on how to interact with the server
* Take in user input and send it to the server
* Take in the servers response