CMPU4021 Distributed Systems

Assignment

Publishing date: 28th October 2021
Due Date: 25th of November 2021, 1pm

This assignment represents the practical assessment for this subject, 30% of the overall mark.

For this assignment you are required to design and implement a distributed application that is a simulator of an electronic **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 purchase transactions.

Deliverables: Source code, Design Document

Client Requirements

- The clients should be able to use the all services offered by the auction server.

Clients should:

- 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 Requirements

The server should offer the following services:

- Join auction
- Leave auction
- Bid on an item.
- List auction items

The server should:

- Receive connections from multiple clients, i.e. support concurrent multiple clients connections.
- 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.
- Any item not sold should be auctioned again (automatically).

Extra functionalities

The server should also be able to:

- Allow including new items for sale in the auction server while the auction is running.
- Register clients: unique name will suffice.

Implementation

The assignment **must** be implemented using **Java *sockets***. Java multithreading, locks and monitors should be used where necessary. Basic Java JDK should be used. It should work with **Java version 8**. No additional libraries/packages should be used. If data storage is necessary, **files** should be used.

Submission

You must provide a design document which describes:

- your application architecture,
- setup
- detailed instructions how to run it.

The document should **not exceed 3 pages**, and must also 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. **25th of November 2021**
- 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."**

What should you submit?

1. The design document. It MUST be submitted as **ONE** word/pdf document.
2. **Code** (.java, .class, .bat files and any other files).
3. Put all the code and the design document in a directory named **YOURNAME (i.e. John Smith), ZIP it** and upload it in Brightspace by 11:00am, **25th of November 2021.**

Your code **must** include setup and run instructions and 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 pre-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
```

Note1: 'rem' – comments;

Assessment Criteria

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

Marking Scheme

- Requirements implementation (50%)
- Extra functionalities (10%)
- General coding (error handling, comments) (13%)
- Design document (7%)
- Quality of the outcome (10%)
- Setup/Run (10%)