A) P2P program

1. Server.java

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
public class Server
{
        private ServerSocket severSocket = null;
        private Socket socket = null;
        private InputStream inStream = null;
        private OutputStream outStream = null;
        public Server()
        {
        }
        public void createSocket() {
        try {
        ServerSocket serverSocket = new ServerSocket(3339);
        while (true) {
        socket = serverSocket.accept();
        inStream = socket.getInputStream();
        outStream = socket.getOutputStream();
        System.out.println("Connected");
        createReadThread();
        createWriteThread();
        }
        } catch (IOException io) {
        io.printStackTrace();
        }
        }
        public void createReadThread() {
        Thread readThread = new Thread() {
        public void run() {
        while (socket.isConnected()) {
        try {
        byte[] readBuffer = new byte[200];
        int num = inStream.read(readBuffer);
        if (num > 0) {
        byte[] arrayBytes = new byte[num];
        System.arraycopy(readBuffer, 0, arrayBytes, 0, num);
        String recvedMessage = new String(arrayBytes, "UTF-8");
        System.out.println("Received message :" + recvedMessage);
        } else {
        notify();
        }
        ;
        //System.arraycopy();
        } catch (SocketException se) {
        System.exit(0);
        } catch (IOException i) {
        i.printStackTrace();
```

```
            }
            }
            }
            };
            readThread.setPriority(Thread.MAX_PRIORITY);
            readThread.start();
            }
            public void createWriteThread() {
            Thread writeThread = new Thread() {
            public void run() {
            while (socket.isConnected()) {
            try {
            BufferedReader inputReader = new BufferedReader(new InputStreamReader(System.in));
            sleep(100);
            String typedMessage = inputReader.readLine();
            if (typedMessage != null && typedMessage.length() > 0) {
            synchronized (socket) {
            outStream.write(typedMessage.getBytes("UTF-8"));
            sleep(100);
            }
            }/* else {
            notify();
            }*/
            ;
            //System.arraycopy();
            } catch (IOException i) {
            i.printStackTrace();
            } catch (InterruptedException ie) {
            ie.printStackTrace();
            }
            }
            }
            };
            writeThread.setPriority(Thread.MAX_PRIORITY);
            writeThread.start();
            }
            public static void main(String[] args) {
            Server chatServer = new Server();
            chatServer.createSocket();
            }
}
```

2. Client.java

```
import java.io.*;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;
public class Client {
private Socket socket = null;
private InputStream inStream = null;
private OutputStream outStream = null;
```

```
public Client() {
}
public void createSocket() {
try {
socket = new Socket("localHost", 3339);
System.out.println("Connected");
inStream = socket.getInputStream();
outStream = socket.getOutputStream();
createReadThread();
createWriteThread();
} catch (UnknownHostException u) {
u.printStackTrace();
} catch (IOException io) {
io.printStackTrace();
}
}
public void createReadThread() {
Thread readThread = new Thread() {
public void run() {
while (socket.isConnected()) {
try {
byte[] readBuffer = new byte[200];
int num = inStream.read(readBuffer);
if (num > 0) {
byte[] arrayBytes = new byte[num];
System.arraycopy(readBuffer, 0, arrayBytes, 0, num);
String recvedMessage = new String(arrayBytes, "UTF-8");
System.out.println("Received message :" + recvedMessage);
}/* else {
// notify();
}*/
;
//System.arraycopy();
}catch (SocketException se){
System.exit(0);
} catch (IOException i) {
i.printStackTrace();
}
}
}
};
readThread.setPriority(Thread.MAX_PRIORITY);
readThread.start();
}
public void createWriteThread() {
Thread writeThread = new Thread() {
public void run() {
while (socket.isConnected()) {
try {
BufferedReader inputReader = new BufferedReader(new InputStreamReader(System.in));
sleep(100);
String typedMessage = inputReader.readLine();
if (typedMessage != null && typedMessage.length() > 0) {
```

```
synchronized (socket) {
outStream.write(typedMessage.getBytes("UTF-8"));
sleep(100);
}
}
;
//System.arraycopy();
} catch (IOException i) {
i.printStackTrace();
} catch (InterruptedException ie) {
ie.printStackTrace();
}
}
}
};
writeThread.setPriority(Thread.MAX_PRIORITY);
writeThread.start();
}
public static void main(String[] args) throws Exception {
Client myChatClient = new Client();
myChatClient.createSocket();
/*myChatClient.createReadThread();
myChatClient.createWriteThread();*/
}
}
```

B) Multiuser chat (in python)

1. server.py

```
import socket
import threading

# Connection Data
host = '127.0.0.1'
port = 55555

# Starting Server
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((host, port))
server.listen()

# Lists For Clients and Their Nicknames
clients = []
nicknames = []

# Sending Messages To All Connected Clients
def broadcast(message):
    for client in clients:
        client.send(message)

# Handling Messages From Clients
```

```
def handle(client):
    while True:
        try:
            # Broadcasting Messages
            message = client.recv(1024)
            broadcast(message)
        except:
            # Removing And Closing Clients
            index = clients.index(client)
            clients.remove(client)
            client.close()
            nickname = nicknames[index]
            broadcast('{} left!'.format(nickname).encode('ascii'))
            nicknames.remove(nickname)
            break

# Receiving / Listening Function
def receive():
    while True:
        # Accept Connection
        client, address = server.accept()
        print("Connected with {}".format(str(address)))

        # Request And Store Nickname
        client.send('NICK'.encode('ascii'))
        nickname = client.recv(1024).decode('ascii')
        nicknames.append(nickname)
        clients.append(client)

        # Print And Broadcast Nickname
        print("Nickname is {}".format(nickname))
        broadcast("{} joined!".format(nickname).encode('ascii'))
        client.send('Connected to server!'.encode('ascii'))

        # Start Handling Thread For Client
        thread = threading.Thread(target=handle, args=(client,))
        thread.start()

receive()
```

2. client.py

```
import socket
import threading

# Choosing Nickname
nickname = input("Choose your nickname: ")

# Connecting To Server
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 55555))
```

```
# Listening to Server and Sending Nickname
def receive():
    while True:
        try:
            # Receive Message From Server
            # If 'NICK' Send Nickname
            message = client.recv(1024).decode('ascii')
            if message == 'NICK':
                client.send(nickname.encode('ascii'))
            else:
                print(message)
        except:
            # Close Connection When Error
            print("An error occured!")
            client.close()
            break

# Sending Messages To Server
def write():
    while True:
        message = '{}: {}'.format(nickname, input(''))
        client.send(message.encode('ascii'))

# Starting Threads For Listening And Writing
receive_thread = threading.Thread(target=receive)
receive_thread.start()

write_thread = threading.Thread(target=write)
write_thread.start()
```