***DITRIBUTED SYSTEMS***

***ASSIGNMENT***

22nd of November 2018

*PAUL PETRISOR*

*C15735611*

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

*Running the Auction :*

To run the server:

* java -classpath . AuctionServer 1234

To run the clients:

* java -classpath . AuctionClient localhost 1234 paul (or any other username)

Running using .bat files :

I have created 1 server.bat fiel and 3 .bat files for clients.

* For Mac
  + First run the server using *– bash server.bat*
  + Run the clients using – *bash client1 , bash client2 …*
* For windows
  + Type the file name and press enter

*The system*

The application was built starting from the chat application example provided in lab 4 of this course.

The architecture used was Java sockets and Java multi-threading. The server will run and will allow multiple clients to connect to the auction and place bids. The server will be able to send same message to all the client or send different messages depending on clients request (Fig. 1).

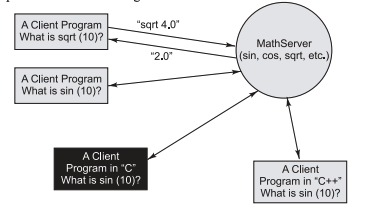


Fig. 1 A socket based math server and clients

The *AuctionServer* class implements the Runnable interface.

Inside this class we provide implementation for the abstract method run() where we specify the thread’s operations.

The *AuctionClient* class creates an instance of the new class called the Runnable object. In the start() method the client constructs a new Thread object and invokes the start() method. The Java Runtime Environment calls back to the run() method of the Thread() class where is specified the execution flow of the thread. When run() method is completed the thread is terminated (Fig. 2).

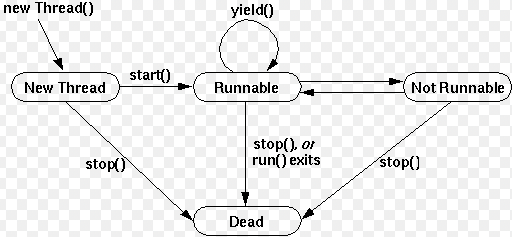


Fig. 2

A new class *Item* has been created to help managing the items to be added to the auction. An item will have a name and a starting price. The price will be updated when the auction is running through the setNewPrice() setter method.

Java *Timer* and *TimerTask* are java util classes and have been used to schedule tasks in a background thread.

TimerTask class is used to define a task which can be run once or for many number of times. The Timer represents the scheduler.

*How the system works?*

The server is running and waiting for clients to connect on the specified port number provided.

The clients will connect by entering the host address, port number and their name.

When the first client connects the server will prompt a message saying he is the first to join and we’ll have to wait for at least another client for the auction to begin. When the second client connects the auction can begin.

The server selects the first item from an ArrayLists of 6 items and it is displayed the item name and starting price. The bidding can now begin.

A timer has been used for this auction to limit the waiting time to 60 seconds if no input is received back from the clients.

If a bid was entered the timer will reset back to 60 seconds and the client entering the bid is notified by a message that he’s the highest bidder while the other clients will be notified about the new price. If within the next minute no higher bid is entered the client will be notified he won the item while the other clients will be notified that the item was sold and for what price.

The next item from array will be selected, if any, and displayed up for sale.

If no bid is received for an item within the 60 seconds the item will be added to the end of array and relisted.

If no more items are left for current auction all the client will be notified and the auction will end.