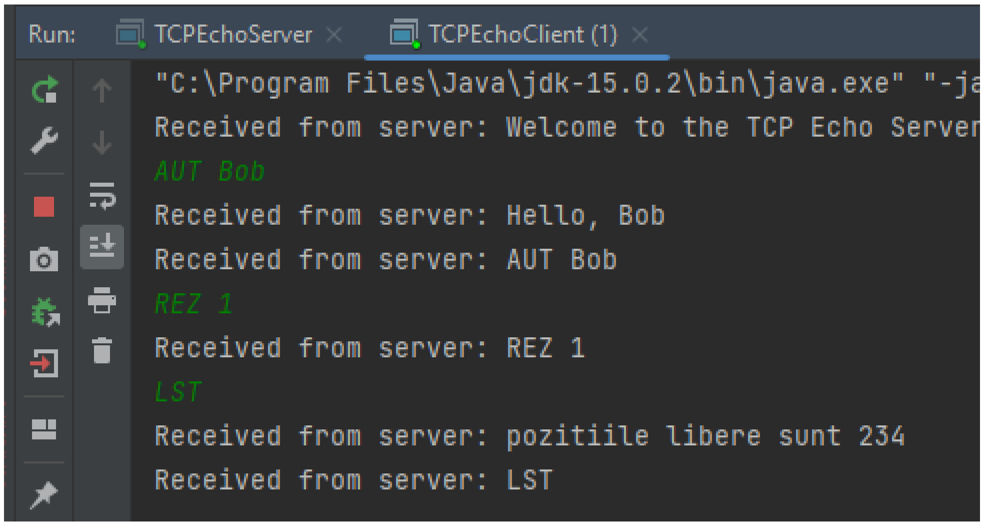
Tema 1

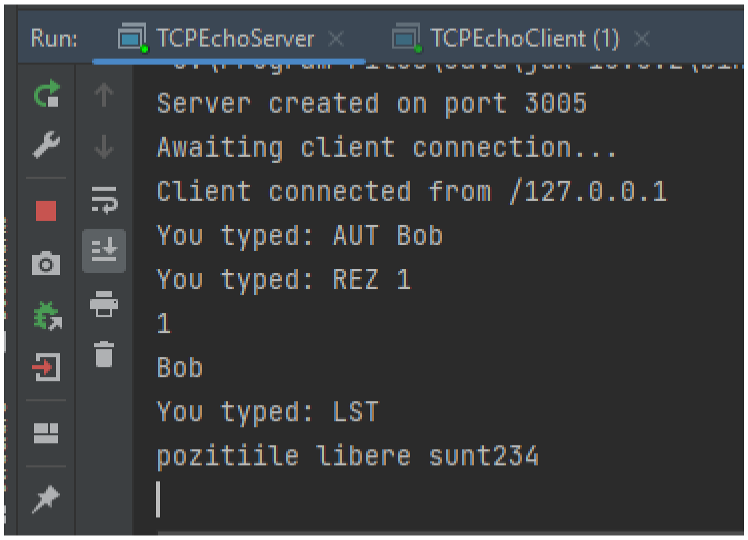
Am reusit sa fac un sistem distribuit care creaza si inchide conexiuni, rezerva locurile de autobus pentru utilizatori si listeaza pozitiile libere.

Am creat o clasa server, in care folosind ServerSocket creez un server socket ce primeste conexiuni, apoi intr-un fir de executie ce foloseste metodele dataOutputStream si dataInputStream trimit si prrimesc mesaje de la client. Daca mesajul clientului incepe cu anumite comenzi. Precum "AUT", serverul va face anumite modificari asupra unui vector.

Clasa client se conecteaza la server si trimite/primeste mesaje.



```
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-ja
Received from server: Welcome to the TCP Echo Server
AUT Bob
Received from server: Hello, Bob
Received from server: AUT Bob
REZ 1
Received from server: REZ 1
LST
Received from server: pozitiile libere sunt 234
Received from server: LST
```



```
Server created on port 3005
Awaiting client connection...
Client connected from /127.0.0.1
You typed: AUT Bob
You typed: REZ 1
1
Bob
You typed: LST
pozitiile libere sunt234
```

SERVER

```
package org.example;

import java.net.*;
import java.io.*;

public class TCPEchoServer {
    public static void main(String args[]) {
        int[] rezervat = {0, 0, 0, 0, 0};
        String Persoana = "";
        String loc;
        String [] initiala=new String[rezervat.length];
        int port = 3005;
        ServerSocket serverSocket = null;
        DataInputStream dataInputStream = null;
        DataOutputStream dataOutputStream = null;
        try {
// open a server socket
            serverSocket = new ServerSocket(port);
            System.out.println("Server created on port " + port);
            System.out.println("Awaiting client connection...");
// await for a client connection
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected from " +
clientSocket.getInetAddress());
            dataInputStream = new
DataInputStream(clientSocket.getInputStream());
            dataOutputStream = new
DataOutputStream(clientSocket.getOutputStream());
        } catch (IOException e) {
            System.out.println("Problems initializing server: " + e);
            System.exit(1);
        }
// communicate with the client
        try {
            dataOutputStream.writeUTF("Welcome to the TCP Echo Server!");
            String input;
            while (true) {
// read data in from client
                input = dataInputStream.readUTF();


                System.out.println("You typed: " + input);
                if (input.indexOf("AUT ") == 0) {
                    Persoana = input.substring(4);
                    dataOutputStream.writeUTF("Hello, " + Persoana);
                } else if (input.indexOf("REZ ") == 0) {
                    loc = input.substring(4);
                    rezervat[Integer.parseInt(loc)] = 1;
                    initiala[Integer.parseInt(loc)]=Persoana;
                    System.out.println(rezervat[Integer.parseInt(loc)]);
                    System.out.println(initiala[Integer.parseInt(loc)]);

                } else if (input.equals("BYE")) {
                    dataOutputStream.writeUTF("You disconected from server");
                    dataInputStream.close();
                    dataOutputStream.close();
                    serverSocket.close();
                } else if (input.equals("LST")) {
                    int poz = 0;
                    for (int i = 0; i < rezervat.length; i++) {
                        if (rezervat[i] == 0)
                            poz = poz * 10 + i;
                    }
                    dataOutputStream.writeUTF("pozitiile libere sunt " +
poz);
                    System.out.println("pozitiile libere sunt" + poz);

                }


// write data back to client
                dataOutputStream.writeUTF(input);


            }
        } catch (IOException e) {
            System.out.println("Client disconnected from server");
        }
        try {
            serverSocket.close();
        } catch (Exception e) {
        }
    }
}
```

CLIENT

```
package org.example;

import java.net.*;
import java.io.*;

class TCPEchoReader extends Thread {
    public TCPEchoReader(DataInputStream input) {
        dataInputStream = input;
        active = true;
    }

    public void run() {
        while (active) {
            try {
                String message = dataInputStream.readUTF();
                System.out.println("Received from server: " + message);
            } catch (IOException e) {
                System.out.println(e);
                active = false;
            }
        }
    }

    public boolean active;
    public DataInputStream dataInputStream;
}

public class TCPEchoClient {
    public static void main(String[] args) {
        String address = "localhost";
        int port = 3005;
        Socket socket = null;
        DataInputStream dataInputStream = null;
        DataOutputStream dataOutputStream = null;
        BufferedReader keyboardReader = null;
// Connect to the server...
        try {
            socket = new Socket(address, port);
// Obtain the streams...
            dataInputStream = new DataInputStream(socket.getInputStream());
            dataOutputStream = new
DataOutputStream(socket.getOutputStream());
            keyboardReader = new BufferedReader(new
InputStreamReader(System.in));
        } catch (IOException e) {
            System.out.println("Problems initialising: " + e);
            System.exit(1);
        }
        try {
// Start the listening thread...
            TCPEchoReader reader = new TCPEchoReader(dataInputStream);
            reader.setDaemon(true);
            reader.start();
            String input;
            while (true) {
// read data in from the keyboard
                input = keyboardReader.readLine();
// send data to server
                dataOutputStream.writeUTF(input);
            }
        } catch (IOException e) {
        }
        try {
            socket.close();
        } catch (IOException e) {
        }
        }
}
```