Distributed Systems Assignment

DT228
BSc in Computer Science

Cian Morar

C16460726

School of Computer Science

Technological University, Dublin

21st November 2019

Declaration

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

Setup

There is very little setup involved in running the application, simply compile the java files with

> *javac -classpath . *.java*

Then run the client and server from the command line. The client file takes the command line argument "localhost". Alternatively, simply compile the java files and then run the client.bat and server.bat

Design

This auction service is intended to allow for one server to handle multiple client connections. Concurrency lies at it's heart. This program consists of five unique java files with a total of seven classes. All of which are outlined below.

- **Item.java**
  - The item class extends serializable and contains all the basic fields useful for dealing with an item object. It contains fields for the current bid, the current client who placed that bid and the name of the item. The class also comes with the necessary getter and setter methods for these fields.
- **Auction.java**
  - This class contains all the basic functionality pertaining to the auction, very often it is used by the server. The client mostly interacts with the auction via the client handler.
- **Client.java**
  - This class also implements serializable. The only field needed for validating the client is the username that they choose when connecting to the client. All usernames must be unique and the application will not allow a client to join the auction with a username that already in use. As per the specifications of the project, the client can place a bid on the current item, they are allowed to add new items while the auction is happening, they can view all the items for sale in the auction and finally they can leave the auction. The main method of the Client class performs all these actions. It is where the interaction with the ClientHandler thread occurs.
- **Server.java**
  - The server is where all the clients connect in order to gain access to the auction and it's items. In order to handle multiple clients in concurrency, the Server class consists of a while loop in which the main purpose is to listen and accept all incoming client requests. Once the client is accepted it will create a ClientHandler thread for each one. This enables the server to be able to handle requests from multiple clients at once.
- **ClientHandler.java**
  - This class implements Runnable and each client will interact with the server through this class. This is where the auction timers are stored. However, the auction timer needs to be a global timer i.e. every client should see the same timer for each item

in the auction. Therefore, the timers are all static so they are accessible by all instances of the timer classes.

- The MyTimerTask class is responsible for tracking the bid period of each item in the auction. Once this period expires, this item is removed from the auction and the winner (if any) is announced. Then the timer resets for the next item.
- The CheckTime class is responsible for providing the server with a countdown. Every second, this checks whether or not the main bid period has expired.
-

UML Diagram

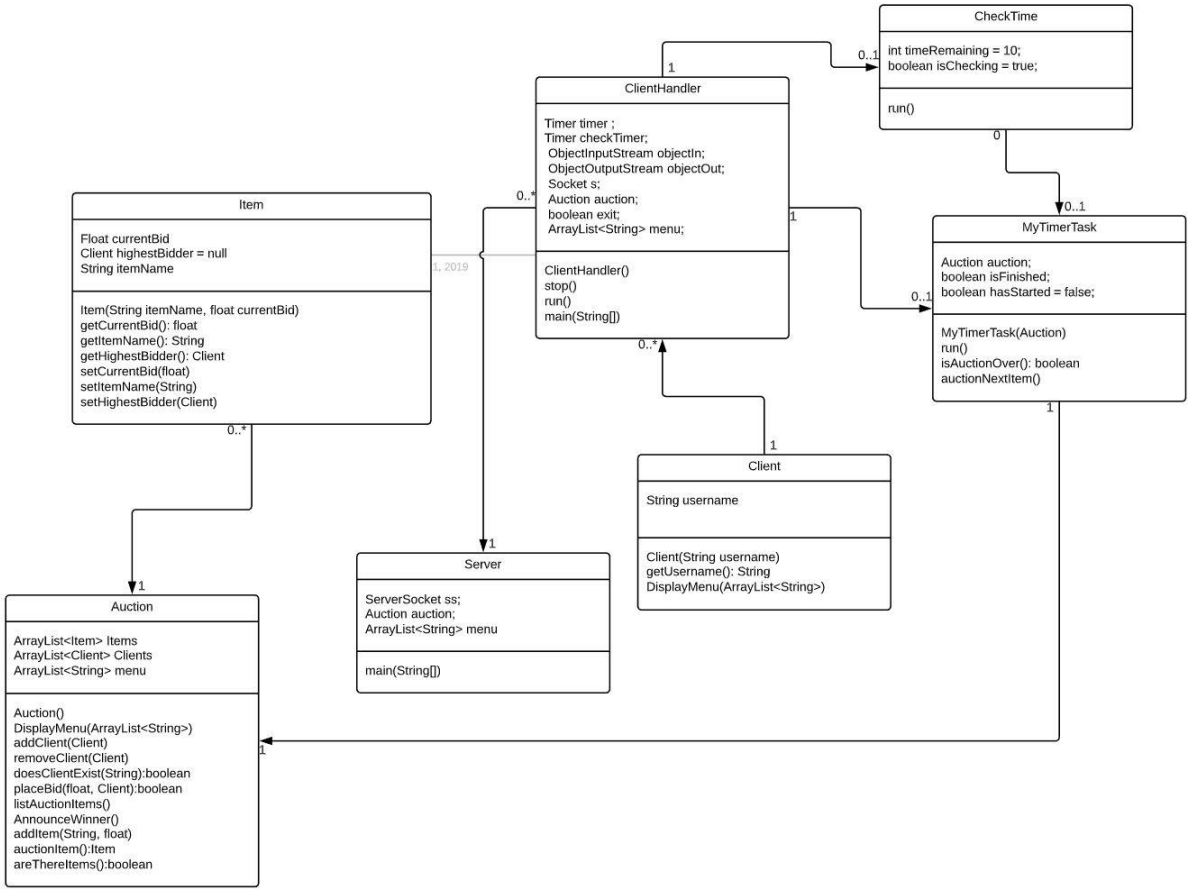

Figure 1 - UML Diagram

Functionality Overview

The server sits and waits for any incoming client requests. When it receives a request from a client, the server creates a Client Handler thread so that it can handle multiple requests from multiple clients at once. Once the client is accepted into the auction (by providing a unique username) they are presented with a menu. The options are as follows:

1. Place a bid
2. Add a new Item
3. View all items for auction
4. Leave auction