Experiment No. 1

//1) Aim: To develop distributed application through implementation client server communication on java socket

1. Server side :-

import java.io.\*;

import java.net.\*;

public class Server {

public static void main(String[] args) {

try {

ServerSocket serverSocket = new ServerSocket(1234); // Create a server socket on port 1234

System.out.println("Server listening on port 1234...");

Socket clientSocket = serverSocket.accept(); // Wait for a client to connect

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

// Create input and output streams for client communication

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

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

// Read messages from the client and respond

String message;

while ((message = in.readLine()) != null) {

System.out.println("Received message from client: " + message);

out.println("Server response: " + message.toUpperCase()); // Send back a response

}

// Close connections

in.close();

out.close();

clientSocket.close();

serverSocket.close();

} catch (IOException e) {

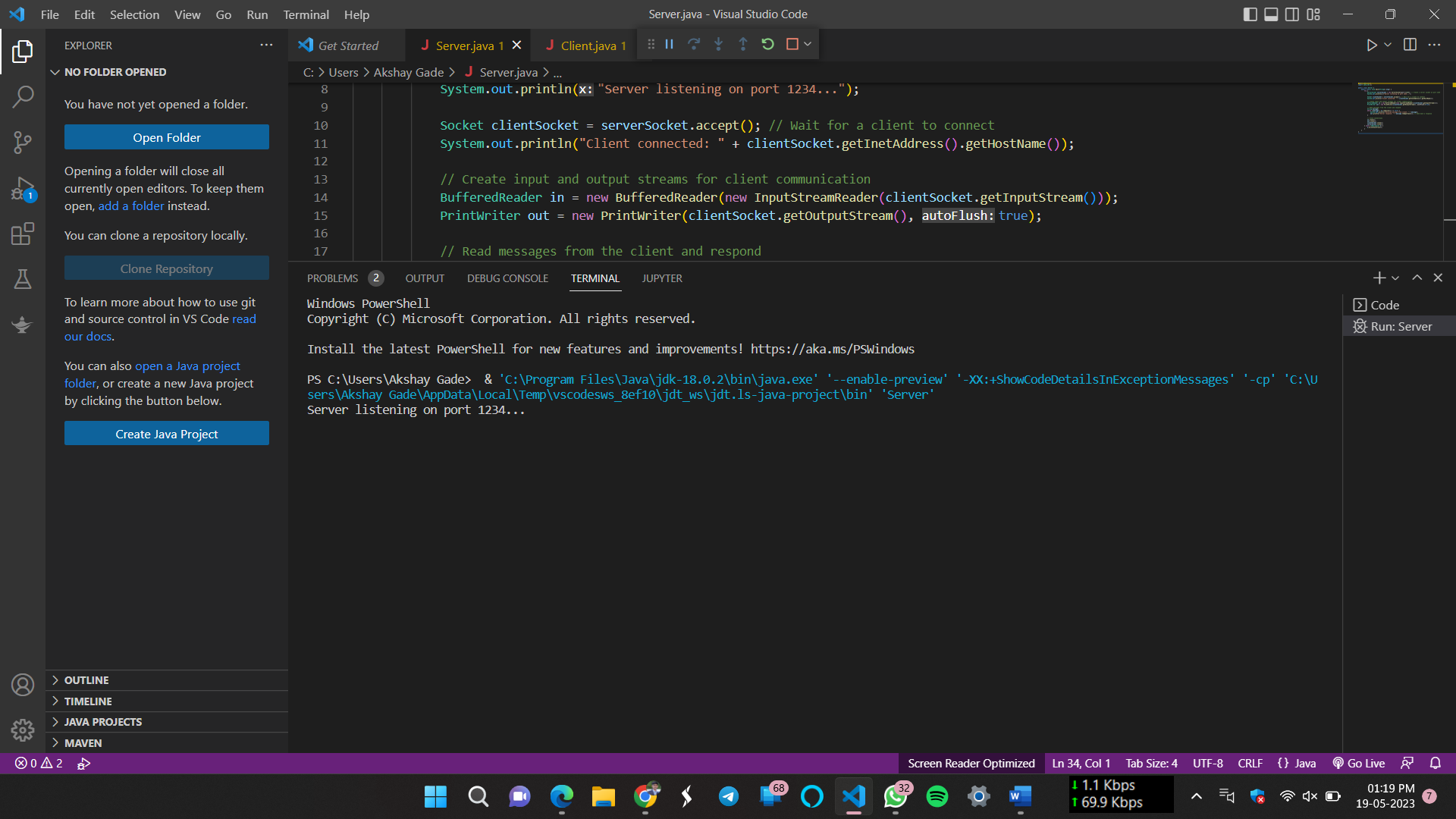
e.printStackTrace();

}

}

}

Output :-



1. Client side :-

import java.io.\*;

import java.net.\*;

public class Client {

public static void main(String[] args) {

try {

Socket clientSocket = new Socket("localhost", 1234); // Connect to the server on localhost:1234

System.out.println("Connected to server: " + clientSocket.getInetAddress().getHostName());

// Create input and output streams for client communication

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

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

// Send messages to the server and receive responses

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

String message;

while ((message = consoleIn.readLine()) != null) {

out.println(message); // Send message to the server

String response = in.readLine(); // Receive response from the server

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

}

// Close connections

in.close();

out.close();

clientSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Output :-

