DT211C BSc. (Honours) Degree in Computer Science (Infrastructure)

DT228 BSc. (Honours) Degree in Computer Science

DT282 BSc. (Honours) Degree in Computer Science (International)

Distributed Systems [CMPU4021]

Assignment

Published date: Thursday 19 October 2017

Due Date: Thursday 16 of November 2016, 23:59

This assignment represents the practical assessment for this subject and is worth 24% of the overall mark.

For this assignment you are required to design and implement a distributed application that is a simulator of an **on-line auction system**.

There will be **multiple clients and a server**. The server will offer items for sale. The client module will allow the user to bid for items. An item is sold to the highest bidder. Your code will enable users only to bid for items – it will not conduct actual credit card transactions.

Deliverables: Source code, Design Document, and Demo.

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).

Implementation

The assignment **must** be implemented using **Java *sockets***. Basic Java JDK with minimal class path should be used. It should work with version 7 or 8. No additional libraries/packages should be used. If data storage is necessary, a **file** should be used.

Submission

You must provide documentation which describes your architecture and implementation. The documentation should **not exceed 3 pages**, and must include the following:

- Your **name** and **student number**.
- The subject title, i.e. **Distributed Systems**
- The assignment title i.e. **Assignment**
- The date, i.e. **17th of November 2016**
- The following declaration**: "I declare that this work, which is submitted as part of my coursework, is entirely my own, except where clearly and explicitly stated."**
- A clear statement (provide the command line example) of how to start the server and clients.

If your documentation does not include all these details, it will not be marked, and you will be considered as having failed to submit the assessment.

All the code (.java, .class and .bat files) and documentation must be submitted using **webcourses** – in zipped format.

Your code **must** include the **server.bat** and **client.bat** batch files. They will specify (contain) the command line for starting the server and clients respectively. All code must be compiled and ready to run, when `server.bat` and `client.bat` are executed.

Example of the server.bat file:

```
rem Usage: java AuctionServer port
java -classpath . AuctionServer 1234
```

Example of the client.bat file:

```
rem Usage: java AuctionClient host port name
java -classpath . AuctionClient localhost 1234 john
```

Note: 'rem' – comments;

Assessment Criteria

Failure to provide code, documentation or demo will result in a mark of 0%.
You will be penalised an absolute value of 10% for every day that your assignment is late.

The program **must** be demo-ed in the labs. The demos will be scheduled.

Marking Scheme

1. Design document (10%).
2. Functionality implementation (55%).
3. General coding (API, error handling, comments) (15%).
4. Quality of the outcome (10%).
5. Setup/Demo (10%).