Distributed Systems Assignment 2021

Lecturers Name:	Dr Edina Hatunić-Webster
Student Name and Number:	Djibril Coulybaly C18423664
Programme:	DT228 BSc in Computer Science
Module:	Distributed Systems
Submission Date:	29th November 2021

Declaration

Djibril Coulybaly

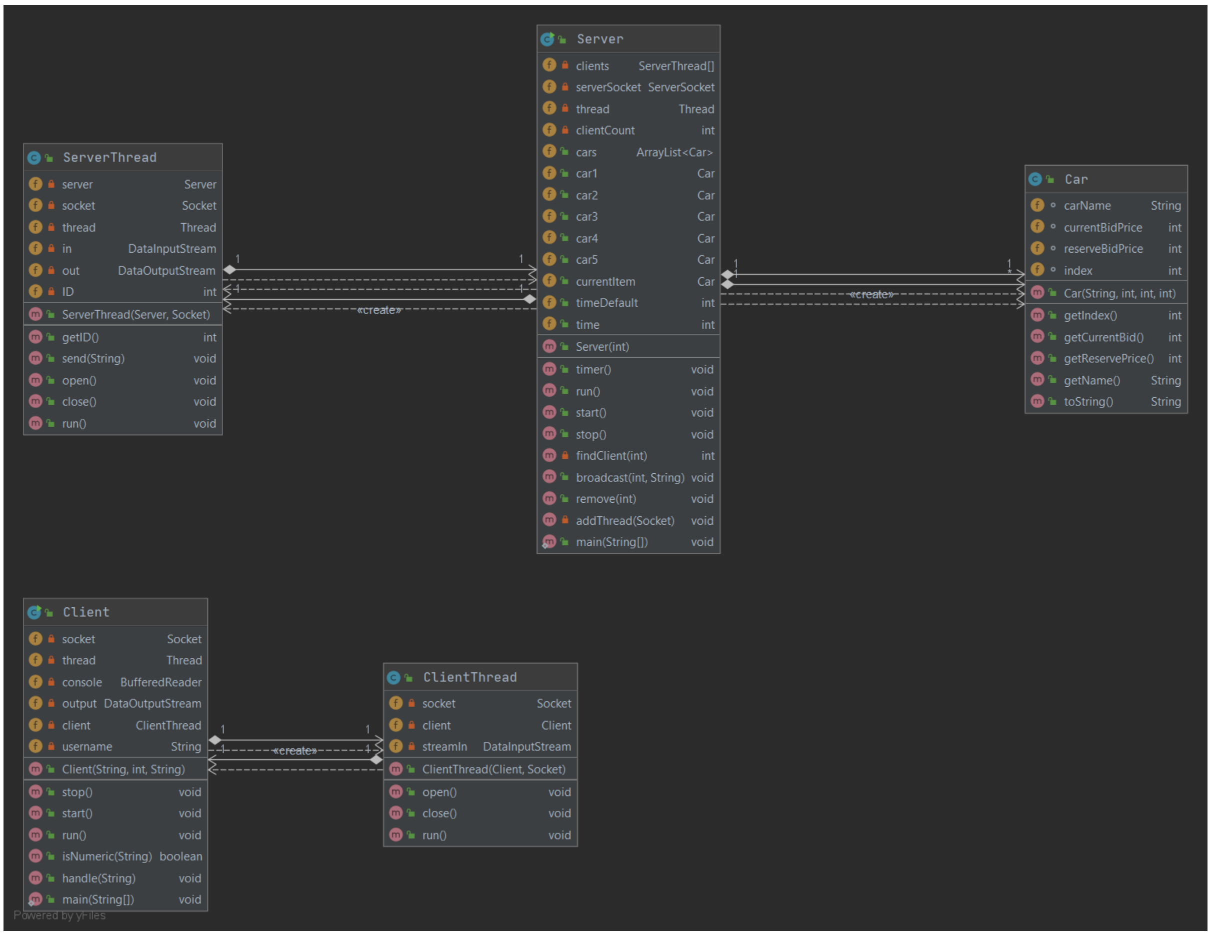
____

29/11/2021

Introduction

This assignment will be focusing on the design and implementation of a distributed application that is a simulator of an electronic auction system using the concepts learned throughout this module. This was programmed in Java using code and notes taken from the labs to structure the application. This application highlights the logic between the client side and server side and their ability to communicate with one another. The auction will allow clients to bid on various cars, while the server will support concurrent clients using the auction and provide the logic behind the auction.

Application Architecture



The client-server architecture model is the foundations of this application. As described by Ian Sommerville, it allows the functionalities of the application to be "organized into services, with each service delivered from a separate server. Clients are users of these services and access servers to make use of them"**(1)**. The model can be split into three parts:

1. The servers **(class Server)** which host the auction and allows multiple clients to use it concurrently **(class ServerThread** and **ClientThread)**
2. The clients **(class Client)** that use the auction to bid on a car **(class Car)**
3. The network on which the client can gain access to the server and in turn the auction

Its important to mention that the client cannot run without the server, hence when executing the application, the server must be running before a client can join the auction.

Setup

As illustrated in the application architecture section, the class diagrams shows the fields, constructor and methods for each class, alongside the relationships between other classes.

On the server side, the class Server creates the necessary threads, sockets and streams that it requires, alongside the connections for the clients to join and partake in the auction. If any messages needs to be relayed to the client, it is implemented by the server. It also provides the list of cars available for auction, the logic to process a bid if it has been sold or not, and a timer to allow the clients to participate in a timely manner.

Furthermore on the client side, the class Client creates the necessary threads, sockets and streams that it requires and allows the client to partake and place a bid. Logic is processed on the client side for each bidding value before submission.

How to run the distributed application?

To run the distributed application, first launch the batch file 'Server.bat' inside the folder named 'Djibril Coulybaly'. This will run the server side of the application and start the auction whilst looking for a client to join. Secondly, to see the interaction between more than one client using the application, 2 client batch files labelled 'Client1.bat' and 'Client2.bat' will create two clients named Djibril and Sarah respectively. This will give them access to the auction and allow them to participate in bids.

Bellow shows the code for each of the batch files

1. Server.bat

```
javac *.java
REM Usage: java Server port
java -classpath . Server 1004
```

2. Client1.bat

```
javac *.java
REM Usage: java Client host port name
java -classpath . Client localhost 1004 Djibril
```

3. Client1.bat

```
javac *.java
REM Usage: java Client host port name
java -classpath . Client localhost 1004 Sarah
```

Bibliography

1. Sommerville I. Software Engineering. 9th ed. Harlow: Pearson Education; 2019.