/\*

Name : Rohit Narayan Telgote

PRN : 1941054

Batch : B4

\*/

// Aim : Design distributed application which consists of a server and client using threads

**Server.java**

import java.io.\*;

import java.net.\*;

public class Server {

public static void main(String args[]) {

int port = 6789;// port number

Server server = new Server(port);// calling constructructor of server final and initialisig in

// object(server) of serverfina;

server.startServer(); // calling startserver function

}

// declare a server socket and a client socket for the server;

// declare the number of connections

ServerSocket echoServer = null;// declaring echo server

Socket clientSocket = null;// declaring client socket

int numConnections = 0;

int port;

public Server(int port) {

this.port = port;// port is initialised in member variable port

}

public void stopServer() // Switching off the server

{

System.out.println("Server cleaning up.");

System.exit(0);

}

public void startServer() // starting the server

{

// Try to open a server socket on the given port

// Note that we can't choose a port less than 1024 if we are not

// privileged users (root)

try {

echoServer = new ServerSocket(port);// initialising server by calling server socket function

} catch (IOException e) {// if error persist

System.out.println(e);

}

System.out.println("Server is started and is waiting for connections.");

System.out.println("With multi-threading, multiple connections are allowed.");

System.out.println("Any client can send -1 to stop the server.");

// Whenever a connection is received, start a new thread to process the

// connection

// and wait for the next connection.

while (true) {

try {

clientSocket = echoServer.accept();// clientsocket is intialised

numConnections++;// number of connectections is increament

Server2Connection oneconnection = new Server2Connection(clientSocket, numConnections, this);// caling

// constructor

//of

// server2

// funcction

new Thread(oneconnection).start();// run function is called for each thread

} catch (IOException e) {

System.out.println(e);

}

}

}

}

class Server2Connection implements Runnable {

BufferedReader is;

PrintStream os;

Socket clientSocket;

int id;

Server server;

public Server2Connection(Socket clientSocket, int id, Server server) {

this.clientSocket = clientSocket;

this.id = id;

this.server = server;

System.out.println("Connection " + id + " established with: " + clientSocket);

try {

is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));// data input stream of server

os = new PrintStream(clientSocket.getOutputStream());// output stream of serve

} catch (IOException e) {

System.out.println(e);

}

}

public void run() {

String line;

try {

boolean serverStop = false;

while (true) {

line = is.readLine(); // getting line from client

System.out.println("Received " + line + " from Connection " + id + ".");// displaying

int n = Integer.parseInt(line);

if (n == -1) {// checking if to stop the server or client

serverStop = true;

break;

}

if (n == 0)

break;

os.println("" + n \* n); // writing square in input buffer of client

}

System.out.println("Connection " + id + " closed.");

// closing client connection

is.close();

os.close();

clientSocket.close();

if (serverStop)

server.stopServer();// this is used to close all the connection

} catch (IOException e) {

System.out.println(e);// if error occured

}

}

}

**Client.java**

import java.io.\*;

import java.net.\*;

public class Client {

public static void main(String[] args) {

String hostname = "localhost";// hostname

int port = 6789; // port numbrt

// declaration section:

// clientSocket: our Clientfinal socket

// os: output stream

// is: input stream

Socket clientSocket = null;

DataOutputStream os = null;

BufferedReader is = null;

// Initialization section:

// Try to open a socket on the given port

// Try to open input and output streams

try {

clientSocket = new Socket(hostname, port); // initialising client socket

os = new DataOutputStream(clientSocket.getOutputStream());// output stream of client socket

is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));// input stream of client

// socket

} catch (UnknownHostException e) {

System.err.println("Don't know about host: " + hostname);

} catch (IOException e) {

System.err.println("Couldn't get I/O for the connection to: " + hostname);

}

// If everything has been initialized then we want to write some data

// to the socket we have opened a connection to on the given port

if (clientSocket == null || os == null || is == null) {

System.err.println("Something is wrong. One variable is null.");

return;

}

try {

while (true) {

System.out.print("Enter an integer (0 to stop connection, -1 to stop server): ");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));// taking input from terminal

String keyboardInput = br.readLine();// storing input in keyboard

os.writeBytes(keyboardInput + "\n");// writing keyboard into input stream buffer of server

int n = Integer.parseInt(keyboardInput);

if (n == 0 || n == -1) {// condiition to get out from loop

break;

}

String responseLine = is.readLine();// getting input stream in response line

System.out.println("Server returns its square as: " + responseLine);

}

// clean up:

// close the output stream

// close the input stream

// close the socket

os.close();

is.close();

clientSocket.close();

} catch (UnknownHostException e) {

System.err.println("Trying to connect to unknown host: " + e);

} catch (IOException e) {

System.err.println("IOException: " + e);

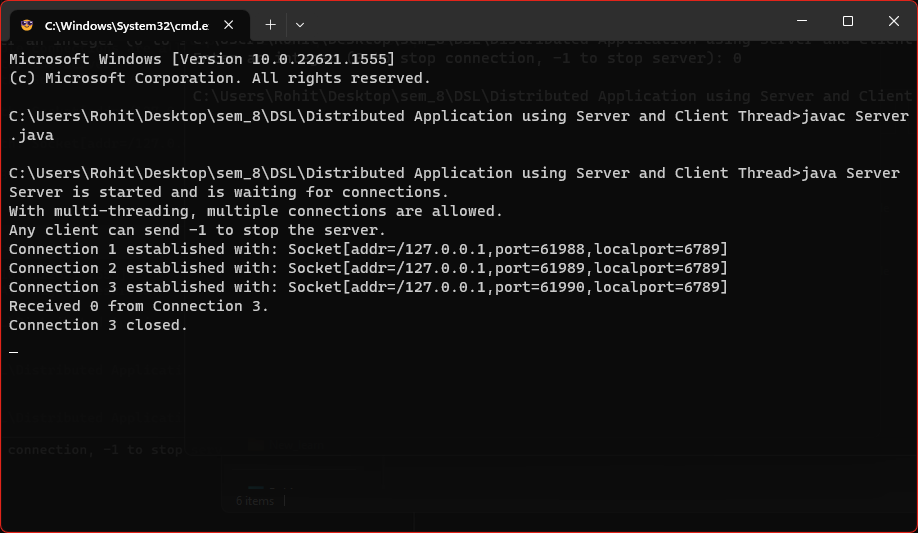
}

}

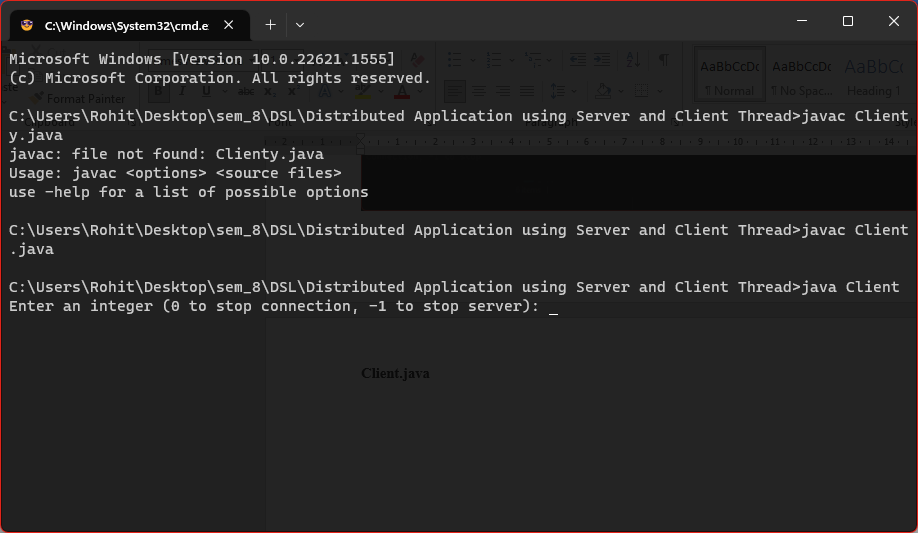
}

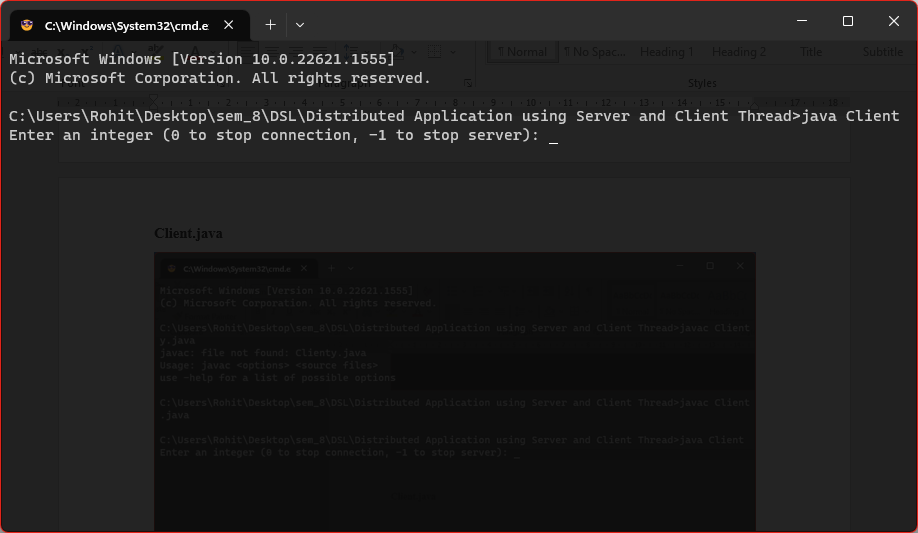
**Output :**

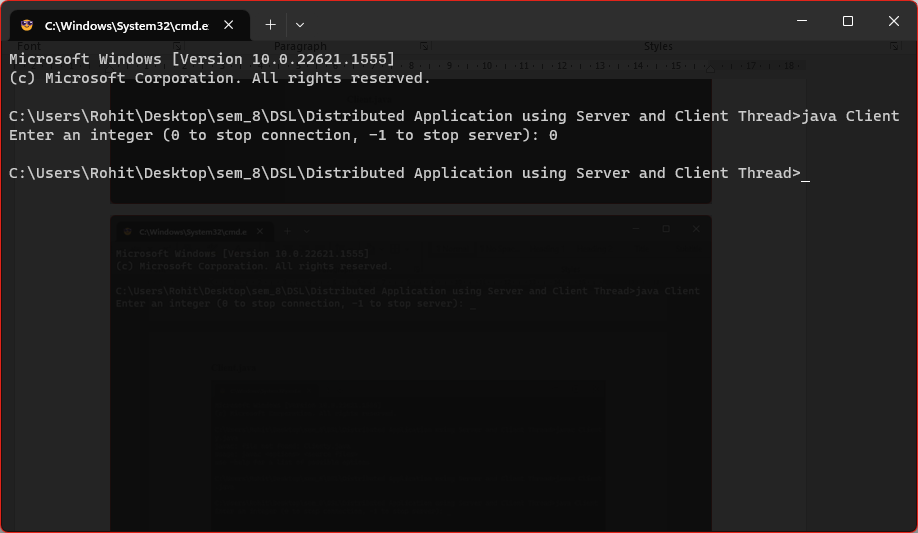
**Server.java**

****

**Client.java**

****

****

****