**CODE:**

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)

{

if (args.length != 2)

System.out.println("Two arguments required: <multicast-host> <port-number>"); else

{

try

{

InetAddress group = InetAddress.getByName(args[0]); //gettings var

args

int port = Integer.parseInt(args[1]); //getting port number from

second argument passed

Scanner sc = new Scanner(System.in); System.out.print("Enter your name: "); name = sc.nextLine(); //storing user's name

//Creating MultiCastSocket on port number passed in MulticastSocket socket = new MulticastSocket(port);

// Since we are deploying socket.setTimeToLive(0);

//this on localhost only (For a subnet set it as 1) socket.joinGroup(group);

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

// Spawn a thread for reading messages

t.start();

// sent to the current group 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;

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!");

}

}

}

}

**OUTPUT:**

