**PROBLEM STATEMENT:**

**Write a TCP Day-Time server program that returns the current time and date. Also write a TCP client program that sends request to the server to get the current time and date. Choose your own formats for the request/reply messages.**

**DESIGN OF REQUEST/REPLY PROTOCOL:**

Server-Side:

1. The server listens on a designated port for incoming client connections.
2. Upon receiving a client connection, the server reads the incoming request continuously.
3. If the received request matches the expected "GetTime" request, the server generates the current time and date and sends a response with the "OK" status code and the current time and date in the desired format.
4. If the request is invalid or any error occurs during processing, the server sends a response with the "ERROR" status code and an error message.
5. The server closes the connection for client after getting “Exit” as message from the client.
6. Server keeps listening to the any other client request again.

Client-Side:

1. The client establishes a connection to the server using the server's IP address and port.
2. The client sends the "GetTime" request.
3. The client reads the response from the server.
4. If the status code is "OK", the client extracts and displays the current time and date provided in the response.
5. Client can send as many as request they want to the server as “GetTime”.
6. If the status code is "ERROR", the client displays the error message from the response.
7. The client can exit by typing “Exit” on the console.

**SOURCE CODE:**

Server-Side code:

import java.io.\*;

import java.net.\*;

public class DayTimeServer {

public static void main(String[] args) {

int port = 12345; // Choose a suitable port number

BufferedReader br=null;

try (ServerSocket serverSocket = new ServerSocket(port)) {

System.out.println("Server is listening on port " + port);

while (true) {

try (Socket clientSocket = serverSocket.accept()) {

System.out.println("Client connected: " + clientSocket.getInetAddress());

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

String msg=br.readLine();

if(msg==null) continue;

if(msg.equals("GetTime")){

// Get current time and date

String dateTime = java.time.LocalDateTime.now().toString();

System.out.println("Client requested for date and time...");

// Send the response to the client

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

out.println("Server Date and Time: " + dateTime);

}

// Close the client socket

else if(msg.equals("exit")){

clientSocket.close();

}

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

Client-Side code:

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class DayTimeClient {

public static void main(String[] args) {

String serverAddress = "localhost"; // Change to the server's IP address if needed

int serverPort = 12345; // Same port as the server

while(true){

try (Socket socket = new Socket(serverAddress, serverPort)) {

// Create input and output streams

System.out.println("Enter MSG to get date & time::");

String msg=new Scanner(System.in).nextLine();

if(msg.equals("exit")) break;

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

// Send a request to the server

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

out.println(msg);

// Receive and print the server's response

String response = in.readLine();

System.out.println("Server Response: " + response);

} catch (IOException e) {

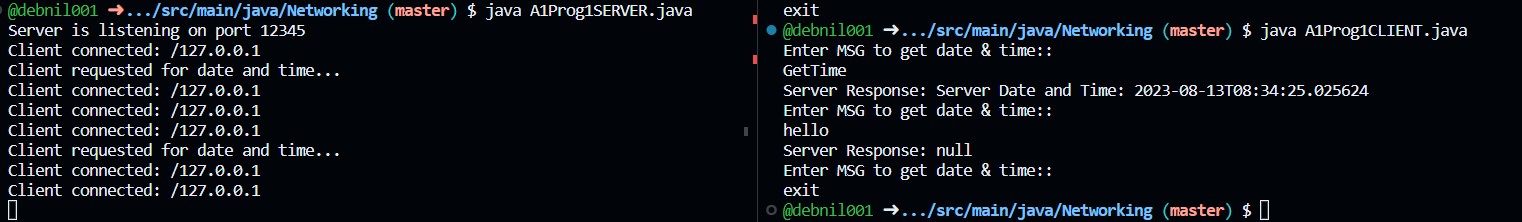
e.printStackTrace();

}

}

}

}

OUTPUT:

**PROBLEM STATEMENT:**

**Write a TCP Math server program that accepts any valid integer arithmetic expression, evaluates it and returns the value of the expression. Also write a TCP client program that accepts an integer arithmetic expression from the user and sends it to the server to get the result of evaluation. Choose your own formatsfor the request/reply messages.**

**DESIGN OF REQUEST/REPLY PROTOCOL:**

Server-Side:

1. The server listens on a designated port for incoming client connections.
2. Upon receiving a client connection, the server reads the incoming request.
3. The server attempts to evaluate the received mathematical expression using a user defined calculate function.
4. If the expression is valid and can be evaluated successfully, the server sends the result of the evaluation.
5. If the expression is invalid or evaluation fails, the server sends a response and an error message “Invalid Expression”.
6. The server keeps listening to the connected client or for any new request.

Client-Side:

1. The client establishes a connection to the server using the server's IP address and port.
2. The client reads a mathematical expression from the user until user gives “Exit” as message.
3. The client sends the expression to the server.
4. The client reads the response from the server until it encounters a “Exit”.
5. If the status code is "OK", the client extracts and displays the result provided in the response.
6. If invalid expression is sent by the client, then it displays the error message from the response.
7. The client closes the connection if “Exit” is typed on the console.

**SOURCE CODE:**

Server-Side code:

package Networking;

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Server {

//calculate function for the expression evaluation

public static int calculate(String s) {

int len=s.length();

int res=0;

char sign='+';

Stack<Integer> st=new Stack<>();

int i;

for(i=0;i<len;i++){

char c=s.charAt(i);

if(Character.isDigit(c)){

int val=c-48;

while(i+1<len && Character.isDigit(s.charAt(i+1))){

val=val\*10+(s.charAt(i+1)-48);

i++;

}

if(sign=='+'){

st.push(val);

}

else if(sign=='-'){

st.push(-val);

}

else if(sign=='\*'){

res=st.pop();

st.push(val\*res);

}

else if(sign=='/'){

res=st.pop();

st.push(res/val);

}

}

else if(c!=' '){//must be sign

if(i>0 && s.charAt(i-1)!=' ' && !Character.isDigit(s.charAt(i-1))){

System.out.println("Invalid Expression!");

break;

}

sign=c;

}

}

if(i!=len) return Integer.MAX\_VALUE;

res=0;

while(!st.isEmpty()){

res+=st.pop();

}

return res;

}

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

ServerSocket ss=new ServerSocket(50000);

//server is initialized on port 50000 with default loop back address

System.out.println("Server started...ready to accept client");

PrintWriter out=null;//to send output to the client side

BufferedReader in=null;//to get input from the client side

while(true){

Socket client=ss.accept();//ready to listen to the client

System.out.println("Client connected"+client);//new client has arrived

out=new PrintWriter(client.getOutputStream(),true);//setting output stream with the client

in=new BufferedReader(new InputStreamReader(client.getInputStream()));//setting input stream with the client

while(true){// as long as client sends expression, server will evaluate until exit is sent

String exp=in.readLine();//reading the expression from the client side

System.out.println("Client: "+exp);//printing the current expression on the console

if(exp.equals("exit")) break;//if exit is sent

int res=Server.calculate(exp);//evaluating the result for the given expression

if(res==Integer.MAX\_VALUE)

out.println("Invalid Expression!!! Please enter valid expression");

else

out.println("Evaluated result from server "+res);//sending the result back to the client

}

//freeing up all the resources

client.close();

out.close();

in.close();

}

}

}

Client-Side code:

package Networking;

import java.io.\*;

import java.net.Socket;

import java.util.Scanner;

public class A1Prog2CLIENT {

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

Socket sc=new Socket("localhost",50000);//creating connection to the server's port and address

PrintWriter out=null;

BufferedReader in=null;

while(true){

out=new PrintWriter(sc.getOutputStream(),true);// output stream object for getting output from server

in=new BufferedReader(new InputStreamReader(sc.getInputStream()));// input stream object for getting input from server

Scanner console=new Scanner(System.in);

System.out.println("Enter an arithmetic expression:");

String exp=console.next();//reading expression from console of the client

out.println(exp);//sending to the server in form of output stream

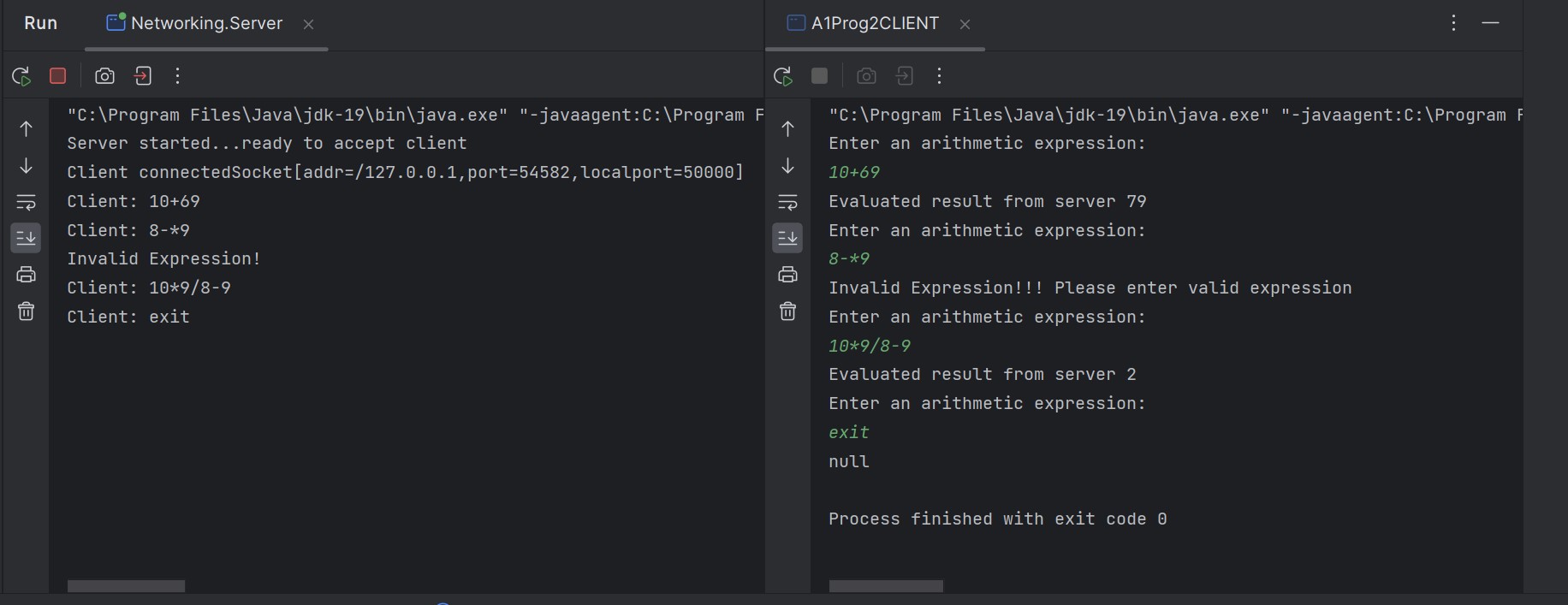
System.out.println(in.readLine());//getting the result and printing on the screen

if(exp.equals("exit")) break;//if the console message is exit then exit from the program

}

}

}

OUTPUT:

**PROBLEM STATEMENT:**

**Implement a UDP server program that returns the permanent address of a student upon receiving a request from a client. Assume that, a text file that stores the names of students and their permanent addresses is available local to the server. Choose your own formats for the request/reply messages.**

**DESIGN OF REQUEST/REPLY PROTOCOL:**

Server-Side:

1. The server listens on a designated port for incoming client connections.
2. Upon receiving a client connection, the server reads the incoming request.
3. The server attempts to search the provided name on the local storage.
4. If the provided name is not found then server sends “Student not found” to the server.
5. If name is present in the local file, it sends the corresponding address of the given student name by searching into the file.
6. The server keeps listening for new clients.

Client-Side:

1. The client establishes a connection to the server using the server's IP address and port.
2. The client reads a student name from console.
3. The client sends the name to the server.
4. The client gets only one chance to get the address.
5. If the given student is present in the server, client will get the address of the specified student.
6. If the name is not present in the server it will get a message from server as “Student not found”
7. In either case client exits from the program.

**SOURCE CODE:**

Server-Side code:

package Networking;

import java.io.\*;

import java.net.\*;

class UDPServer {

public static void main(String[] args) {

final int serverPort = 12345;

final String studentInfoFile = "C:\\Users\\HP\\Documents\\NetBeansProjects\\JavaSocket\\src\\main\\java\\Student.txt";

//student file location in local machine

try (DatagramSocket serverSocket = new DatagramSocket(serverPort)) {

System.out.println("Server listening on port " + serverPort);//start listening the client req if any

byte[] receiveData = new byte[1024];

while (true) {

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

serverSocket.receive(receivePacket);//data received from client

System.out.println("client requested for student info...");

String studentName = new String(receivePacket.getData(), 0, receivePacket.getLength());//converting the student name from bytes to string

String response = findStudentAddress(studentName, studentInfoFile);//function call to search in file

InetAddress clientAddress = receivePacket.getAddress();

int clientPort = receivePacket.getPort();//get the clients port to send the data

byte[] sendData = response.getBytes();

DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, clientAddress, clientPort);

serverSocket.send(sendPacket);//packet sent to the client

System.out.println("Info send to the client");

}

} catch (IOException e) {

e.printStackTrace();

}

}

//function to search in file

private static String findStudentAddress(String studentName, String studentInfoFile) {

try (BufferedReader br = new BufferedReader(new FileReader(studentInfoFile))) {

String line;

while ((line = br.readLine()) != null) {

String[] parts = line.split(":");

if (parts.length == 2 && parts[0].trim().equalsIgnoreCase(studentName)) {

return "OK\n" + parts[1].trim() + "\n";

}

}

} catch (IOException e) {

e.printStackTrace();

}

return "ERROR\nStudent not found.\n";

}

}

Client-Side code:

package Networking;

import java.io.\*;

import java.net.\*;

class UDPClient {

public static void main(String[] args) {

final String serverAddress = "127.0.0.1"; // server's IP address

final int serverPort = 12345; //server port

try (DatagramSocket clientSocket = new DatagramSocket()) {//creating UDP connection

BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Enter student name: ");

String studentName = userInput.readLine();//getting the student name to search in server

byte[] sendData = studentName.getBytes();//converting the data into bytes

InetAddress serverIPAddress = InetAddress.getByName(serverAddress);

DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, serverIPAddress, serverPort);

clientSocket.send(sendPacket);//sending the data in form of packet through UDP connection

byte[] receiveData = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);//response received from client

String response = new String(receivePacket.getData(), 0, receivePacket.getLength());

System.out.println("Server response: " + response);//printing the response

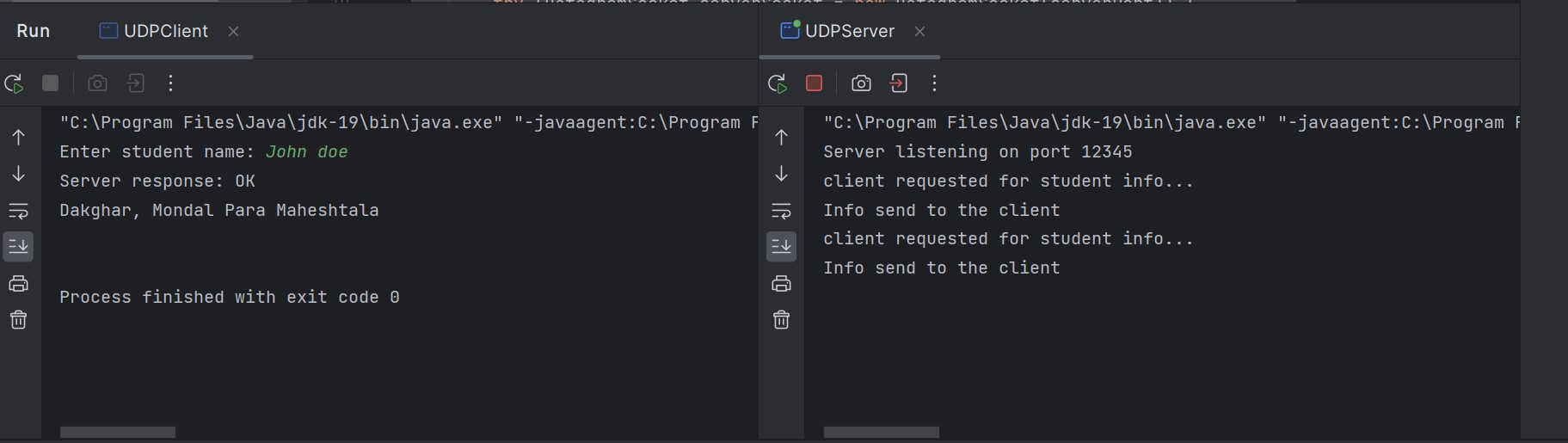
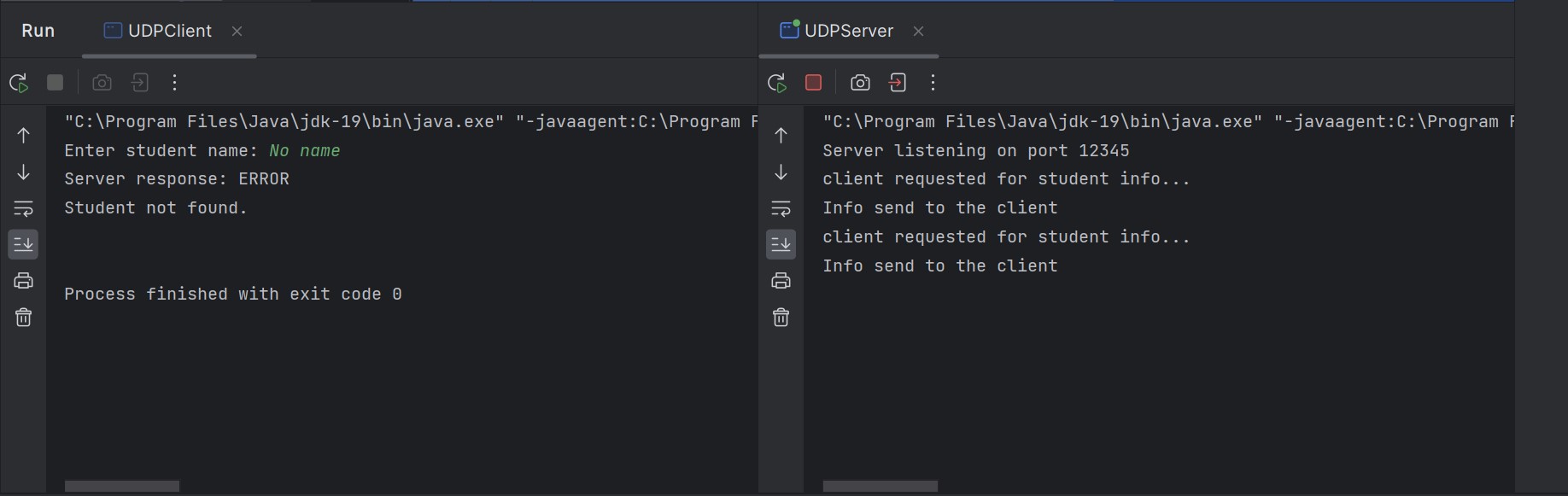
} catch (IOException e) {

e.printStackTrace();

}

}

}

**OUTPUT:**