Computer Networks Lab 7 Assignment

Name: Muvvala N V Sai Gurudeep

Roll No: SE20UARI103

Group: A2

Branch: AI

1. Develop a file transfer application using TCP Sockets

Server Code:

import java.io.\*;

import java.net.\*;

class TCPServer{

public static void main(String argv[]) throws Exception{

String clientSentence;

String capitalizedSentence;

// Creating server socket with port number. Note we should have same port number as mentioned in the Client code

ServerSocket welcomeSocket = new ServerSocket(6789);

System.out.println("Reading contents of file sent by client...");

// Always accept the connection whenever the client requests

while(true){

// Accept the connection when connection requested

Socket connectionSocket = welcomeSocket.accept();

// Create an input stream to read from Client

BufferedReader inFromClient = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));

// Create an output stream to send to client

DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());

// Read things sent by client

clientSentence = inFromClient.readLine();

// Convert the information read to captial case.

capitalizedSentence = clientSentence.toUpperCase() + '\n';

// Send the capitalised sentence to client thus telling receivied the message correctly.

outToClient.writeBytes(capitalizedSentence);

}

}

}

Client Code:

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

class TCPClient{

public static void main(String args[]) throws Exception{

String sentence;

String modifiedSentence;

// Reading from file

File myObj = new File("ClientMessage.txt");

// Accessing contents of file

Scanner inFromUser = new Scanner(myObj);

System.out.println("Sending file to server...");

// Accessing contents of file line by line

while(inFromUser.hasNextLine()){

// Storing a line in curline from file

String curline = inFromUser.nextLine();

// Initialising a clientSocket with desired port and ip address

Socket clientSocket = new Socket("127.0.0.1",6789);

// Want to send to the server from client

DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());

// Want to read from server by client

BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

sentence = curline;

// Writing it to server

outToServer.writeBytes(sentence + '\n');

// Reading the servers meesage

modifiedSentence = inFromServer.readLine();

// Printing the servers message

System.out.println("From Server: " + modifiedSentence);

// Closing the client socket

clientSocket.close();

}

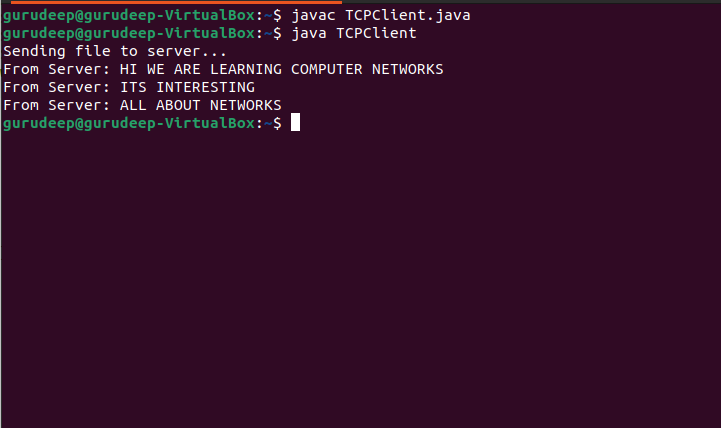
}

}

Explanation:

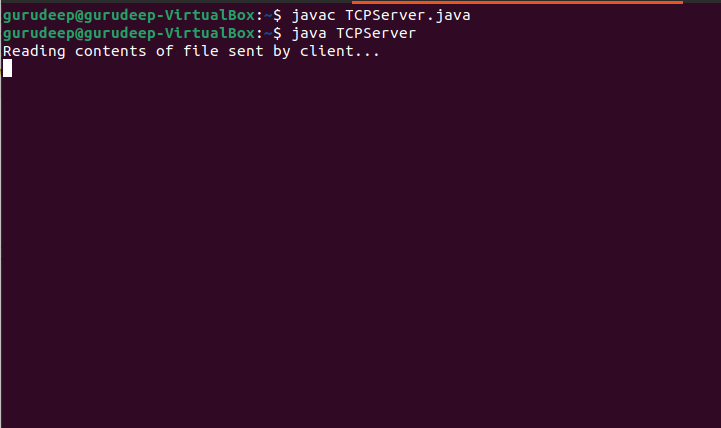
Client Code:

1. First we create an object of type File to access a file named “ClientMessage.txt”.
2. Next we create an object for the Scanner class called inFromUser to access this file object contents.
3. We go in a while loop till we reach the end of file i.e all contents are read:
   1. When encountered with a new line we would store it in a variable called curline of type String.
   2. We later initialize a clientSocket with server host number Ip:127.0.0.1 and server port number 6789.
   3. We create an output stream to send data from client to server.
   4. We create an input stream to receive data from client to server.
   5. We take a message in the curline and write it to the server. But before writing it to the server it should be converted into bytes.
   6. We then wait to receive a message from the server. If received then we store that message in the modifiedSentence variable.
   7. Next we print the server's message and close the ClientSocket.



Server Code:

1. Here we create serverSocket with server port number 6789.
2. Server will always be in a listening state waiting for connection. Hence we have a infinite while loop:
   1. Server accepts the request whenever the client requests. For transferring of messages to client if needed the server takes client ip address and client port number with the help of connectionSocket.
   2. Next we create an inputStream to read from the client.
   3. Create an output stream to write to the client.
   4. Reads the message or data sent by the client and stores it in clientSentence.
   5. Capitalizes the read sentence to uppercase and sends it to the client as an indication that message has been received correctly.



Therefore in this way a file transfer application is implemented using TCP sockets.