ㅣ

Park, Doyoung (201835453 박도영)

# 과제: [Socket Programming 2]

* 해당 프로그램에 주석 (comment)를 한글로 단다. (추가할 수 있는 것은 최대한 추가)

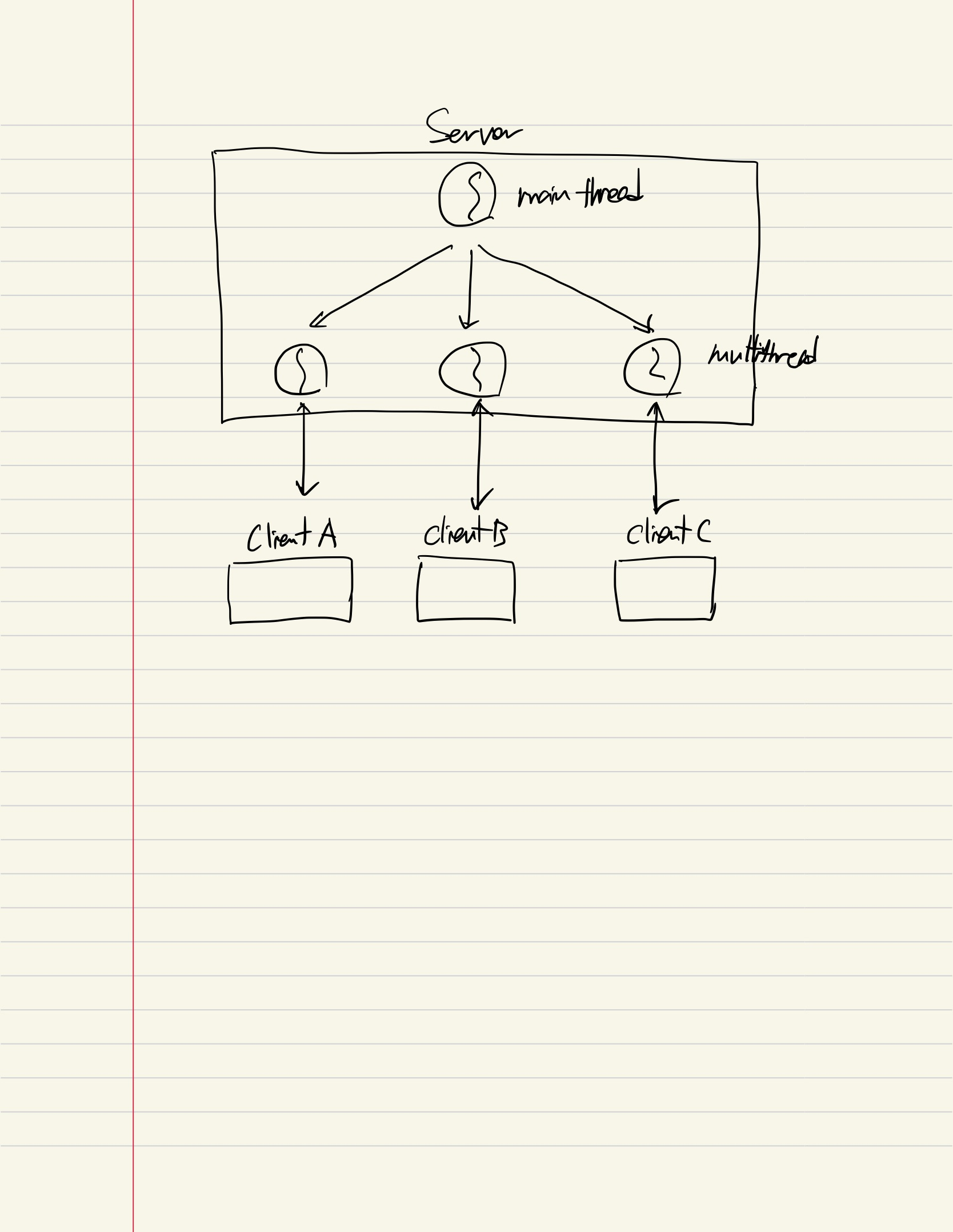
# #개요

|  |  |
| --- | --- |
|  | This is a Chatting program. Using multithread, many people can use this program simultaneously.  They can use a ‘secret message’ function. |

**Requirements**

* < 요구조건 #1: 간략한 핵심 요구 조건 설명>
* 내 코드를 추가하고 주석 달기
* 서버는 동시에 여러명의 클라이언트를 다루어야 함
* 서버는 방 정보를 다루고 모든 클라이언트의 대화를 방에 있는 모두에게 전달함
* 귓속말 기능을 통해서 원하는 사람에게만 메시지를 보내는 기능 추가

# REPORT

* 과제의 기본 요구 사항을 이곳에 원하는 만큼 작성 (e.g. 구조도, protocol 내용 상세 설명 등)
* 소스코드는 마지막에 제공하는 양식을 이용하여 제출 (zip파일로도 제출)
* 
* 서버를 먼저 실행해야 함.
* 서버 실행시킨 후 클라이언트 실행
* 콘솔창에 사용자의 이름 입력하기
* 원하는 클라이언트 수만큼 클라이언트 파일 실행
* 채팅창 오른쪽에는 참여중인 클라이언트 목록이 있음
* 전체 메시지 보내기를 하려면 하단의 입력창에 메시지 입력 후 엔터키 누름
* 귓속말 기능을 사용하려면 오른쪽 목록에서 메시지를 받아야 하는 사람을 마우스 우클릭 후 secret msg 버튼 마우스로 클릭. 그 다음에 나오는 입력창에 메시지 입력 후 send 버튼 누르기
* 채팅방에서 클라이언트가 나가고 싶으면 오른쪽 하단의 exit버튼 누름

## SOURCE CODES : Client file

|  |
| --- |
| package test; import java.io.\*; import java.net.\*; import java.awt.\*; import java.awt.event.\*; import java.util.Scanner;  import org.w3c.dom.events.MouseEvent; import javax.swing.\*;  class Client extends Frame implements ActionListener,MouseListener {  TextField tf,tf2;  TextArea ta;  Button b,b2;  List list;  Dialog dia;  BufferedReader br;  BufferedWriter bw;  PopupMenu pm;  String id;  MenuItem mi;   Client(String id) throws Exception {   super(id+"'s chatting"); // 채팅장 제목  this.id = id;  mi = new MenuItem("secret msg"); // 비밀채팅 버튼  pm = new PopupMenu(); // 우클릭 설정  pm.add(mi); // 우클릭시 비밀채팅버튼 생성  dia = new Dialog(this); // 대화창  tf = new TextField(15); // 메세지 보낸 텍스트필드  ta = new TextArea(); // 텍스트창  b = new Button("exit"); // 종료 버튼  b2 = new Button("send"); //보내기 버튼  list = new List();  tf2=new TextField(); // 비밀채팅 보내는 텍스트 필드  list.add(pm);  Panel p1 = new Panel(); // panel1 생성  Panel p2 = new Panel(); // panel 2 생성   p1.setLayout(new BorderLayout()); // panel1 레이아웃 생성  p2.setLayout(new BorderLayout()); // panel2 레이아웃 생성   p1.add(ta);  p1.add(list, "East");  p2.add(tf);  p2.add(b, "East");  add(p1, "Center");  add(p2, "South");   list.addMouseListener(this);  tf.addActionListener(this);  b.addActionListener(this);  b2.addActionListener(this);  mi.addActionListener(this);  }     void initNet() throws Exception {   Socket socket = new Socket("localhost", 8877); // 해당 컴퓨터 ip주소에 포트번호 8877로 소켓 생성  InputStream is = socket.getInputStream();  InputStreamReader isr = new InputStreamReader(is);  br = new BufferedReader(isr); // 소켓을 통한 인풋 스트림 생성   OutputStream os = socket.getOutputStream();  OutputStreamWriter osw = new OutputStreamWriter(os);  bw = new BufferedWriter(osw); // 소켓을 통한 아웃풋 스트림 생성   sendMsg("enter/" + id); // 새로운 사용자 들어온 경우  }  public void readMsg() {  try {  while (true) {   String line = br.readLine();  System.*out*.println(line);  String array[] = line.split("/");  switch (array[0]) {   case "enter": // 새로운 사용자 들어온 경우  ta.append(array[1] + " entered into this room.\n");  break;   case "msg": // 전체 전송 메세지  ta.append(array[1] + "\n");  break;   case "guestlist":  list.removeAll();  int len = array.length;  for (int i = 1; i < len; i++)  list.add(array[i]);  break;   case "귓속말": // 비밀 채팅 보내는 경우   String arr[] = array[1].split("&");  ta.append("["+arr[0]+"->"+arr[1]+"]"+arr[2] + "\n");  break;     }   }   } catch (Exception e) { // 파일 읽다가 에러  System.*out*.println("Error during reading message");  }  }     public void sendMsg(String msg) throws Exception { // 입력한 메세지 outstream으로 보냄  bw.write(msg + "\n");  bw.flush();  }   public void actionPerformed(ActionEvent e) {   if (e.getSource() == tf) { //텍스트필드로 메세지 보낸 경우  try {  sendMsg("msg/" + tf.getText()); // 전체 메세지 전송  tf.setText("");   } catch (Exception ee) { // 메세지 보내다가 에러난 경우우  System.*out*.println("Error during sending message");  }   } else if (e.getActionCommand().equals("exit")) { //exit 버튼을 누른 경우, 사용자 나감   try {  System.*exit*(0); // 채팅 종료, 사용자 나감   } catch (Exception e1) { // exit 하다가 에러난 경우  e1.printStackTrace();  }  }  else if(e.getSource()==b2){ // secret msg 버튼 누른 경우, 비밀 채팅 보냄   try {  ta.append("["+id+"->"+list.getSelectedItem()+"]"+tf2.getText() + "\n");  sendMsg("귓속말/"+id+"&"+list.getSelectedItem()+"&"+tf2.getText());  dia.setVisible(false);  dia.dispose();   } catch (Exception e1) { // 비밀 채팅 보내다가 에러난 경우  e1.printStackTrace();  }  }   else if(e.getSource() == mi){ // secret 메세지를 보내고자 하는 경우   dia.add(tf2);  dia.add(b2); // 버튼 추가   dia.setLayout(new GridLayout(2,0)); // layout  dia.setBounds(300, 300, 200, 200);  dia.setVisible(true); // 화면 보여줌  }  }   public void mousePressed(java.awt.event.MouseEvent e) { // 마우스 이벤트 설정   if(e.getButton()==3 && list.getSelectedItem() != null ){  pm.show(list, e.getX(), e.getY());  }   }  @Override  public void mouseClicked(java.awt.event.MouseEvent e) {  }   @Override  public void mouseEntered(java.awt.event.MouseEvent e) {  }  @Override  public void mouseExited(java.awt.event.MouseEvent e) {  }   @Override  public void mouseReleased(java.awt.event.MouseEvent e) {  }     public static void main(String args[]) throws Exception {  System.*out*.print("Type a user name:");  Scanner scanner= new Scanner(System.*in*);   Client client = new Client(scanner.nextLine()); // 클라이언트 소켓 생성   client.initNet(); // 클라이언트 소켓 초기화   client.setBounds(200, 200, 500, 400);  client.setVisible(true); // 화면에 채팅창 생성   client.readMsg(); // 메세지 읽음  }  } |

## SOURCE CODES #2: Server file

|  |
| --- |
| package test; import java.net.ServerSocket; import java.net.Socket; import java.util.ArrayList; import java.io.\*; import java.net.\*; import java.util.\*;  class Server {  ArrayList<Guest> list;   void initNet() throws Exception {  list = new ArrayList<Guest>(); // 사용자 list   ServerSocket ss = new ServerSocket(8877); // 포트번호 9977 로 소켓 열기   while (true) {  Socket s = ss.accept(); // 사용자의 접속을 입력 받음  Guest g = new Guest(this, s); // 접속한 사용자의 소켓을 서버에 만듦   g.start();  addGuest(g); // 접속한 사용자를 접속자 목록에 추가  }  }   void addGuest(Guest g) { // 접속한 사용자를 접속자 목록에 추가  list.add(g);  System.*out*.println("number of Users:" + list.size()); // 접속한 사용자를 접속자 수 출력  }   public void talkMsg(String talk, String talk2, String talk3) { // 메세지 보내기  //talk : 보낸 사람  //talk2 : 메세지를 받은 사람  //talk3 : 메세지 내용   for (Guest g : list) { // 전체 접속자에게 메세지 전송   if(g.id.equals(talk2)){ // 사용자에게 귓속말 보내는 경우  try {  g.sendMsg("귓속말/"+talk+"&"+talk2+"&"+talk3);  } catch (Exception e) {  System.*out*.println("during sending a secret message"+e.getMessage());  }  }  }  }   void removeGuest(Guest g) { // 접속자가 나가는 경우   list.remove(g); // 접속자 목록에서 지움  System.*out*.println("접속자수:" + list.size()); // 접속자 수 1감소   }     void broadcast(String msg) throws Exception { // 전체 사용자에게 보내는 메세지   for (Guest g : list) {  g.sendMsg(msg);  }  }     void makeGuestlist() throws Exception { // --> 예시: guestlist/이민갑/김갑동/김장동/   StringBuffer buffer = new StringBuffer("guestlist/");   for (Guest g : list) {  buffer.append(g.id + "/"); // 접속자 리스트를 만듦  }  broadcast(buffer.toString()); // 접속자 리스트를 콘솔창에 출력  }     public static void main(String args[]) throws Exception {   Server server = new Server(); // 서버 객체 실행   server.initNet(); // 서버 소켓 초기화   }  }     class Guest extends Thread {   String id;  Server server;  Socket socket;  BufferedReader br;  BufferedWriter bw;    Guest(Server server, Socket socket) throws Exception { // 서버에 새로 접속한 유저의 소켓 만듦   this.server = server;  this.socket = socket;   InputStream is = socket.getInputStream();  InputStreamReader isr = new InputStreamReader(is);  br = new BufferedReader(isr); // input 스트림 생성   OutputStream os = socket.getOutputStream();  OutputStreamWriter osw = new OutputStreamWriter(os);  bw = new BufferedWriter(osw); // output 스트림 생성  }     public void run() { // 실행   try {   while (true) {  String line = br.readLine();  System.*out*.println(line+"is read.");   String array[] = line.split("/"); // 유저가 보낸 메세지 분할 --> 케이스별로 다르게 인식   switch (array[0]) {   case "enter": // 사용자 들어온 경우  id = array[1];  server.makeGuestlist(); // 사용자 리스트에 추가  server.broadcast(line); // 들어왔다는 알림  break;   case "msg": // 전체 전송 메세지  String str = "msg/[" + id + "]" + array[1];  server.broadcast(str);  break;     case "귓속말": // 귓속말로 보낸 경우  String[] talk=array[1].split("&");  server.talkMsg(talk[0],talk[1],talk[2]);   //talk[0] : 보내는 사람  //talk[1] : 받는 사람  //talk[1] : 보내는 메세지   }  }  } catch (Exception e) { // 예외발생한 경우, 누구 때문인지 알려줌  // e.printStackTrace();  System.*out*.println(e.getMessage()+"get message");  System.*out*.println("Error because of "+ id + "'s message");    //server.removeGuest(this);   try {  server.broadcast("exit/" + id +"is exited"); // 사용자 나간 경우 알려줌   } catch (Exception e1) { // 에러난 경우  e1.printStackTrace();  }  }  }   public void sendMsg(String msg) throws Exception {   bw.write(msg + "\n"); // 보낸 메세지를 버퍼에 올림  bw.flush(); // 버퍼에 올린 메세지를 스트림을 통해 전송  } } |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [설명] park 이 방에 들어온 상황 |
|  | |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [설명] kim 이 방에 들어온 상황 |
|  | |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [설명] Jang 이 방에 들어온 상황 |
|  | |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [설명] Park 이 모두에게 “hello”를 입력했고 Kim과Jang 도 메시지를 보냄 |
|  | |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [설명] Park이 Jang에게 “hi” 라고 귓속말을 보냄 |
|  | |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [설명] Jang이 Park에게 “good” 이라고 귓속말을 보냄 |
|  | |