CMPE436
ConcurrentDistributed Programming
Term Project - Auctioneer

Mehmet Gülşen

December 2017

1 Abstract

This project includes an android application and a java server for making auctions. There can be multiple clients and multiple auctions going on at the same time. The data of the auction items is stored in a MySQL database in the server. Server and android app connected with TCP sockets and server is multi-threaded to handle multiple client requests. To prevent race conditions that can appear as a result of multiple clients accessing the same auction, I have implemented an algorithm taught in class that uses binary semaphores.

2 Introduction

Auctions are considered to be serious events and are usually used for business deals. Most people have not even seen a real auction since they happen very rarely. Online auctions can also carry security risks and therefore are not preferred.

However, auctions do not have to be serious business deals. They can also be used for casual things such as games. Currently, there are not many options for casual online auctions.

This project is an auction platform that aims to be as fast and practical as possible. We do not want the users to worry about security, so we don't ask them for important information, such as credit card information or identifications. Users simply need to pick themselves a nickname and use that nickname to create an auction or bid on one.

The lack of a registration system might be considered a security problem. However, the trade-off is that the users don't need to waste time registering or logging in. All auctions are private, only people who have the auction ID number can see the auction item. This helps minimizing the potential damage of the lack of a registration system. All the people who might harm the auction are the people who the auction item owner give the auction id to, so we are handing over the responsibility of their actions to the auction item owners themselves. Our platform should be only be used for casual interactions, so if the item owners don't trust someone, they simply need to not give them access to their items.

3 Approach

Our system will consist of two parts: the Android application (clients) and the Java Server. The server is not moved to the Amazon, it is working on the local network.

3.1 Server

The server will stay open and it will serve multiple clients. The server will also need to store data using a database.
The project is using a MySQL database to store its data. We use an Apache server to run the MySQL Server on. We have created a database called "auctioneer", and created a table on it called "auctions" to store auction information. We are storing the id, the current bid, minimum increase amount, deadline, auction item owner and the last bid owner.
A new user is created for the Java Server to use to connect to the database and the necessary permissions are granted. The JDBC API which is an interface for java to connect to databases have been used. The server connects to the database through this new user from port 3306 when the service it provides to the client requires a database query.

The server needs to be able to serve multiple clients at the same time. Therefore, a multi-threaded aproach is used to implement the server. TCP protocol is used for the communications between the server and the client.

Clients first arrive at the server's ServerSocket which uses the port number 9090. Then, a new thread called ClientManager is generated with the client socket. ClientManager listens to the client and gives them the appropriate responses. When it finishes responding to the client, it waits for further requests from the same client. If the client terminates the connection, an exception is thrown. This exception is handled and the thread ClientManager terminates.

There are 3 different types of requests the client can make to the server. First, the client says what kind of request it is about to make.

The first type of request is used for getting the information of an auction.

The client will give the auction id. The server makes a query to the server for that id and gets the information of the auction item. If the server can find the item, it will first tell to the client that the request was successful, and give the information of the auction. If the server cannot find the item, it will tell the client that there was an error, and give an error message to the client.

The second type of request is for making a new bid. The client will tell the server the auction id, the last bid made, the new bid, and the name of the bidder. The server first locates the auction item from database and performs some checks. First of all, the item must exist. Second, the client must be up-to-date regarding this item, which the server determines through checking the current bid on the database and the last bid made that the client says. Third, the new bid must be high enough. And at last, the auction must not have expired. If there are any problems with these 4 points, server tells the client that an error has occurred and gives an error message.

If there are no problems with the 4 checks, the server updates the auction item with the new bid and the name of the bidder. It makes another query to get the last values of the item and return them to the client.

The last type of request is for creating a new auction. The client will give us the starting bid, the minimum increase amount, a deadline (in YYYY-MM-DD HH:mm:SS format), and item owner. The server creates the auction using these specifications, and also sets the last bidder's name as the item owner's name. Then, it makes a query to the database for the last created auction, and returns its information to the client.

As it was mentioned before, the server needs to be able to listen to multiple clients and serve them at the same time. However, since all clients are using the same database, this situation causes race conditions. To combat this problem, the Reader-Writer 2 solution from the lecture notes have been implemented. According to this solution, when a reader is in the critical section, it blocks writers but does not block other reader. The writers will only be allowed when there are no readers left. Also, when a writer enters a critical section, it blocks all readers and other writers. This algorithm is implemented using a counter for readers and two binary semaphores.

3.2 Clients

The clients are Android applications. The applications need access to internet in order to connect to the server. When the application is launched, the user will be on Home page.

When the homepage is loaded, the client starts another thread to establish the server connection. In the homepage, the user has 3 options.

If the user presses the "Select Nickname" button, the nickname selection page will be loaded. This page simply has an input field for a name and the submit button to save the user input. There is no communication with the server on this page. The nickname the user selects is stored in the client itself. When the user is done, they can return to homepage via "Go Home" button.

If the user presses the "Create Auction" button, the auction item creation page will be loaded. This page has 3 input fields: the starting bid (number), the minimum increase amount (number), and the deadline (date-time in YYYY-MM-DD HH:mm:SS format). User needs to have picked a nickname beforehand and also fill in all 3 fields in order to create an item. Otherwise, the submit button will simply produce a message to first do them. When the user accomplishes these tasks and presses the submit button, the client will use the already established connection to communicate with the server.The communication is accomplished through another thread. When the server return the auction information of the new item, the auction page for that item will be loaded.

If the user presses the "Find Auction" button, the user will be taken to a page with an number input field and a button. The user needs to type in the id of the auction item they need to find and press the button to go to the auction page for that item.

In the auction page, all of the information about the item is listed. There is a "refresh" button to get the up-to-date information from the server and reload the page. Users will also be making their bids from this page. If they have picked their nicknames, they can type in their bid and press the button to make a bid. The client will make the necessary communication with the server through another thread. After that, it will print a success or error message and reload the page with the up-to-date item information.

4 Conclusion

This is a project that has multiple clients and a server. Therefore, the server had to be implemented in a multithreaded manner. This caused it to become a much more difficult project to finish.

First of all, multithreaded code is more difficult to use than sequential code. Also, it is much more difficult to debug such code. I was only able to overcome this difficulty by adding print statements to everywhere. Also, dangers such as race conditions and deadlocks also had to be taken into account as a result of using multithreaded code.

In order to deal with the race conditions and deadlocks, I have implemented the second solution introduced in the lecture to the readers-writers problem. This part of the project could be improved. In an auction, the order people make bids is extremely important. This project is not very good regarding this issue. Consider this scenario: there are multiple readers in the critical section. Two people make bids and then the readers exit the critical section. Now, one of the bidders will enter the critical section. In an ideal setting, the one who arrived first should have the priority, however, in this project it is pretty random.

Another shortcoming of this project is the Android application part. For example, the date time field of auction creation is simply a string field. However, the program can crash the server if the user input for it is not as it should be. As I am not familiar with Android programming, I had to learn almost everything on the fly. This caused my code to be of lower quality than it should be. My code would be better if I had more experience with Android, and I would have a little more experience if I had started the project earlier.

5 References

https://www.youtube.com/playlist?list=PL6gx4Cwl9DGBsvRxJJOzG4r4k_zLKrnxl
https://developer.android.com/training/keyboard-input/style.html
https://stackoverflow.com/questions/23990165/setting-value-to-textview-in-android
https://stackoverflow.com/questions/2378607/what-permission-do-i-need-to-access-internet-from-an-android-application
https://developer.android.com/samples/DoneBar/AndroidManifest.html
https://stackoverflow.com/questions/13194081/how-to-open-a-second-activity-on-click-of-button-in-android-app

6 Appendix

6.1 Client

Code can be found here : https://github.com/mehmetgulsen/Auctioneer

6.1.1 MainActivity.java

```
package com.example.monster.auctioneer;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
```

```
public class MainActivity extends AppCompatActivity {

    public static final String hostname = "192.168.43.126";
    public static final int port = 9090;

    public static String nickname = "";

    static Socket clientSocket;
    static DataOutputStream outToServer;
    static BufferedReader inFromServer;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button nameButton = (Button) findViewById(R.id.btnName);
        Button findButton = (Button) findViewById(R.id.btnFind);
        Button createButton = (Button) findViewById(R.id.btnCreate);

        //This thread is used for establishing the connection
        Thread connectionThread = new Thread() {
            public void run() {
                try {
                    MainActivity.clientSocket =
                            new Socket(MainActivity.hostname,
                                    MainActivity.port);
                    MainActivity.outToServer =
                            new DataOutputStream(clientSocket.getOutputS
                    MainActivity.inFromServer =
                            new BufferedReader(new
                                    InputStreamReader(clientSocket.getIn
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        };


        //Create connection each time the user goes to the home page
        MainActivity.clientSocket = null;
        MainActivity.outToServer = null;
        MainActivity.inFromServer = null;

        connectionThread.start();
        try {
            connectionThread.join();
            Log.i("Main", "Connection success");
        } catch (InterruptedException e) {
            e.printStackTrace();
        }


        nameButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent i = new Intent(getApplicationContext(),
                        NicknameActivity.class);
                startActivity(i);
            }
        });

        findButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(getApplicationContext(),
                        GetIdActivity.class);
                startActivity(i);

            }

        });
```

```
        createButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(getApplicationContext(),
                        CreateActivity.class);
                startActivity(i);

            }
        });

    }

}
```

6.1.2 GetIdActivity.java

```
package com.example.monster.auctioneer;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;

public class GetIdActivity extends AppCompatActivity {

    static String auction_id;

    EditText idInput;
    Button submit, home;

    static boolean error;
    static String errorMessage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_get_id);

        //Input field for Auction ID
        idInput = (EditText) findViewById(R.id.idInput);

        //"Go Home" button
        home = (Button) findViewById(R.id.btnIdToHome);
        home.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(getApplicationContext(),
                        MainActivity.class);
                startActivity(i);
            }
        });

        //Find auction button
        submit = (Button) findViewById(R.id.btnSubmitId);
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                GetIdActivity.auction_id =
                        idInput.getText().toString();
//Read the input
                if(auction_id.equals("")){
                    Toast.makeText(GetIdActivity.this,
                            "Please fill in the input field.",
                            Toast.LENGTH_SHORT).show();
                    return;
                }

                //Communication thread for finding auction item
            Thread t = new Thread() {
            public void run() {
                try {
                    //"0" means that we are making a query for auction i
                    MainActivity.outToServer.writeBytes("0\n");
                    //The id of the auction we are looking for
                    MainActivity.outToServer.writeBytes(GetIdActivity.a

                    //Get response status. "1" means there is an error"
                    String response = MainActivity.inFromServer.readLine
                    if (response.equals("1")) {
                       GetIdActivity.errorMessage = MainActivity.inFrom
                       GetIdActivity.error = true;
                    } else {
                     //Read auction information
                      AuctionActivity.auction_id = MainActivity.inFromS
                      AuctionActivity.current_bid = MainActivity.inFrom
                      AuctionActivity.minimum_bid = MainActivity.inFrom
                      AuctionActivity.datetime = MainActivity.inFromSer
                      AuctionActivity.auction_owner = MainActivity.inFro
                      AuctionActivity.bid_owner = MainActivity.inFromSe

                      GetIdActivity.error = false;
                    }

                } catch (Exception e) {
                    e.printStackTrace();
                }

                    }
                };

                //Make the communication
                t.start();
                try {
```

```
                    t.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                //Show error message if there is an error
                //Otherwise, go to the auction page.
                if (GetIdActivity.error) {
                    Toast.makeText(GetIdActivity.this,
                            GetIdActivity.errorMessage, Toast.LENGTH_SHO
                } else {
                    Intent i = new Intent(getApplicationContext(),
                            AuctionActivity.class);
                    startActivity(i);
                }


            }
        });
    }
}
```

6.1.3 NicknameActivity.java

```
package com.example.monster.auctioneer;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class NicknameActivity extends AppCompatActivity {

    String name;
    EditText nameInput;

    Button submit, home;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nickname);

        //Input field for typing the nickname
        nameInput = (EditText) findViewById(R.id.nameInput);
         //show the current nickname in the input field
        nameInput.setText(name);


        //Setting the nickname
        submit = (Button) findViewById(R.id.btnSubmitName);
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                name = nameInput.getText().toString();
//reading user input
                MainActivity.nickname = name;
                Toast.makeText(NicknameActivity.this,
                "Nickname is " + name, Toast.LENGTH_SHORT).show();
            }
        });

        //"Go home" button
        home = (Button) findViewById(R.id.btnNameToHome);
        home.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(getApplicationContext(),
                MainActivity.class);
                startActivity(i);
            }
        });
    }
}
```

6.1.4 CreateActivity.java

```
package com.example.monster.auctioneer;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;

public class CreateActivity extends AppCompatActivity {

    static EditText inputStart, inputMinimum, inputDatetime;

    static String starting_bid, minimum_bid, datetime;

    Button home, submit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create);

        inputStart = findViewById(R.id.createStartBid);
        inputMinimum =  findViewById(R.id.createMinimumBid);
        inputDatetime =  findViewById(R.id.createDatetime);
```

```
//"Go Home" button
home = (Button) findViewById(R.id.btnCreateToHome);
home.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent(getApplicationContext(),
        MainActivity.class);
        startActivity(i);
    }
});

//Auction creation button
submit = (Button) findViewById(R.id.btnCreateAuction);
submit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        //User has to have a nickname to create an auction item
        if (MainActivity.nickname == null
        || MainActivity.nickname.equals("")) {
            Toast.makeText(CreateActivity.this,
            "You need to select a nickname",
            Toast.LENGTH_SHORT).show();
            return;
        }

        //Read user input.
        CreateActivity.starting_bid =
            inputStart.getText().toString();
        CreateActivity.minimum_bid =
            inputMinimum.getText().toString();
        CreateActivity.datetime =
            inputDatetime.getText().toString();
        if(datetime.equals("") || starting_bid.equals("") || mi
            Toast.makeText(CreateActivity.this,
            "Please fill in all the input fields.",
            Toast.LENGTH_SHORT).show();
            return;
        }

        //Communication thread for auction item creation
        Thread t = new Thread() {
            public void run() {
                try {
                    //Request type "2" is for auction item creat
                    MainActivity.outToServer.writeBytes("2" + "\

                    //Send auction item specifications
                    MainActivity.outToServer.writeBytes(
                    CreateActivity.starting_bid + "\n");
                    MainActivity.outToServer.writeBytes(
                    CreateActivity.minimum_bid + "\n");
                    MainActivity.outToServer.writeBytes(
                    CreateActivity.datetime + "\n");
                    MainActivity.outToServer.writeBytes(
                    MainActivity.nickname + "\n");

                    //Read the info of the item created.
                    //This is especially for id.
                    String status =
                        MainActivity.inFromServer.readLine();
                    AuctionActivity.auction_id =
                        MainActivity.inFromServer.readLine();
                    AuctionActivity.current_bid =
                        MainActivity.inFromServer.readLine();
                    AuctionActivity.minimum_bid =
                        MainActivity.inFromServer.readLine();
                    AuctionActivity.datetime =
                        MainActivity.inFromServer.readLine();
                    AuctionActivity.auction_owner =
                        MainActivity.inFromServer.readLine();
                    AuctionActivity.bid_owner =
                        MainActivity.inFromServer.readLine();
```

```
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }

            };

            //Make the communication
            t.start();
            try {
                t.join();
            } catch (Exception e) {
                e.printStackTrace();
            }

            //Go to the auction page for the new item.
            Intent i = new Intent(getApplicationContext(),
            AuctionActivity.class);
            startActivity(i);


        }
    });
  }
}
```

6.1.5 AuctionActivity.java

```
package com.example.monster.auctioneer;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;

public class AuctionActivity extends AppCompatActivity {

    public static String auction_id;
    public static String current_bid;
    public static String minimum_bid;
    public static String datetime;
    public static String auction_owner;
    public static String bid_owner;
    static String bid;

    EditText input;

    static boolean error;
    static String errorMessage;

    Button home, submit, refresh;
    TextView auctionIdField, auctionOwnerField,
            minimumField,deadlineField,currentBidField,
            currentBidOwnerField;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_auction);

        //Writing the auction information on the screen
        auctionIdField = (TextView) findViewById(R.id.fieldAuctionId);
        auctionIdField.setText(auction_id);
```

```
auctionOwnerField = (TextView) findViewById(R.id.fieldAuctionOwn
auctionOwnerField.setText(auction_owner);

minimumField = (TextView) findViewById(R.id.fieldMinimum);
minimumField.setText(minimum_bid);

deadlineField = (TextView) findViewById(R.id.fieldDeadline);
deadlineField.setText(datetime);

currentBidField = (TextView) findViewById(R.id.fieldCurrentBid)
currentBidField.setText(current_bid);

currentBidOwnerField = (TextView) findViewById(R.id.fieldBidOwne
currentBidOwnerField.setText(bid_owner);

//User input field for next bid.
input = (EditText) findViewById(R.id.inputBid);

//"Go home" button
home = (Button) findViewById(R.id.btnAuctionToHome);
home.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent(getApplicationContext(),
                MainActivity.class);
        startActivity(i);
    }
});

//Make a new bid button
submit = (Button) findViewById(R.id.btnBidSubmit);
submit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        //User has to have a nickname to make a bid
        if (MainActivity.nickname == null
                || MainActivity.nickname.equals("")) {
```

```
    Toast.makeText(AuctionActivity.this,
            "You need to select a nickname",
            Toast.LENGTH_SHORT).show();
    return;
}

bid = input.getText().toString();   //reading user input
if(bid.equals("")){
    Toast.makeText(AuctionActivity.this,
            "Please fill in the input field.",
            Toast.LENGTH_SHORT).show();
    return;
}

//Communication thread for making a bid
Thread t = new Thread() {
    public void run() {
        try {
            //Request type "1" is for making a bid.
            MainActivity.outToServer.writeBytes(
                    "1" + "\n");
            //Send the necessary auction info and the ne
            MainActivity.outToServer.writeBytes(
                    AuctionActivity.auction_id + "\n");
            MainActivity.outToServer.writeBytes(
                    AuctionActivity.current_bid + "\n")
            MainActivity.outToServer.writeBytes(
                    AuctionActivity.bid + "\n");
            MainActivity.outToServer.writeBytes(
                    MainActivity.nickname + "\n");

            String status =
                    MainActivity.inFromServer.readLine(

            //response type "1" means there is an error
            if (status.equals("1")) {
                //Set the error flag.
                //Make a query to reread auction inform
```

```
                        AuctionActivity.error = true;
                        AuctionActivity.errorMessage =
                                MainActivity.inFromServer.readL

                        MainActivity.outToServer.writeBytes(
                                "0\n");
                        MainActivity.outToServer.writeBytes(
                                auction_id + "\n");
                        status =
                                MainActivity.inFromServer.readL
                    } else {
                        AuctionActivity.error = false;
                    }

                    //Read the updated auction info
                    AuctionActivity.auction_id =
                            MainActivity.inFromServer.readLine(
                    AuctionActivity.current_bid =
                            MainActivity.inFromServer.readLine(
                    AuctionActivity.minimum_bid =
                            MainActivity.inFromServer.readLine(
                    AuctionActivity.datetime =
                            MainActivity.inFromServer.readLine(
                    AuctionActivity.auction_owner =
                            MainActivity.inFromServer.readLine(
                    AuctionActivity.bid_owner =
                            MainActivity.inFromServer.readLine(


                } catch (Exception e) {
                    e.printStackTrace();
                }

            }
        };

        //Make the communication
```

```
            t.start();
            try {
                t.join();
            } catch (Exception e) {
                e.printStackTrace();
            }

            //Print success/error message
            if (error) {
                Toast.makeText(AuctionActivity.this,
                        errorMessage, Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(AuctionActivity.this,
                        "Bid was successful",
                        Toast.LENGTH_SHORT).show();
            }

            //Refresh the page
            auctionIdField.setText(AuctionActivity.auction_id);
            currentBidField.setText(AuctionActivity.current_bid);
            minimumField.setText(AuctionActivity.minimum_bid);
            deadlineField.setText(AuctionActivity.datetime);
            auctionOwnerField.setText(AuctionActivity.auction_owner
            currentBidOwnerField.setText(AuctionActivity.bid_owner)

        }
    });

    //Button for refreshing the page
    refresh = (Button) findViewById(R.id.btnAuctionRefresh);
    refresh.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            //communication thread to read auction information
            Thread t = new Thread() {
                public void run() {
                    try {
```

```
                //request type "0" is for reading auction in
                MainActivity.outToServer.writeBytes(
                        "0" + "\n");
                MainActivity.outToServer.writeBytes(
                        AuctionActivity.auction_id + "\n");
                String status =
                        MainActivity.inFromServer.readLine(

                //response status "1" means there is an erro
                if (status.equals("1")) {
                    AuctionActivity.error = true;
                    AuctionActivity.errorMessage =
                            MainActivity.inFromServer.readL
                } else {
                    //read auction information
                    AuctionActivity.error = false;
                    AuctionActivity.auction_id =
                            MainActivity.inFromServer.readL
                    AuctionActivity.current_bid =
                            MainActivity.inFromServer.readL
                    AuctionActivity.minimum_bid =
                            MainActivity.inFromServer.readL
                    AuctionActivity.datetime =
                            MainActivity.inFromServer.readL
                    AuctionActivity.auction_owner =
                            MainActivity.inFromServer.readL
                    AuctionActivity.bid_owner =
                            MainActivity.inFromServer.readL
                }


            } catch (Exception e) {

            }
        }
    };

    //make the communication
```

```
                t.start();
                try {
                    t.join();
                } catch (Exception e) {
                    e.printStackTrace();
                }

                //print the message if there is an error.
                //otherwise, refresh the page
                if (AuctionActivity.error) {
                    Toast.makeText(AuctionActivity.this,
                            errorMessage, Toast.LENGTH_SHORT).show();
                } else {
                    auctionIdField.setText(AuctionActivity.auction_id);
                    currentBidField.setText(AuctionActivity.current_bid
                    minimumField.setText(AuctionActivity.minimum_bid);
                    deadlineField.setText(AuctionActivity.datetime);
                    auctionOwnerField.setText(AuctionActivity.auction_ov
                    currentBidOwnerField.setText(AuctionActivity.bid_owr

                    Toast.makeText(AuctionActivity.this,
                            "Refreshed",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

6.1.6 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schen
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
 tools:context="com.example.monster.auctioneer.MainActivity">

<Button
     android:id="@+id/btnName"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_marginBottom="8dp"
     android:layout_marginEnd="8dp"
     android:layout_marginStart="8dp"
     android:layout_marginTop="8dp"
     android:text="SELECT NICKNAME"
     app:layout_constraintBottom_toBottomOf="parent"
     app:layout_constraintEnd_toEndOf="parent"
     app:layout_constraintHorizontal_bias="0.502"
     app:layout_constraintStart_toStartOf="parent"
     app:layout_constraintTop_toTopOf="parent"
     app:layout_constraintVertical_bias="0.21" />

<Button
     android:id="@+id/btnFind"
     android:layout_width="122dp"
     android:layout_height="wrap_content"
     android:layout_marginBottom="8dp"
     android:layout_marginEnd="8dp"
     android:layout_marginStart="8dp"
     android:layout_marginTop="8dp"
     android:text="FIND AUCTION"
     app:layout_constraintBottom_toBottomOf="parent"
     app:layout_constraintEnd_toEndOf="parent"
     app:layout_constraintStart_toStartOf="parent"
     app:layout_constraintTop_toTopOf="parent"
     app:layout_constraintVertical_bias="0.458" />

<Button
     android:id="@+id/btnCreate"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_marginBottom="8dp"
```

```
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="CREATE AUCTION"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.697" />

</android.support.constraint.ConstraintLayout>
```

6.1.7 activity_get_id.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schen
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.monster.auctioneer.GetIdActivity">

    <Button
        android:id="@+id/btnIdToHome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Go Home"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.071"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.049" />
    <Button
        android:id="@+id/btnSubmitId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="FIND"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.498" />

    <EditText
        android:id="@+id/idInput"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:ems="10"
        android:hint="Auction ID"
        android:inputType="number"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.503"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.363" />

</android.support.constraint.ConstraintLayout>
```

6.1.8 activity_create.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schem
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.monster.auctioneer.CreateActivity">

    <Button
        android:id="@+id/btnCreateToHome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Go Home"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.092"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.06" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="28dp"
        android:text="Starting Bid"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnCreateToHome" />

    <EditText
        android:id="@+id/createStartBid"
        android:layout_width="187dp"
        android:layout_height="39dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="104dp"
        android:ems="10"
        android:hint="starting bid"
        android:inputType="number"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.833"
        app:layout_constraintStart_toEndOf="@+id/textView2"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="52dp"
        android:text="Minimum increase"
        app:layout_constraintEnd_toStartOf="@+id/createMinimumBid"
        app:layout_constraintHorizontal_bias="0.565"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <EditText
        android:id="@+id/createMinimumBid"
        android:layout_width="185dp"
        android:layout_height="39dp"
        android:layout_marginEnd="24dp"
        android:layout_marginTop="28dp"
        android:ems="10"
        android:hint="minimum"
        android:inputType="number"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/createStartBid" />

    <EditText
```

```
        android:id="@+id/createDatetime"
        android:layout_width="249dp"
        android:layout_height="46dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:hint="YYYY-MM-DD HH:mm:SS"
        android:inputType="datetime"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.86"
        app:layout_constraintStart_toEndOf="@+id/textView5"
        app:layout_constraintTop_toBottomOf="@+id/createMinimumBid" />

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="64dp"
        android:text="Datetime"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView4" />

    <Button
        android:id="@+id/btnCreateAuction"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="CREATE AUCTION"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.809" />
```

```
</android.support.constraint.ConstraintLayout>
```

6.1.9 activity_auction.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schem
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.monster.auctioneer.CreateActivity">

    <Button
        android:id="@+id/btnCreateToHome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Go Home"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.092"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.06" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="28dp"
        android:text="Starting Bid"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnCreateToHome" />
```

```
<EditText
    android:id="@+id/createStartBid"
    android:layout_width="187dp"
    android:layout_height="39dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="104dp"
    android:ems="10"
    android:hint="starting bid"
    android:inputType="number"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.833"
    app:layout_constraintStart_toEndOf="@+id/textView2"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="52dp"
    android:text="Minimum increase"
    app:layout_constraintEnd_toStartOf="@+id/createMinimumBid"
    app:layout_constraintHorizontal_bias="0.565"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView2" />

<EditText
    android:id="@+id/createMinimumBid"
    android:layout_width="185dp"
    android:layout_height="39dp"
    android:layout_marginEnd="24dp"
    android:layout_marginTop="28dp"
    android:ems="10"
    android:hint="minimum"
    android:inputType="number"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/createStartBid" />

    <EditText
        android:id="@+id/createDatetime"
        android:layout_width="249dp"
        android:layout_height="46dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:hint="YYYY-MM-DD HH:mm:SS"
        android:inputType="datetime"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.86"
        app:layout_constraintStart_toEndOf="@+id/textView5"
        app:layout_constraintTop_toBottomOf="@+id/createMinimumBid" />

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="64dp"
        android:text="Datetime"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView4" />

    <Button
        android:id="@+id/btnCreateAuction"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="CREATE AUCTION"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.809" />

</android.support.constraint.ConstraintLayout>
```

6.1.10 activity_nickname.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schem
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.monster.auctioneer.NicknameActivity">

    <EditText
        android:id="@+id/nameInput"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:ems="10"
        android:hint="Enter your name"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.503"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.258" />

    <Button
        android:id="@+id/btnSubmitName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Set Name"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.378" />

    <Button
        android:id="@+id/btnNameToHome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Go Home"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.071"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.049" />
</android.support.constraint.ConstraintLayout>
```

6.2 Server

6.2.1 Main.java

```
//Mehmet G l en
//2013400075
//mehmetgulsen95@hotmail.com
//CMPE436-Term
```

```
import java.net.ServerSocket;
import java.net.Socket;

public class Main {

        public static void main(String argv[]) throws Exception {
                Lock lock = new Lock();


                ServerSocket welcomeSocket = new ServerSocket(9090);

                //Generate a ThreadManager for each client
                //all clients use the same lock
                while (true) {
                        System.out.println("Server is waiting for client
                        Socket connectionSocket = welcomeSocket.accept(
                        System.out.println("Client arrival");

                        new ClientManager(connectionSocket,lock).start(

                }



        }

}
```

6.2.2 ClientManager.java

```
//Mehmet G  l  e n
//2013400075
//mehmetgulsen95@hotmail.com
//CMPE436-Term

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;
import java.sql.ResultSet;
import java.sql.Timestamp;
import java.util.Calendar;

public class ClientManager extends Thread{
        Socket connectionSocket;
        Lock lock;

        public ClientManager(Socket s, Lock l){
                connectionSocket = s;
                lock = l;
        }

        public void run(){

        try {
                //to read data from the client
                BufferedReader inFromClient =
                                new BufferedReader(new
                                InputStreamReader(connectionSocket.getIn

                //to send data to client
                DataOutputStream outToClient= new
                DataOutputStream(connectionSocket.getOutputStream());


                while(true){
                        System.out.println("Waiting for input");

                        //First line of a client request is request type
                        //"0" : Get auction information
                        //"1" : Make a new bid.
                        //"2" : Create a new auction item
                        //If the user closes the connection, this will g
                        //NullPointerException. We will handle it and te
                        String requestType =
                            inFromClient.readLine();
                        System.out.println(
                            "request type is: " + requestType);

                        if(requestType.equals("0")){     //auction info
                                String auction_id =
                                    inFromClient.readLine();
                                System.out.println(
                                    "auction id: "+ auction_id);
                                String query =
                                    "SELECT * FROM auctions"
                                    +" WHERE id = '"+auction_id+"'";

                                //Don't let the writers in while reading
                                lock.startRead();
                                        DBConnect db = new DBConnect();

                                //make the database query
                                ResultSet rs = db.getData(query);

                                //Data is read. No need to lock others o
                                lock.endRead();

                                if(rs.next()){
                                        //information from the database
                                        String current_bid =
                                            rs.getString("current_bid")
                                        String minimum_bid =
                                            rs.getString("minimum_bid")
                                        String datetime =
                                            rs.getString("datetime");
                                        String item_owner =
                                            rs.getString("item_owner");
                                        String bid_owner =
                                            rs.getString("bid_owner");
                                                //Response status "0" m
                                        outToClient.writeBytes("0\n");
                                System.out.println("0");
```

```
                //Sending auction information
                outToClient.writeBytes(auction_i
                System.out.println(auction_id);
                outToClient.writeBytes(current_b
                System.out.println(current_bid)
                outToClient.writeBytes(minimum_
                System.out.println(minimum_bid)
                outToClient.writeBytes(datetime+
                System.out.println(datetime);
                outToClient.writeBytes(item_own
                System.out.println(item_owner);
                outToClient.writeBytes(bid_owner
                System.out.println(bid_owner);

        }else{
                //database couldn't find the ite
                //Response status "1" means erro
                outToClient.writeBytes("1"+"\n"
                outToClient.writeBytes(
                    "Auction does not exist."+"\
        }
}
else if(requestType.equals("1")){        //makin

        //Current datetime of
        //the server in YYYY-MM-DD HH:mm:SS form
        Calendar cal = Calendar.getInstance();
        Timestamp now = new
            Timestamp(cal.getTimeInMillis());

        //reading bid and auction information fr
        String auction_id =
            inFromClient.readLine();
        String last_bid =
            inFromClient.readLine();
        String bet =
            inFromClient.readLine();
        String nickname =
            inFromClient.readLine();

        //Lock everyone out
        lock.startWrite();
        //Query to find the auction item.
        String query =
        "SELECT * FROM auctions WHERE"+
            " id = '"+auction_id+"'";
        DBConnect db = new DBConnect();
        ResultSet rs = db.getData(query);

        //To bid, auction must exist
        if(!(rs.next())){
                System.out.println(
                    "1!Auction does not exist."
                outToClient.writeBytes(
                    "1"+"\n");
                outToClient.writeBytes(
                    "Auction does not exist."+"
                lock.endWrite();
                continue;
        }
                //The client must be up-to-date
        if(!last_bid.equals(
            rs.getString("current_bid"))){
                outToClient.writeBytes("1"+"\n"
                outToClient.writeBytes(
                "Someone else has bid before yo
                lock.endWrite();
                continue;
        }
        int int_bet = Integer.parseInt(bet);
        int int_current_bid =
            Integer.parseInt(rs.getString("curr
        int int_minimum_bid =
            Integer.parseInt(rs.getString("minim

        //Bid must be high enough.
```

```
if( int_bet < int_minimum_bid +int_curre
        outToClient.writeBytes("1"+"\n"
        outToClient.writeBytes(
            "Your bet is too small."+"\r
        lock.endWrite();
        continue;
}

String datetime = rs.getString("datetime
Timestamp ts = Timestamp.valueOf(datetim

//Cannot bid on expired auctions.
if(now.after(ts)){
        outToClient.writeBytes("1"+"\n"
        outToClient.writeBytes(
            "This auction has expired."+
        lock.endWrite();
        continue;
}

//query to update the item
query = "UPDATE `auctions` SET"+
    " current_bid = '"+bet+"', bid_owner
    +nickname+"' WHERE auctions.id "+
    "= '"+auction_id+"'";
db.runQuery(query);

//get the updated info of the item.
query =  "SELECT * FROM auctions"+
" WHERE id = '"+auction_id+"'";
rs = db.getData(query);
rs.next();

String type = "0";
String current_bid =
    rs.getString("current_bid");
String minimum_bid =
    rs.getString("minimum_bid");
```

```
            datetime = rs.getString("datetime");
            String item_owner =
                rs.getString("item_owner");
            String bid_owner =
                rs.getString("bid_owner");
            lock.endWrite();
                    //Send the updated auction infor
            outToClient.writeBytes(type+"\n");
            outToClient.writeBytes(auction_id+"\n")
            outToClient.writeBytes(current_bid+"\n"
            outToClient.writeBytes(minimum_bid+"\n"
            outToClient.writeBytes(datetime+"\n");
            outToClient.writeBytes(item_owner+"\n")
            outToClient.writeBytes(bid_owner+"\n");
        }
        else{           //auction creation

            //Read the item specifications.
            String starting_bid =
                inFromClient.readLine();;
            String minimum_bid =
                inFromClient.readLine();;
            String datetime =
                inFromClient.readLine();
            String owner =
                inFromClient.readLine();

            //lock everyone out
            lock.startWrite();
            DBConnect db = new DBConnect();
                    //Make the database query to cre
            String query = "INSERT INTO "
                +"`auctions` (`id`, `current_bid`,"+
                " `minimum_bid`, `datetime`,"+
                " `item_owner`, `bid_owner`)"+
                " VALUES (NULL, '"+ starting_bid +
                "', '"+minimum_bid+"', '"+
                datetime+"', '"+owner+"', '"+
                                        owner+"');";
                                                db.runQuery(query);

                                //make the query to get the last createc
                                query = "SELECT * FROM auctions ORDER BY
                                ResultSet rs = db.getData(query);
                                lock.endWrite();
                                        //Read the info of the item from
                                rs.next();
                                String type = "0";
                                String auction_id = rs.getString("id");
                                starting_bid = rs.getString("current_bic
                                minimum_bid = rs.getString("minimum_bid'
                                datetime = rs.getString("datetime");
                                String item_owner = rs.getString("item_c
                                String bid_owner = rs.getString("bid_owr
                                        //Send the auction info to the
                                outToClient.writeBytes(type+"\n");
                                outToClient.writeBytes(auction_id+"\n")
                                outToClient.writeBytes(starting_bid+"\n'
                                outToClient.writeBytes(minimum_bid+"\n"
                                outToClient.writeBytes(datetime+"\n");
                                outToClient.writeBytes(item_owner+"\n")
                                outToClient.writeBytes(bid_owner+"\n");
                        }
                }


        } catch (Exception e) {
                e.printStackTrace();
        }

}
}
```

6.2.3 DBConnect.java

```
//Mehmet G l en
//2013400075
//mehmetgulsen95@hotmail.com
//CMPE436-Term

import java.sql.*;

/*
 * This class is used for connecting to the database
 */
public class DBConnect {
        private Connection con;
        private Statement st;
        private ResultSet rs;

        public DBConnect(){
                try {
                        Class.forName("com.mysql.jdbc.Driver");
                        con = DriverManager.getConnection(
                            "jdbc:mysql://localhost:3306/auctioneer",
                            "java", "123456");
                        System.out.println("database connected!");
                        st = con.createStatement();

                } catch (Exception e) {
                        System.out.println("Error: "+e);
                }
        }

        public ResultSet getData(String query){
                try {
                        rs = st.executeQuery(query);
                        return rs;


                } catch (Exception e) {
                        System.out.println("Error: "+e);
                }
        return rs;
    }

    public void runQuery(String query){
        try {

            st.executeUpdate(query);

        } catch (Exception e) {
            System.out.println("Error: "+e);
        }
    }
}
```

6.2.4 Lock.java

```
//Mehmet  G  l  en
//2013400075
//mehmetgulsen95@hotmail.com
//CMPE436-Term

/*
 * Reader-Writer solution 2 from lecture notes
 */
public class Lock {
    int numReaders = 0;
    Semaphore mutex = new Semaphore(1);
    Semaphore wlock = new Semaphore(1);

    public void startRead(){
        mutex.P();
        numReaders++;
        if(numReaders == 1)
            wlock.P();
        mutex.V();
    }

    public void endRead(){
```

```
                mutex.P();
                numReaders--;
                if(numReaders==0){
                        wlock.V();
                }
                mutex.V();
        }

        public void startWrite(){
                wlock.P();
        }

        public void endWrite(){
                wlock.V();
        }
}
```

6.2.5 Semaphore.java

//Mehmet Gülşen //2013400075 //mehmetgulsen95@hotmail.com //CMPE436-Term

/* * Binary Semaphore from the slides on Piazza. */ public class Semaphore

private int value;

public Semaphore(int value) this.value = value;

public synchronized void P() if (value == 0) try wait(); catch (InterruptedException e)

value = 0;

public synchronized void V() value = 1; notifyAll();