**Practical - 4**

**Aim: Demonstrate the group communication through sockets.**

**Code:**

***GroupCommunication.java***

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class GroupCommunication {

private static final String TERMINATE = "exit";

static String name;

static volatile boolean finished = false;

public static void main(String args[]) {

try {

int port = 1234;

InetAddress multicast\_addr = InetAddress.getByName("239.0.0.0");

try (Scanner sc = new Scanner(System.in)) {

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

name = sc.nextLine();

// Create multicast socket and join multicast address

MulticastSocket socket = new MulticastSocket(port);

// Set TTL to 0 for multicast packets

socket.setTimeToLive(0);

// Include multicast address in socket

socket.joinGroup(multicast\_addr);

// Create new thread for sending datagram

Thread t = new Thread(new ReadThread(socket, multicast\_addr, port));

// Spawn a thread for reading messages

t.start();

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

// Loop until "Exit" message is passed

while (!finished) {

String message;

message = sc.nextLine();

// Check to quit socket if "Exit" message is passed

if (message.equalsIgnoreCase(GroupCommunication.TERMINATE)) {

finished = true;

socket.leaveGroup(multicast\_addr);

socket.close();

break;

}

// Pack the message into a datagram

message = name + ": " + message;

byte[] buffer = message.getBytes();

DatagramPacket datagram = new DatagramPacket(buffer, buffer.length, multicast\_addr, port);

// Send datagram over network

socket.send(datagram);

}

}

}

// Catch socket exception

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 (!GroupCommunication.finished) {

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

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

String message = null;

try {

socket.receive(datagram);

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

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

System.out.println(message);

} catch (IOException e) {

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

}

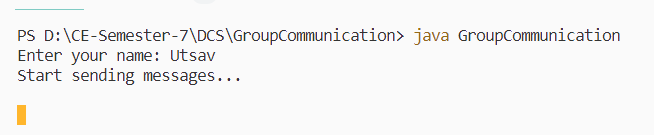
}

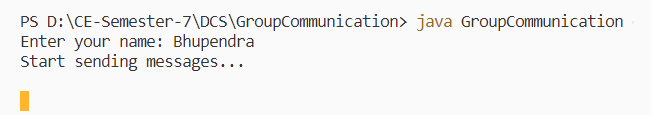
}

}

**Output:**

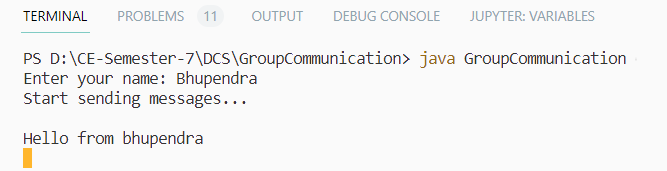
**Three clients in group communication**

****

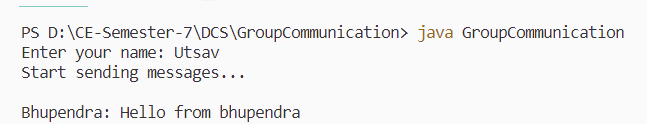
****

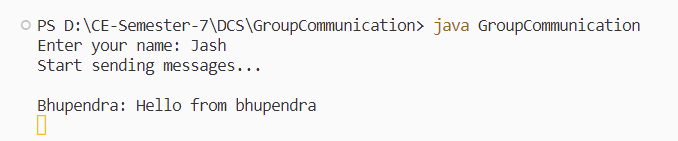
****

**Hello message sent by Bhupendra**

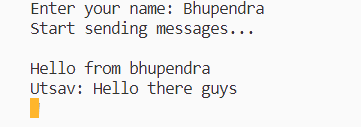
****

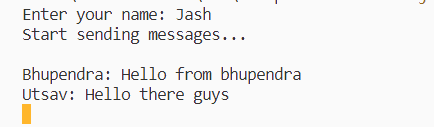
**Received by utsav and jash**

****

****

**Hello from utsav**

****

****