

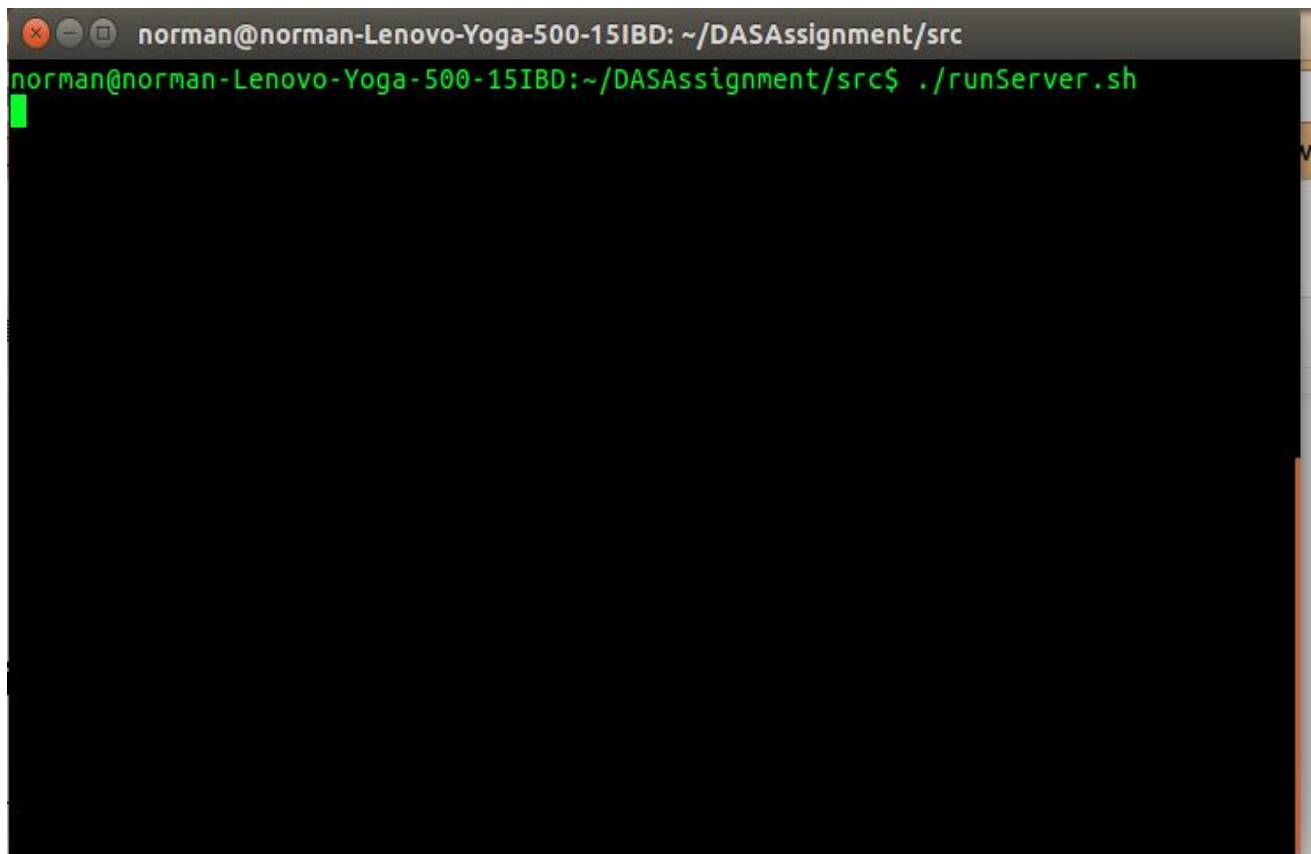
DAS4 Assessed Exercise
2228147T
Norman Tan Jian Liang

1 Overall Design

1.1 Running the Server

Usage:

- Open a Terminal
- Navigate to the src/ folder of the program
- ./runServer.sh or windows run the runServer.bat

A screenshot of a Linux terminal window. The title bar at the top shows standard window controls (close, minimize, maximize) and the text 'norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src'. The terminal content shows the prompt 'norman@norman-Lenovo-Yoga-500-15IBD:~/DASAssignment/src\$' followed by the command './runServer.sh' entered in green text. A green cursor is visible at the end of the command line. The rest of the terminal area is black and empty.

```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
norman@norman-Lenovo-Yoga-500-15IBD:~/DASAssignment/src$ ./runServer.sh
```

1.2 Client UI

Usage:

- Open a Terminal
- Navigate to the src/ folder of the program
- ./runClient.sh or windows run the runClient.bat
- Enter Email

```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
norman@norman-Lenovo-Yoga-500-15IBD:~/DASAssignment/src$ ./runClient.sh
Please Enter your email: Norman@gmail.com
```

```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
norman@norman-Lenovo-Yoga-500-15IBD:~/DASAssignment/src$ ./runClient.sh
Please Enter your email: Norman@gmail.com
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: 
```

1.3 Creating an Auction

Usage:

- Choose the option 1 for Creating an Auction
- Input the name, value of item and closing time in seconds
- Auction creation complete

```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
norman@norman-Lenovo-Yoga-500-15IBD:~/DASAssignment/src$ ./runClient.sh
Please Enter your email: Norman@gmail.com
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: 1
Please Enter name for New Auction Item: Apple
Please Enter value for New Auction Item: 2.99
Please Enter Closing time/date in seconds: 300
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: █
```

1.4 Bidding in an Auction

Usage:

- Enter option 2 to bid for an item (Recommended to use option 3 before entering option 2 to know which auction to bid)
- Enter the item id to bid
- Enter the value to bid

```
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: 2
Item to bid: 2
Current value is: 2.99
Please bid higher then the current bid: 3.99
3.99
You are now the highest bidder
Press Enter key to continue
```

1.5 Displaying Current Auctions Available

Usage:

- Enter option 3 to view all Current Auctions

```
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: 3

Id: 2
Name: Apple
Value: 3.99
Current highest bidder: Norman@gmail.com
Closing time: 13:26

Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: █
```

1.6 Displaying all Auctions

Usage:

- Enter option 4 to view all Auctions happening now and happened in the past

```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
4) List All past Auctions
5) Exit
Input Choice: 4

Id: 1
Name: Bottle
Value: 7.6
Current highest bidder:
Closing time: 11/24/2016 12:35

Id: 2
Name: Apple
Value: 3.99
Current highest bidder: Norman@gmail.com
Closing time: 11/24/2016 13:26

Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: █
```

1.7 Callbacks

Usage:

- Wait until the item or auction is closed after bidding for an item or creating an auction

1) Bidder won bid and Owner is notified

```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
5) Exit
Input Choice: 1
Please Enter name for New Auction Item: Apple
Please Enter value for New Auction Item: 2.99
Please Enter Closing time/date in seconds: 30
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice:
*****Notification*****
Your item bid: 3
Have been completed with the value of: 3.99
Winner: Norman@gmail.com

Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: █
```

Owner Call back

```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
Please bid higher then the current bid: 3.99
3.99
You are now the highest bidder
Press Enter key to continue

Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice:
*****Congratulations!*****
You have won the bid for item: 3
With the value of: 3.99
Owner: NormanTJL91@gmail.com

Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: █
```

Winner Callback

2) Nobody bid for item

```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
Input Choice: 1
Please Enter name for New Auction Item: Pear
Please Enter value for New Auction Item: 4.9
Please Enter Closing time/date in seconds: 30
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice:
*****Notification*****
Your item bid: 4
Have been completed
No one bid for your item
Please Try Again

Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: █
```

Owner Callback if nobody bid for the item

3) Losing a bid to another client

```
*****Bid lost*****
Your bid for item: 5
item name: Orange
item closing time:13:36has been lost
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice: █
```

Callback to losing client to notify that bid is losing

1.8 Pinging server and client

Usage:

- Every five seconds client will ping the server to check if the server is still “alive” if the server is not “alive” the program will state that the server is not online and will exit the client program

```
Choose option
1) Create Auction Item
2) Bid Item
3) List Auction Items
4) List All past Auctions
5) Exit
Input Choice:
Server is not online!
Please restart the program
norman@norman-Lenovo-Yoga-500-15IBD:~/DASAssignment/src$
```

2 Implemented specifications

2.1 general design

The general design is that, a client has to communicate to the server through an server interface, after that the server that has received the request will order the servant to complete a task and return the results to the client. The server interface is a list of methods the client is able to utilise in the server, although initially the general idea is only one interface, server interface, the following task requires a second interface which is callback functions to allow the server to callback to clients. By requiring the server to callback to clients it is required of the clients to be part server part client.

2.2 Client

Client is able to create auction items specifying name, item value and closing date/time specified in seconds, Section 1.3 shows that this specification has been met. Method for listing available auctions is also completed and shown in section 1.5, meeting the specification. By the end of the auction the owner as well as the bidder of the item should be notified of the result, is implemented in section 1.7.

The client is able to disconnect and reconnect to the server(if the server has not been restarted or turned off) using the same email address but it is provided if there is only one person using that email address.

Creating auction items creates an object of auction item and pass it to the server which is then saved into a hashmap of objects which every client can retrieve and view.

The client is able to specify the port number and ip address to connect to the server but the only limitation is that some ports might be used by other background programs that are not known to the user.

2.3 Server

Auctions create by Clients will be used and saved into a csv file every time a method is called and completed this saves the state of the ongoing auctions. Past auctions are retrievable from the section 1.6. There is also a ping function for clients to check if the server is still alive, when that happens the client program is stopped intentionally and will request the user to restart the system.

When a client starts running it registers itself with its client interface to the server, the server then keeps a hashmap of Client with keys as their email and value as their client interface object. This allows the server to

callback to clients when required for example when a bid ends, or they lost a bid but auction is not over yet, or nobody bid on their auction and it ends. The server will notify the owner of the bid and winner of the bid and provide the email address of the owner to the winner and email address of the winner to the own to allow further communication between the two parties. The server will also notify the owner of the auction if nobody has bid on the item and the auction has ended. The server will also notify the client whose bid has been lost but has a chance of raising the bid.

When the server's method has been invoked it saves its state into a csv file. Every time the server is restarted it will load the state saved, but is it unable load all the client's interface for the current auctions. It is able to retrieve the list of all past auctions.

The server has a java method that check if items in the list is expired and save the current available auctions into another hashmap of items.

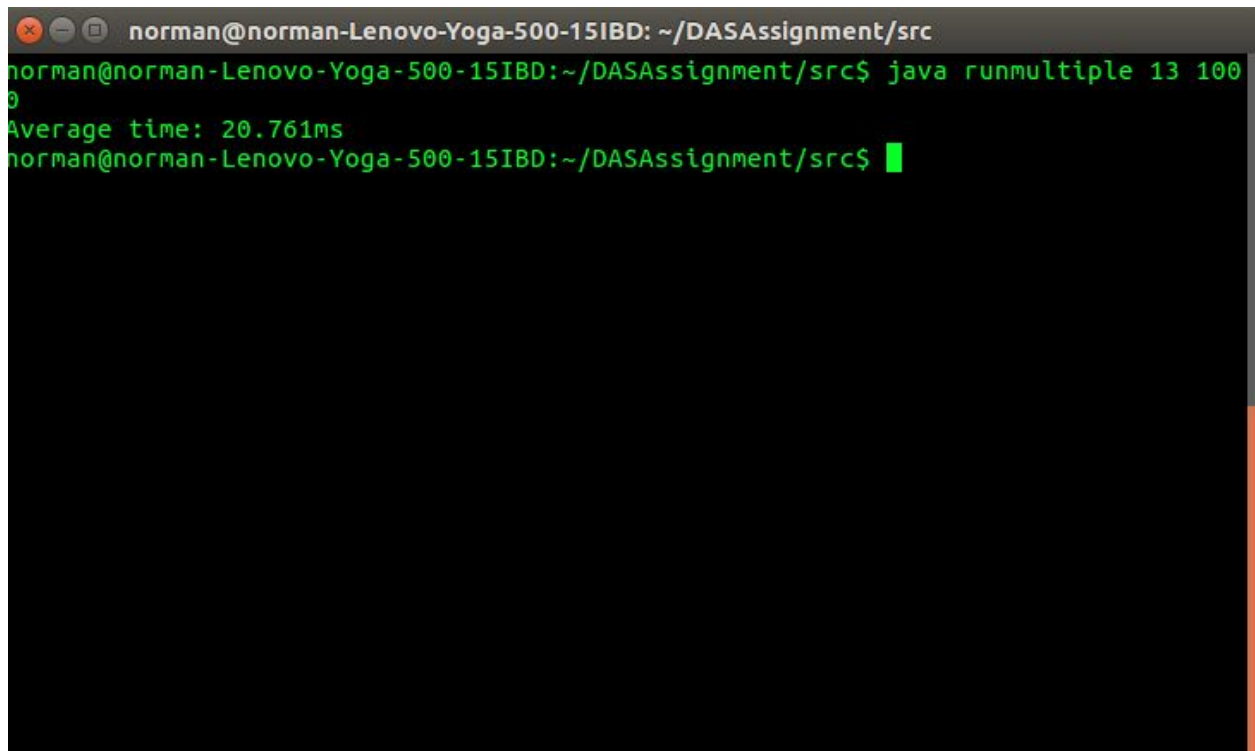
Together with this report is the java files of the project, there will also be a csv file for the saved state and a separate java file namely runmultiple.java to do the rmi call multiple times used to get the results for the performance portion of this report. Usage of the runmultiple.java is java runmultiple <item id to bid> <number of threads to bid>

3 Performance

The test will temporarily stop the callback function as the threads are calling the bid method, the threads did not register as a client therefore on the server terminal there will be nullpointer exceptions

Usage

- Open a terminal
- Navigate to src/
- Java runmultiple <item id> <number of threads to run>



```
norman@norman-Lenovo-Yoga-500-15IBD: ~/DASAssignment/src
norman@norman-Lenovo-Yoga-500-15IBD:~/DASAssignment/src$ java runmultiple 13 100
0
Average time: 20.761ms
norman@norman-Lenovo-Yoga-500-15IBD:~/DASAssignment/src$
```

Time taken is computed in average of per bidding an item

Number of calls	1st Method call Duration (ms)	2nd Method call Duration (ms)	3rd Method call Duration
1	1.0ms	2.0ms	1.0ms
10	2.9	1.9ms	1.9ms
100	11.51ms	18.36ms	8.91ms
1000	58.67ms	14.414ms	15.914ms
10000	91.3582ms	11.5093ms	20.492ms

4 Other improvements

Other improvements might include saving of client interfaces on the server side so when the server is down it can save and load the client interfaces and perform callbacks even after the server is down. Saving everything on a database is also an option and provide an automated service of sending a notification to the clients so that when the clients are not logged on the server they are still able to receive information about the bids they were interested in.

Other viable improvements include adding the view function to include expired items for one day or two days. It is also viable to allow only unique email at one time disallowing others to use the same email as the original user. The server can also be modified that can take in parameters when running the program to receive port numbers to allow users to enter through a certain port number.