|  |  |
| --- | --- |
| Ex.No.9 | IMPLEMENTATION OF GROUPCHAT APPLICATION USING  MULTICAST SOCKETS |

|  |
| --- |
| **AIM:** |

To develop group chat application using multicast sockets and multithreading

|  |
| --- |
| **THEORY:** |

**Multicasting:**

Multicasting in computer network is a group communication, where a sender(s) send data to multiple receivers simultaneously. It supports one – to – many and many – to – many data transmission across LANs or WANs. Through the process of multicasting, the communication and processing overhead of sending the same data packet or data frame in minimized.

**Addressing in Multicasting:**

A multicast address is a logical identifier for a group of [hosts](https://en.wikipedia.org/wiki/Host_(network)) in a [computer network](https://en.wikipedia.org/wiki/Computer_network) that are available to process datagrams or frames intended to be [multicast](https://en.wikipedia.org/wiki/Multicast) for a designated [network service](https://en.wikipedia.org/wiki/Network_service). An IP multicast group address is used by sources and the receivers to send and receive multicast messages. Sources use the group address as the IP destination address in their data packets. Receivers use this group address to inform the network that they are interested in receiving packets sent to that group. Multicast addressing can be used in the [link layer](https://en.wikipedia.org/wiki/Link_layer) (layer 2 in the [OSI model](https://en.wikipedia.org/wiki/OSI_model)), such as [Ethernet](https://en.wikipedia.org/wiki/Ethernet) multicast, and at the [internet layer](https://en.wikipedia.org/wiki/Internet_layer) (layer 3 for OSI) for [Internet Protocol](https://en.wikipedia.org/wiki/Internet_Protocol) Version 4 (IPv4) or Version 6 (IPv6) multicast. The IPv4 and IPv6 are known as class D addressing used mainly for multicasting.

**APIs and methods required to implement Multicast Sockets:**

API: class java .net.datagram, class java.util.\*

Connection establishment Methods: Inetaddress, getByName(),MulticastSocket, joinGroup(),leaveGroup()

Thread: Thread(),start(),run()

Chat establishment: getBytes(),DatagramPacket,send(object),receive(object)

**Differences between Group chat and Chat in Client/Server Mode**

|  |  |
| --- | --- |
| **Group chat** | **Chat in client/server mode** |
| Group chat is based on UDP | Chat in client/server mode is based on either TCP or UDP |
| Multicast addressing | Unicast addressing |
| Multicast can be of either many – many or one -many | Unicast in purely one-one(server-client) |
| Datagram packets are sent and received from source address to destination address | Unicast packets are encapsulated and delivered using TCP via the Transport layer. |

|  |
| --- |
| **ALGORITHM:** |

**1.Chat using Client/Server Mode**

Client side:

* Class NormalChatClient is created.
* A socket is initilised with host name and port number.
* In try block,
* DataInputStream ,DataOutputStream ,BufferedReader are initialised.
* Two strings servermessage and clientmessage are initilised.
* While loop is enabled until clientmessage reads ‘bye’.
* In while loop, the string entered by the user is stored in clientmessage and written using outstream.
* The string is read by servermessage using instream.readUTF() and the servermessage is printed.
* Socket,instream,outstream are closed followed by catch block.

Server side:

* Class NormalChatServer in initialised.
* In main(),try block is initialised.
* A serversocket is created with port number.
* Incoming requests are accepted by accept() passed to client object.
* Loop is enabled.
* DataInputStream ,DataOutputStream ,BufferedReader are initialised.
* Data is read in clientmessage through instream and printed.
* The user entered string is written in servermessage.
* Socket,instream,outstream are closed followed by catch block.

**2.Group Chat**

* start
* Class GroupChat is initialised.
* Variables are initialised.
* In main function, the arguments passed during execution is checked by if loop.
* In else block, a try block is initialised.
* In try block,Inetaddress method is initialised with getByname().
* The argument is passed to ‘port’ of type int.
* Scanner is initialised and the user entered string is stored under name.
* Multicast socket is initialised with port.
* Clients are added using joingroup().
* A new thread is initialised with start() method.
* In a while loop, the messages are read and stored in message string.
* If the string equals exit, leavegroup() is initialised and the socket is closed.
* Else, the string is converted into byte array and is stored in buffer for sending.
* Datagram packet is initialised with buffer, buffer length,inetaddress,port number.
* The datagram packet is sent to the port number followed by catch block and the class is closed.
* A new class readthread implementing runnable is created.
* A constructor is initialised with socket,inetaddress object and port.
* The values are inherited from server class using this keyword.
* Run() Is initialised.
* In while loop with condition finished=true, the datagram packet is initialised for receiving datagram from the user, a string is initialised.
* In a try block,the datagram is received and converted into string from byte array by string() .
* In an if block ,if the string!= name from groupchat class ,it is printed.
* Catch block is initialised and exceptions are handled.
* End

|  |
| --- |
| **CODING:** |

1. **Chat using Client/Server Mode**

**CLIENT:**

import java.net.\*;

import java.io.\*;

public class NormalChatClient

{

public static void main(String args[])

{

try

{

Socket client = new Socket("127.0.0.1",8888);

System.out.println("Connecting..");

System.out.println("Entered Waiting room");

System.out.println("Queue:100+");

System.out.println("Queue:1");

System.out.println("Connected.");

System.out.println("Start Typing");

DataInputStream in=new DataInputStream(client.getInputStream());

DataOutputStream out=new DataOutputStream(client.getOutputStream());

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

String clientmessage= "",servermessage="";

while(!clientmessage.equals("bye"))

{

clientmessage=br.readLine();

out.writeUTF(clientmessage);

out.flush();

servermessage=in.readUTF();

System.out.println(servermessage);

}

client.close();

in.close();

out.close();

}

catch(Exception e)

{

System.out.println("Connection lost");

}

}

}

**SERVER:**

import java.net.\*;

import java.io.\*;

public class NormalChatServer

{

public static void main(String args[])

{

try

{

ServerSocket server = new ServerSocket(8888);

Socket client=server.accept();

System.out.println("Connected");

DataInputStream in=new DataInputStream(client.getInputStream());

DataOutputStream out=new DataOutputStream(client.getOutputStream());

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

String clientmessage="",servermessage="";

while(!clientmessage.equals("bye"))

{

clientmessage=in.readUTF();

System.out.println(clientmessage);

servermessage=br.readLine();

out.writeUTF(servermessage);

}

out.close();

in.close();

server.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**2.Group Chat**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class GroupChat

{

private static final String TERMINATE = "Exit";

static String name;

static volatile boolean finished = false;

public static void main(String[] args)

{

try

{

InetAddress group = InetAddress.getByName("224.45.0.2");

int port = 8888;

Scanner sc = new Scanner(System.in);

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

name = sc.nextLine();

MulticastSocket socket = new MulticastSocket(port);

socket.joinGroup(group);

Thread t = new Thread(new ReadThread(socket,group,port));

t.start();

System.out.println("Start typing messages...\n");

while(true)

{

String message;

message = sc.nextLine();

if(message.equalsIgnoreCase(GroupChat.TERMINATE))

{

finished = true;

socket.leaveGroup(group);

socket.close();

break;

}

message = name + ": " + message;

byte[] buffer = message.getBytes();

DatagramPacket datagram = new DatagramPacket(buffer,buffer.length,group,port);

socket.send(datagram);

}

}

catch(SocketException se)

{

System.out.println("Error creating socket");

se.printStackTrace();

}

catch(IOException ie)

{

System.out.println("Error reading/writing from/to socket");

ie.printStackTrace();

}

}

}

class ReadThread implements Runnable

{

private MulticastSocket socket;

private InetAddress group;

private int port;

private static final int MAX\_LEN = 1000;

ReadThread(MulticastSocket socket,InetAddress group,int port)

{

this.socket = socket;

this.group = group;

this.port = port;

}

@Override

public void run()

{

while(!GroupChat.finished)

{

byte[] buffer = new byte[ReadThread.MAX\_LEN];

DatagramPacket datagram = new DatagramPacket(buffer,buffer.length,group,port);

String message;

try

{

socket.receive(datagram);

message = new String(buffer,0,datagram.getLength(),"UTF-8");

if(!message.startsWith(GroupChat.name))

System.out.println(message);

}

catch(IOException e)

{

System.out.println("Socket closed!");

}

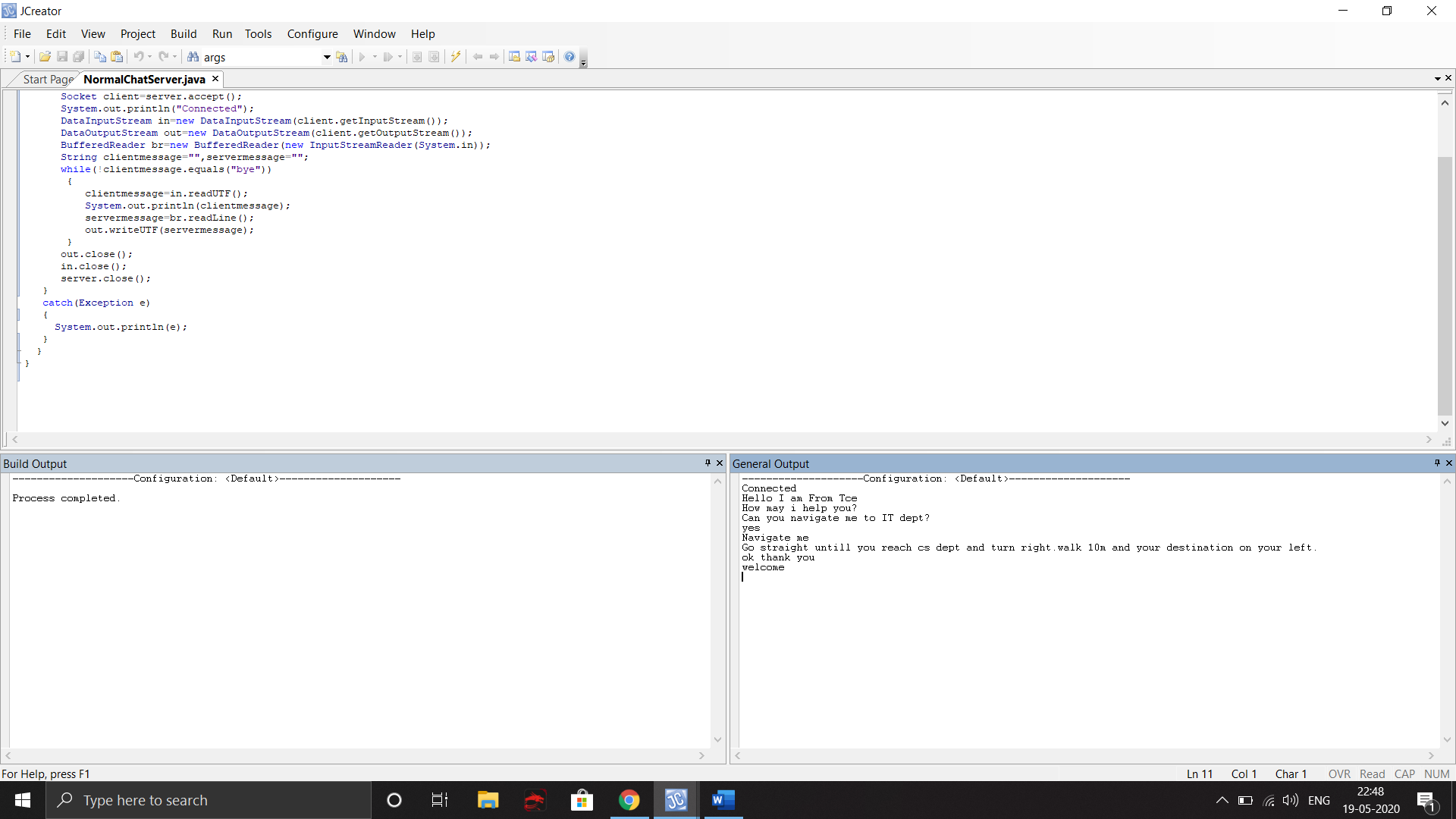
}

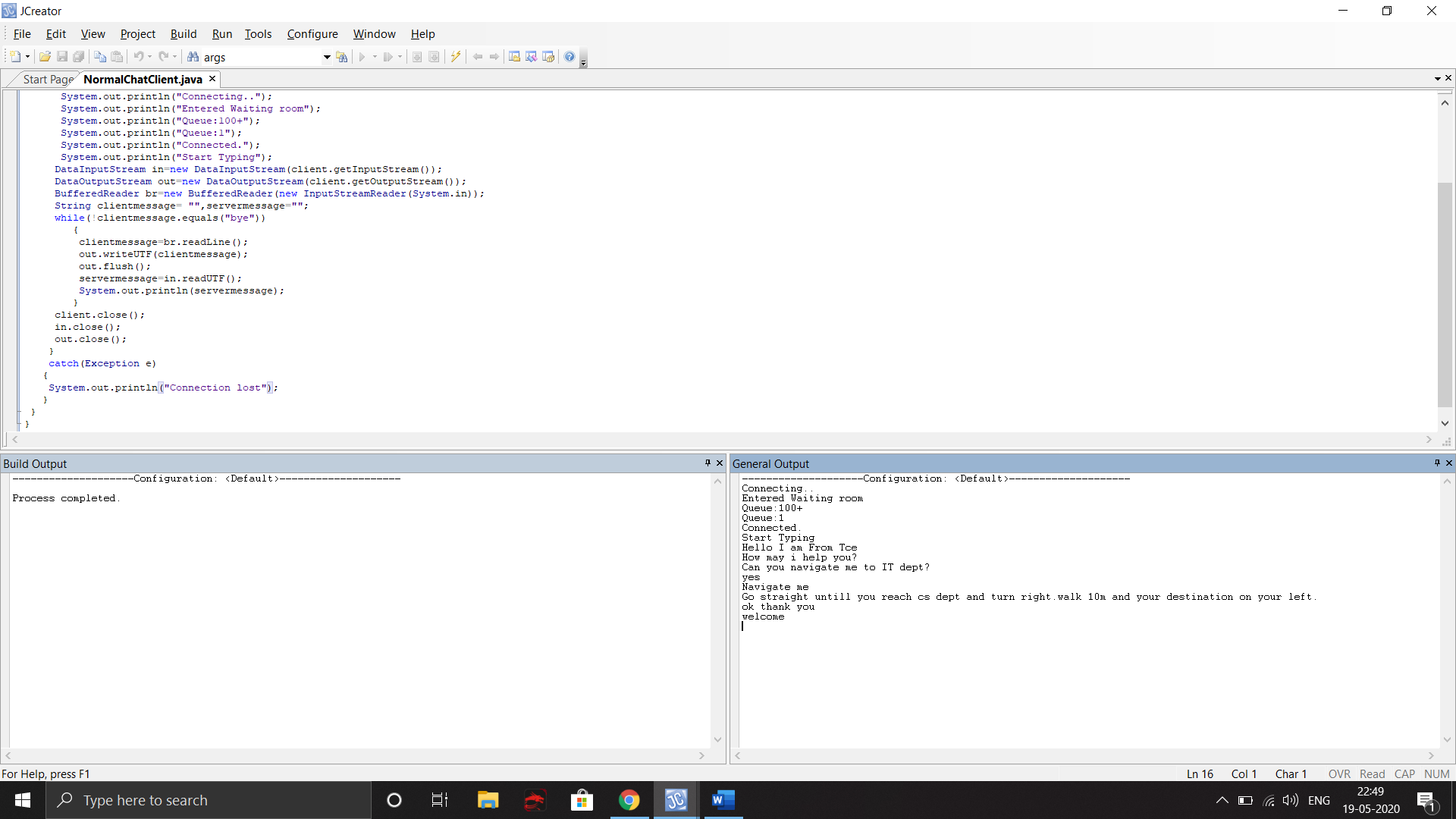
}

}

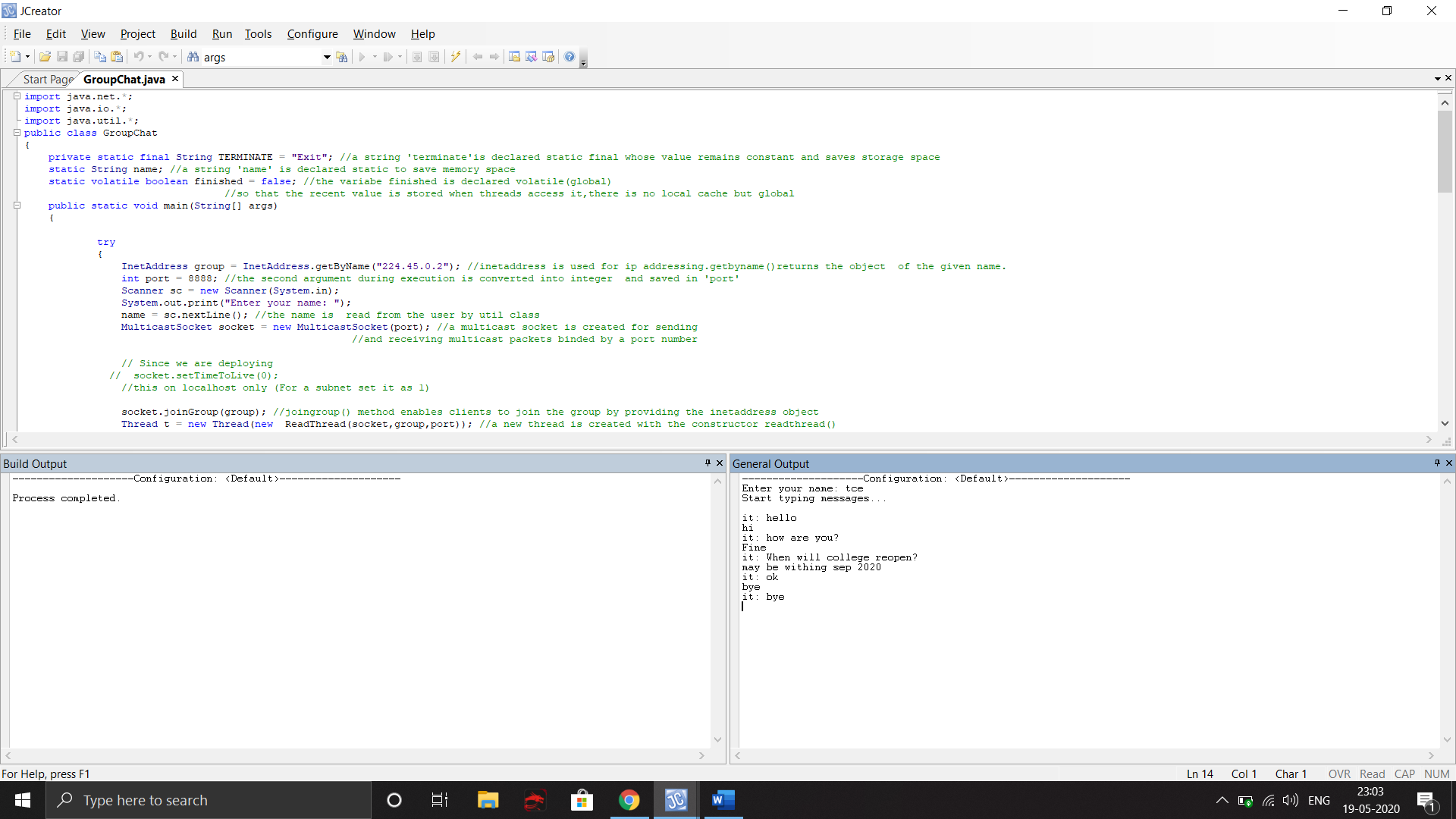
|  |
| --- |
| **SCREEN SHOTS:** |

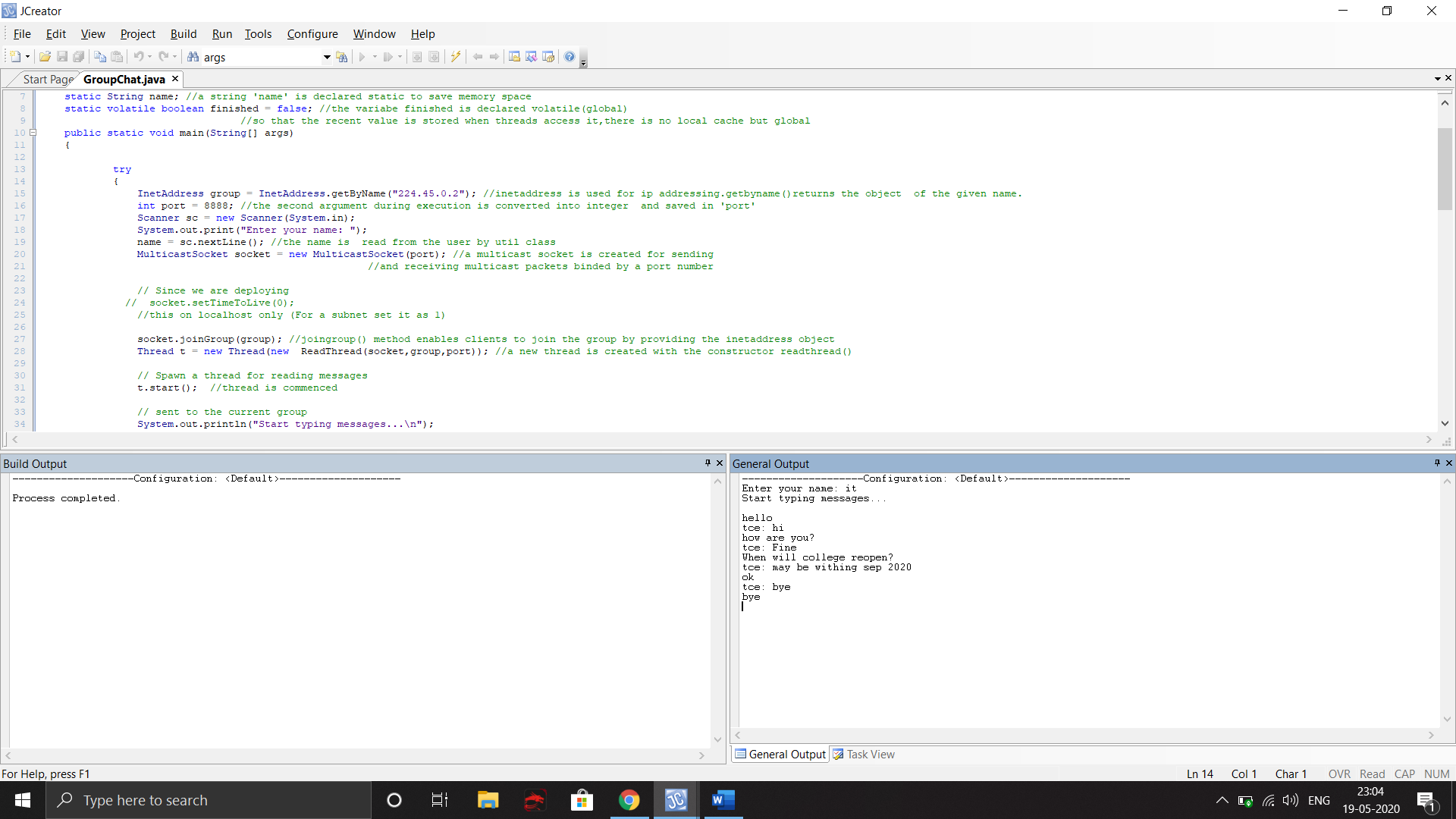
1. **Chat using Client/Server Mode**





**2.Group Chat**





|  |
| --- |
| **RESULT:** |

Thus, the programs for Group chat and Chat in Client/Server Modeare implemented in Java and the results are verified.

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Parameter | Max Marks | Marks Obtained |
| Comparison between Chat in Client /Server mode and Group chat | 10 |  |
| Uniqueness in Coding | 10 |  |
| Sub Total | 20 |  |
| Completion of experiment on time | 3 |  |
| Documentation | 7 |  |
| Sub Total | 10 |  |
| Signature of the faculty with Date |  |  |