



Network 1

10636454

Masa Rateb Abu Aisheh

12027844

First Task :

In this task I will use my computer as server and client in the same time.

The IP for the Server and client is "192.168.1.18" and The port is "7851", As we can see in the picture:

Link-local IPv6 address:	fe80::1e32:d8a7:fc7a:e569%19
IPv4 address:	192.168.1.18

And I will use two computers, one for the server (my computer) and one for the client.

The IP for the client is "192.168.1.13" .

Properties

SSID:	Salman...
Protocol:	Wi-Fi 4 (802.11n)
Security type:	WPA-Personal
Network band:	2.4 GHz
Network channel:	2
Link speed (Receive/Transmit):	72/72 (Mbps)
Link-local IPv6 address:	fe80::8faa:512f:cb76:f068%13
IPv4 address:	192.168.1.3
IPv4 DNS servers:	185.17.235.133 185.17.235.134
Manufacturer:	Intel Corporation
Description:	Intel(R) Wireless-AC 9462
Driver version:	22.200.0.6
Physical address (MAC):	50-EB-71-B0-4E-92

Copy

 [Get help](#)

Socket programming by UDB:

UDB Client code:

```
UDPClient.java X
1 import java.io.*;
2
3
4 public class UDPClient {
5     public static void main(String[] args) throws IOException {
6
7         while(true){
8             BufferedReader datauser = new
9                 BufferedReader(new
10                     InputStreamReader(System.in));
11             DatagramSocket clientsocket = new DatagramSocket( );
12
13             InetAddress ipaddress = InetAddress.getByName("192.168.1.18");
14             byte[] send = new byte[1024];
15             byte[] receive = new byte[1024];
16
17             String sent=datauser.readLine();
18             send =sent.getBytes();
19
20             DatagramPacket sendpacket=new
21                 DatagramPacket(send,send.length,ipaddress,7851);
22             clientsocket.send(sendpacket);
23
24             DatagramPacket receivepacket=new
25                 DatagramPacket(receive,receive.length);
26             clientsocket.receive(receivepacket);
27
28             String modifiedSentence = new String (receivepacket.getData());
29             System.out.println("FROM SERVER:" + modifiedSentence);
30             clientsocket.close();
31
32         }
33     }
34 }
35 }
```

UDB Client code to explain:

1. I will first establish an input stream so that the user can input data (the Vehicle plate ID).
2. `DatagramSocket clientSocket = new DatagramSocket();` I shall create the client socket.
3. After that, provide the client with the server's IP address and port number so they can connect.
4. create an array of the type "byte" to hold the message and convert it to that format while sending and receiving the message.

5. Store the user's input as a string, convert it to byte format, and then store it in the array (the send array defined above), before preparing the message to be sent by creating the packet (the packet contains the message and the size of message and Ip address and port for server).
6. Next, create the packet to receive the server's message (contain the message and the size of it).
7. last, print the message after receiving it and cutting off your connection to the server.

UDB Server code:

```
UDPServer.java X
1 import java.io.BufferedReader;
2 import java.io.File;
3 import java.io.FileReader;
4 import java.io.IOException;
5 import java.io.RandomAccessFile;
6 import java.net.DatagramPacket;
7 import java.net.DatagramSocket;
8 import java.net.InetAddress;
9 import java.net.SocketException;
10 import java.util.Scanner;
11
12 public class UDPServer {
13     public static void main(String[] args) throws IOException {
14
15         String value="null";
16         int i=0;
17
18         DatagramSocket serverSocket=new DatagramSocket(7851);
19         byte[]receiveData=new byte[1024];
20         byte[]sendData=new byte[1024];
21
22         while(true) {
23             DatagramPacket receivePacket=new
24                 DatagramPacket(receiveData,receiveData.length);
25
26             serverSocket.receive(receivePacket);
27             String sen=new
28                 String(receivePacket.getData());
29
30             InetAddress ipaddress = receivePacket.getAddress();
31
32             int port =receivePacket.getPort();
33             String S=null;
34             File file=new File("Data.txt");
35             final RandomAccessFile R;
36             R=new RandomAccessFile(new File("Data.txt"), "r");
37
38             String array[]=new String[2];
```

```

UDPServer.java
36      R=new RandomAccessFile(new File("Data.txt"), "r");
37
38      String array[]=new String[2];
39      FileReader F;
40      F = new FileReader(file);
41      BufferedReader reader = new
42          BufferedReader(F);
43
44      while((S = reader.readLine())!=null){
45
46          String k=S;
47          array =k.split(",");
48          if(array[0].trim().equals(sen.trim())){
49              value=array[1];
50              i=1;
51              break;
52          }
53          else{
54              i=0;
55          }
56      }
57      if(i == 0) {
58          value = "Vehicle is not found";
59      }
60      else {
61      }
62
63      sendData= value.getBytes();
64      DatagramPacket sendPacket=new
65          DatagramPacket(sendData,sendData.length,ipaddress,port);
66      serverSocket.send(sendPacket);
67
68  }
69  }
70  }
71  }
72
73

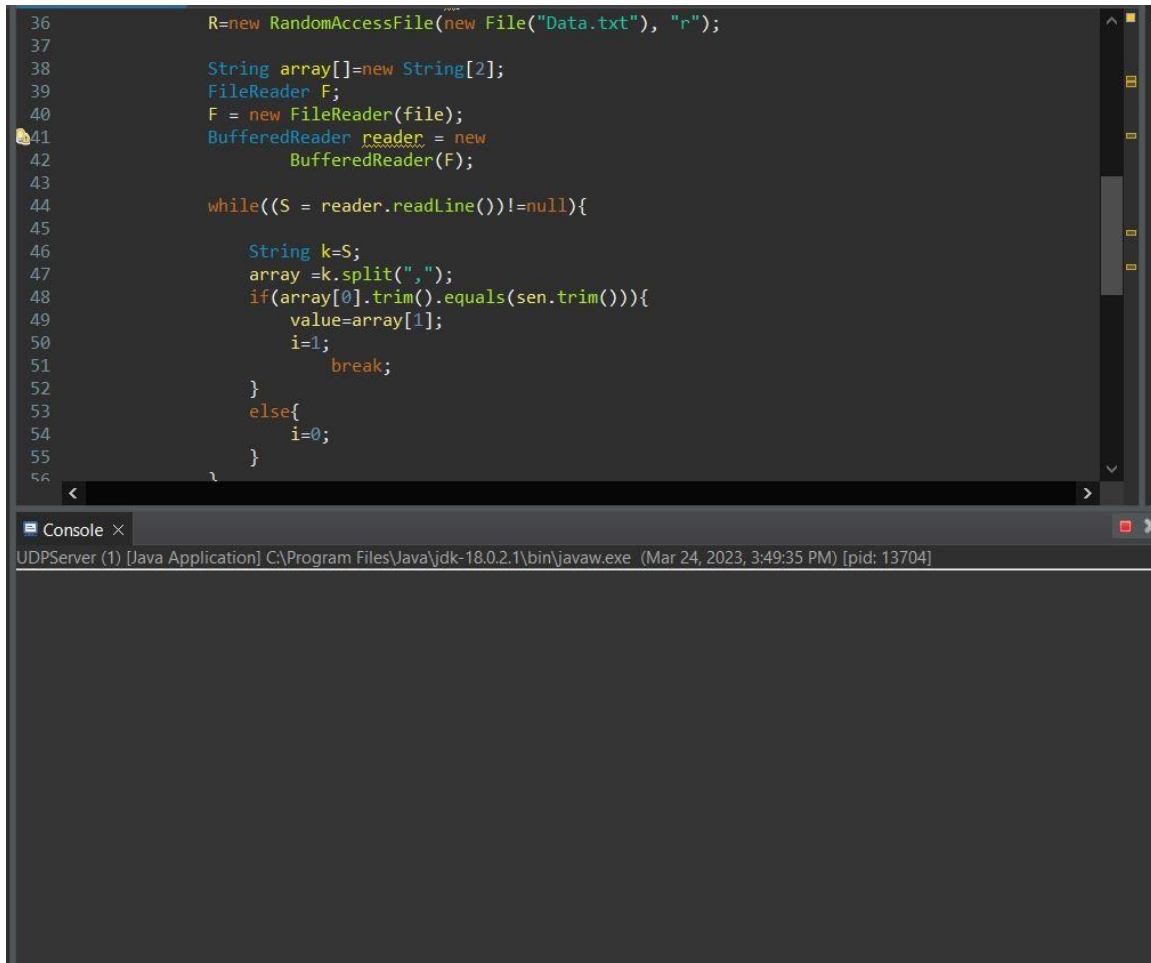
```

UDB Server code to explain:

1. I'll define two arrays of type byte and create a socket on port 7851 to store the message in.
2. The server waits for any client to establish a connection.
3. After creating the packet to receive the message from the client (which includes the message's size), receive the message, and store it in the string.
4. Obtain the sender's IP address and port number and enter them in the send packet (to make sure that the message arrive to the correct client).
5. After that, we use the file (the server has the file, and the file contains all the information), so we split the line in the file by (,), and then I loop through the file until I reach the id (the message from the client), at which point we store the name, of course, in an array of type byte and send it to the client.

Apply the UDB Code:

1. first I run the code of server in the first computer, and the server wait until the client make connection:

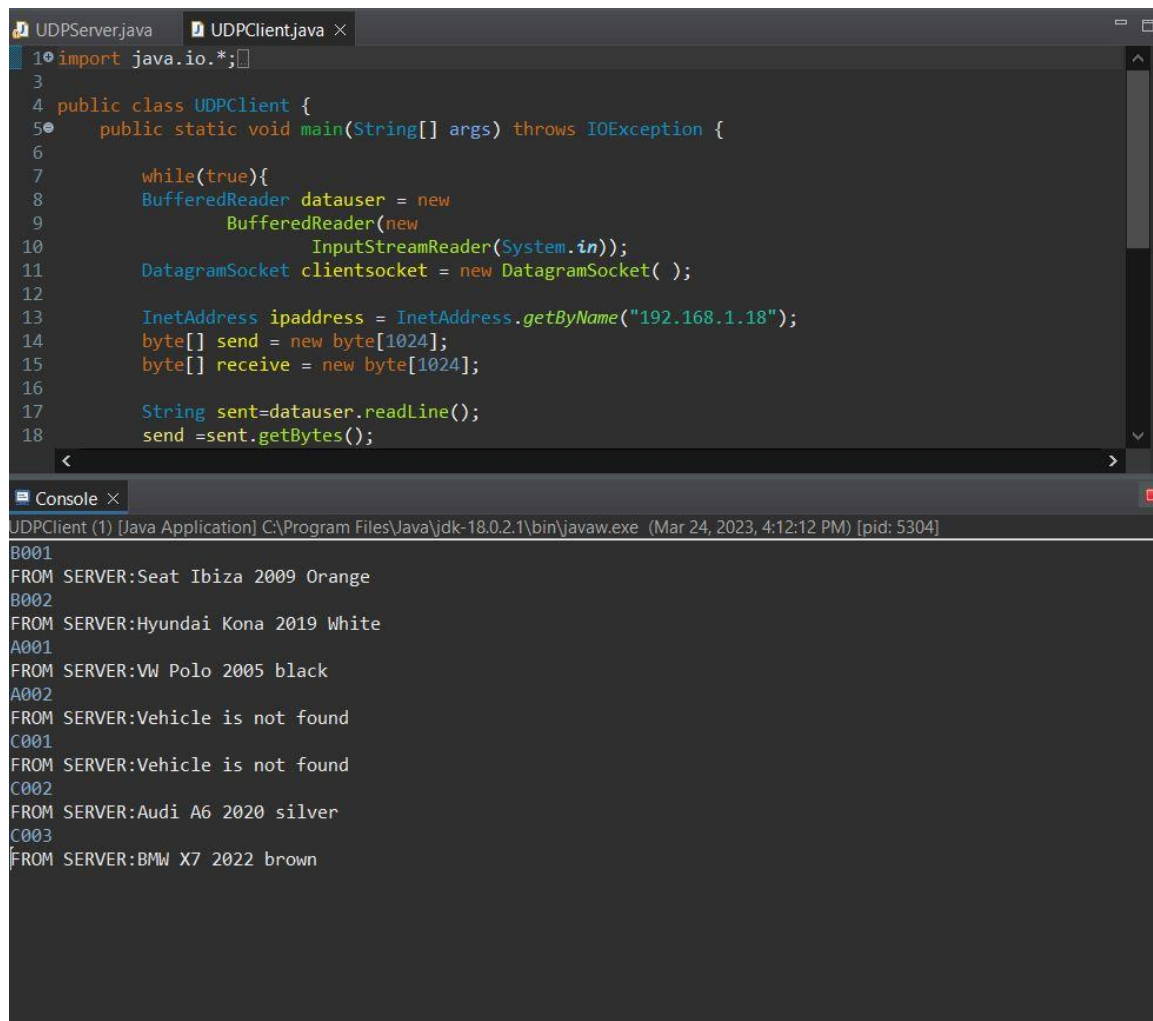


The screenshot displays a Java IDE with a code editor and a console window. The code editor shows a Java program for a UDP server. It starts by creating a `RandomAccessFile` object for `Data.txt` in read mode. Then, it initializes a `String` array of size 2, a `FileReader` object `F`, and a `BufferedReader` object `reader`. A `while` loop reads lines from the file until `readLine()` returns `null`. Inside the loop, it splits the line by a comma, checks if the first part matches the expected sensor name (e.g., "sen"), and if so, it updates the `value` in the array and increments the index `i`. If not, it increments `i` directly. The console window at the bottom shows the command prompt for the `UDPServer` application, indicating it is running in the background.

```
36         R=new RandomAccessFile(new File("Data.txt"), "r");
37
38         String array[]=new String[2];
39         FileReader F;
40         F = new FileReader(file);
41         BufferedReader reader = new
42             BufferedReader(F);
43
44         while((S = reader.readLine())!=null){
45
46             String k=S;
47             array =k.split(",");
48             if(array[0].trim().equals(sen.trim())){
49                 value=array[1];
50                 i=1;
51                 break;
52             }
53             else{
54                 i=0;
55             }
56         }
```

Console ×
UDPServer (1) [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Mar 24, 2023, 3:49:35 PM) [pid: 13704]

2. run the client code and put the data then the server gives me the expected response like:



The screenshot displays a Java IDE with two tabs: `UDPServer.java` and `UDPCClient.java`. The `UDPCClient.java` tab is active, showing the following code:

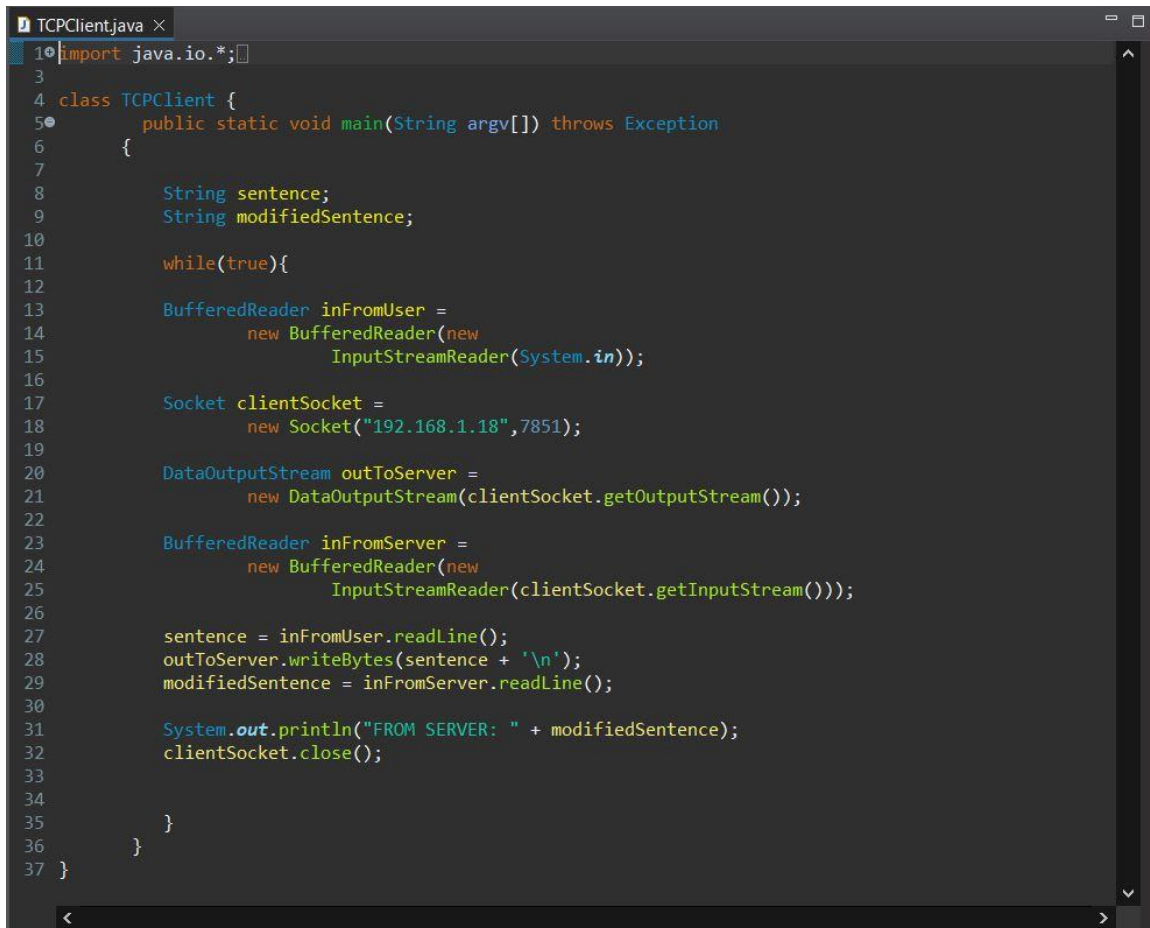
```
1 import java.io.*;
2
3
4 public class UDPCClient {
5     public static void main(String[] args) throws IOException {
6
7         while(true){
8             BufferedReader datauser = new
9                 BufferedReader(new
10                     InputStreamReader(System.in));
11             DatagramSocket clientsocket = new DatagramSocket( );
12
13             InetAddress ipaddress = InetAddress.getByName("192.168.1.18");
14             byte[] send = new byte[1024];
15             byte[] receive = new byte[1024];
16
17             String sent=datauser.readLine();
18             send =sent.getBytes();
```

Below the code editor is a console window titled "Console" showing the output of the application. The output consists of several lines of text, each preceded by a label (B001, B002, A001, A002, C001, C002, C003) and followed by a message from the server:

```
UDPCClient (1) [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Mar 24, 2023, 4:12:12 PM) [pid: 5304]
B001
FROM SERVER:Seat Ibiza 2009 Orange
B002
FROM SERVER:Hyundai Kona 2019 White
A001
FROM SERVER:VW Polo 2005 black
A002
FROM SERVER:Vehicle is not found
C001
FROM SERVER:Vehicle is not found
C002
FROM SERVER:Audi A6 2020 silver
C003
FROM SERVER:BMW X7 2022 brown
```

Socket programming by TCP:

TCP Client code:

A screenshot of a Java IDE window titled 'TCPClient.java'. The code is as follows:

```
1 import java.io.*;
2
3
4 class TCPClient {
5     public static void main(String argv[]) throws Exception
6     {
7
8         String sentence;
9         String modifiedSentence;
10
11         while(true){
12
13             BufferedReader inFromUser =
14                 new BufferedReader(new
15                     InputStreamReader(System.in));
16
17             Socket clientSocket =
18                 new Socket("192.168.1.18", 7851);
19
20             DataOutputStream outToServer =
21                 new DataOutputStream(clientSocket.getOutputStream());
22
23             BufferedReader inFromServer =
24                 new BufferedReader(new
25                     InputStreamReader(clientSocket.getInputStream()));
26
27             sentence = inFromUser.readLine();
28             outToServer.writeBytes(sentence + '\n');
29             modifiedSentence = inFromServer.readLine();
30
31             System.out.println("FROM SERVER: " + modifiedSentence);
32             clientSocket.close();
33
34         }
35     }
36 }
37 }
```

TCP Client code to explain:

1. we create the Stream that enables the user to enter data, then we create the socket for the client and establish a connection with the server by entering the server's IP address and port number, and last we create the output and input Streams.
2. Save and deliver the user-inputted data to the server.
3. print the message you downloaded from the server (receive message).
4. shut down the connection once we have sent and received all the info we require.

TCP Server code:

```
TCPServer.java
29 R=new RandomAccessFile(new File("Data.txt"),"r");
30
31 String array[] = new String[2];
32 FileReader F;
33 F = new FileReader(file);
34 BufferedReader reader = new BufferedReader(F);
35
36 while((s = reader.readLine()) != null){
37     String k=s;
38     array=k.split(",");
39
40     if(array[0].trim().equals(clientSentence.trim())){
41         value = array[1];
42         i = 1;
43         break;
44     }
45
46     else{
47         i = 0;
48     }
49 }
50
51
52 if (i == 0) {
53     value = "Vehicle is not found";
54 }
55
56 else {
57     System.out.println();
58 }
59
60 System.out.println(value);
61 outToClient.writeBytes(value+'\n');
62
63 }
64 }
65 }
66
67

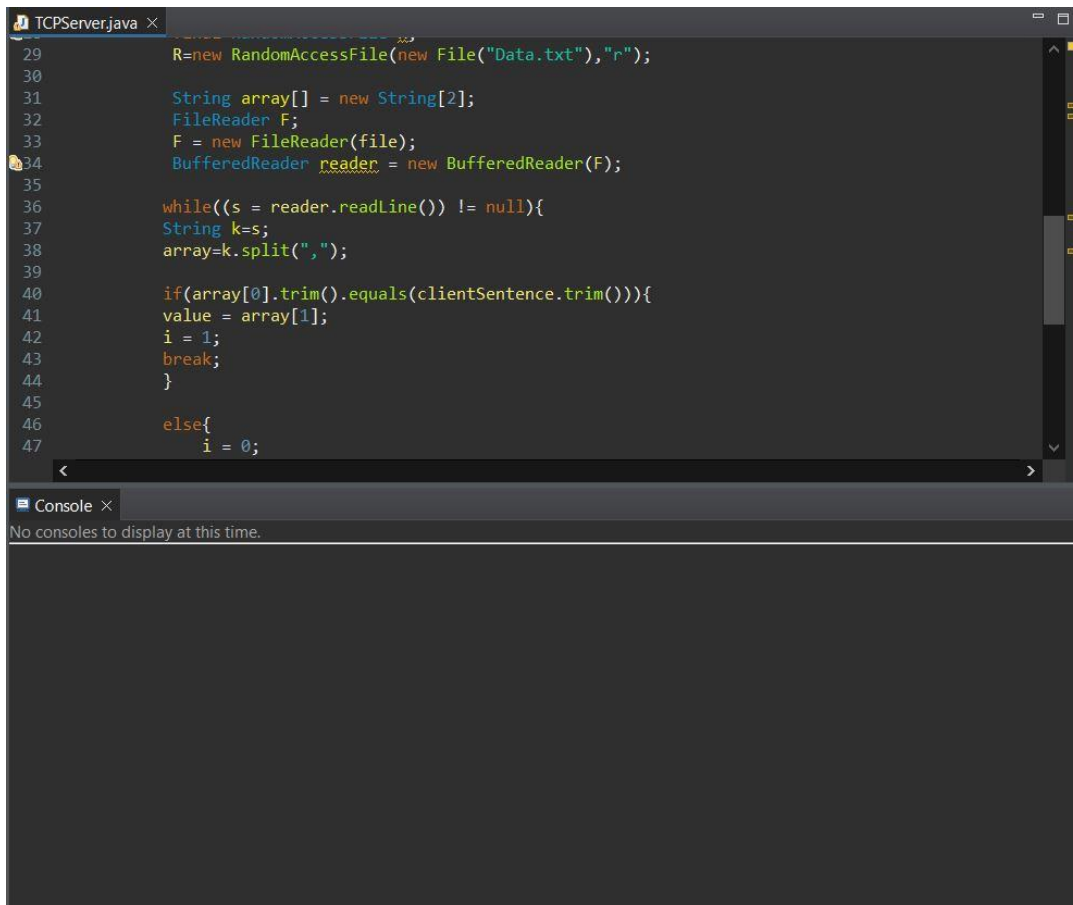
TCPServer.java
1 import java.io.*;
2 import java.net.*;
3
4 class TCPServer {
5     public static void main(String argv[]) throws Exception
6     {
7         String value="0";
8         String clientSentence;
9         String capitalizedSentence;
10        ServerSocket welcomeSocket = new ServerSocket(7851);
11
12        while(true) {
13            Socket connectionSocket = welcomeSocket.accept();
14            BufferedReader inFromClient = new
15                BufferedReader(new
16                    InputStreamReader(connectionSocket.getInputStream()));
17
18            DataOutputStream outToClient = new
19                DataOutputStream(connectionSocket.getOutputStream());
20
21            clientSentence = inFromClient.readLine();
22            //Read Data
23            int i=0;
24            String s=null;
25
26            File file=new File("Data.txt");
27
28            final RandomAccessFile R;
29            R=new RandomAccessFile(new File("Data.txt"),"r");
30
31            String array[] = new String[2];
32            FileReader F;
33            F = new FileReader(file);
34            BufferedReader reader = new BufferedReader(F);
35
36            while((s = reader.readLine()) != null){
37                String k=s;
38                array=k.split(",");
39
```

TCP Server code to explain:

1. Before creating the input stream and output stream to accept the message from the client and resend it, we will first construct the socket for the server and wait till the client sends the message.
2. after the client establishes a connection with the server, the server reads this message, which we then store in the string.
3. after this, we split the line in the file by (,) and search data by the client message (which we had previously stored in the string).
4. after this, we insert the data in the string and send it to the client.

Apply the TCP Code:

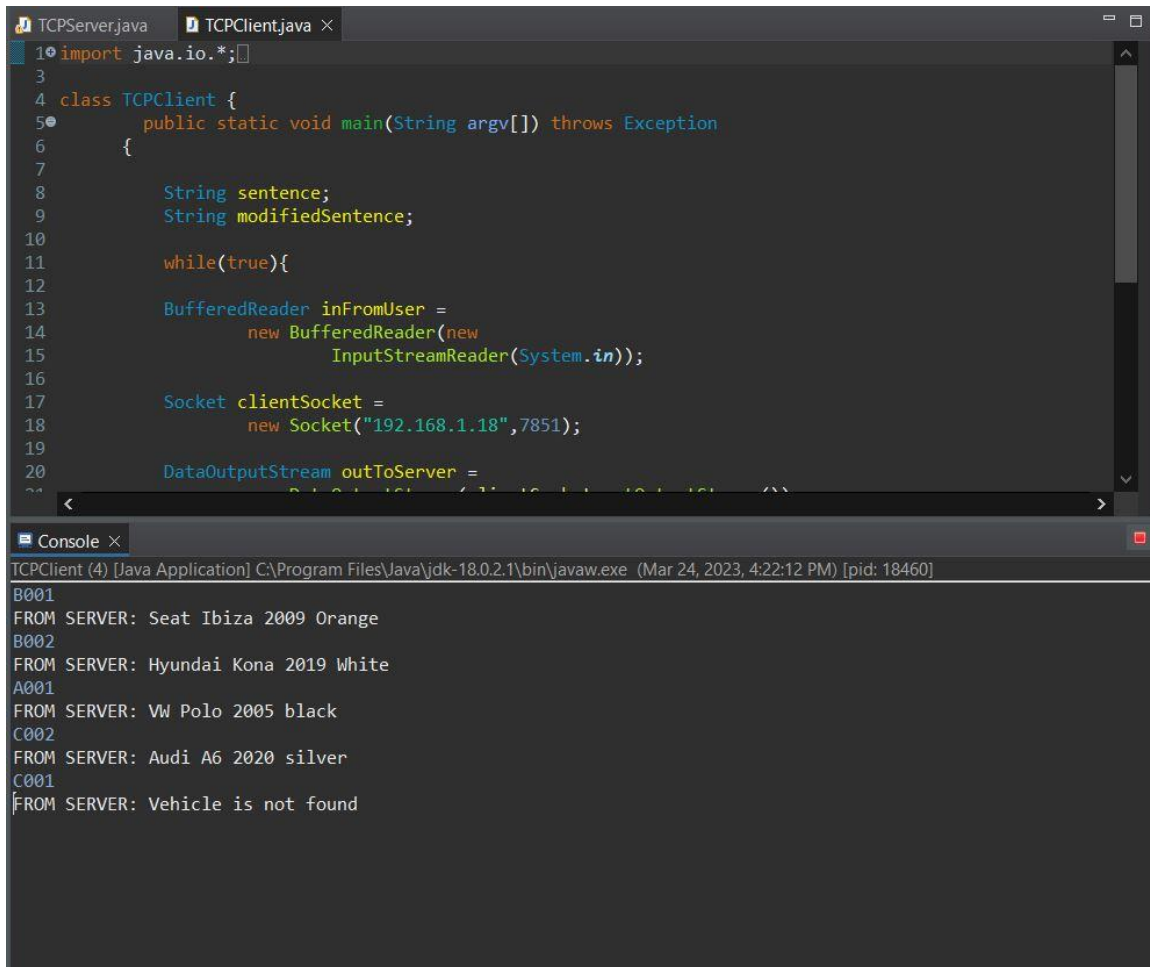
1. first I run the code of server in the first computer, and the server wait until the client make connection:



```
29      R=new RandomAccessFile(new File("Data.txt"),"r");
30
31      String array[] = new String[2];
32      FileReader F;
33      F = new FileReader(file);
34      BufferedReader reader = new BufferedReader(F);
35
36      while((s = reader.readLine()) != null){
37          String k=s;
38          array=k.split(",");
39
40          if(array[0].trim().equals(clientSentence.trim())){
41              value = array[1];
42              i = 1;
43              break;
44          }
45
46          else{
47              i = 0;
```

Console ×
No consoles to display at this time.

2. run the client code and put the data then the server gives me the expected response like:



The screenshot shows a Java IDE with two tabs: `TCPServer.java` and `TCPClient.java`. The `TCPClient.java` tab is active, displaying the following code:

```
1 import java.io.*;
2
3
4 class TCPClient {
5     public static void main(String argv[]) throws Exception
6     {
7
8         String sentence;
9         String modifiedSentence;
10
11         while(true){
12
13             BufferedReader inFromUser =
14                 new BufferedReader(new
15                     InputStreamReader(System.in));
16
17             Socket clientSocket =
18                 new Socket("192.168.1.18", 7851);
19
20             DataOutputStream outToServer =
```

Below the code editor is a console window titled `Console`. It shows the output of the `TCPClient` application, which is a Java application running in the command prompt. The output is as follows:

```
TCPClient (4) [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Mar 24, 2023, 4:22:12 PM) [pid: 18460]
B001
FROM SERVER: Seat Ibiza 2009 Orange
B002
FROM SERVER: Hyundai Kona 2019 White
A001
FROM SERVER: VW Polo 2005 black
C002
FROM SERVER: Audi A6 2020 silver
C001
FROM SERVER: Vehicle is not found
```