소켓프로그래밍 과제 2021037156 탁민주

1. Server

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;

// 2021037156 탁민주 소켓 프로그래밍 과제
public class Server {
    // 여러 클라이언트, 즉 여러 소켓을 저장할 수 있는 Socket 형 리스트를 선언한다. ->
멀티쓰레드 환경 구축
    static ArrayList<Socket> list = new ArrayList<Socket>();
    static Socket socket;

    public static void main(String[] args) {
    try {
        //서버가 시작됨
        ServerSocket sv_socket = new ServerSocket(8080);
        System.out.println("서버가 시작되었습니다.");



        while(true) {

            socket = sv_socket.accept();
            //소켓을 서버 소켓이 수락한 소켓으로 대입한다.
            System.out.println("서버에 연결중입니다.");
            list.add(socket);
            //클라이언트가 수락되면 해당 소켓을 list 에 추가 후 쓰레드 진행
            //각각의 클라이언트에 대한 쓰레드
            /*1. 먼저 클라이언트의 최초 메세지(닉네임)을 받아 어떤 클라이언트가
접속하였는지 출력한다.
              2. 그 후에 메시지가 null 이 아닐때까지 반복문을 돌린다. 그동안 해당
메세지를 list 에 있는 모든 클라이언트에게
                출력한다.
              3. 만약 메세지가 'quit'라면  해당 클라이언트가 서버를 떠났음을 서버 및
모든 클라이언트에게 출력한다.
                 그 후에 리스트에서 해당 소켓은 삭제시킨다. */
            new Thread(new Runnable() {
                String UserID;
                @Override
                public void run() {
                    try {

                        String msg;

                        InputStream input = socket.getInputStream();
                        BufferedReader read = new BufferedReader(new
InputStreamReader(input, "UTF-8"));
                        PrintWriter wr = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream(), "UTF-8")), true);
                        UserID = read.readLine(); //소켓의 버퍼를 읽는다.
                        for (int i = 0; i < list.size(); i++) { // 리스트에 있는
소켓의 개수들만큼 접속 문구를 띄운다.
```

```
                        OutputStream out = list.get(i).getOutputStream();
                        PrintWriter writer = new PrintWriter(new
BufferedWriter(new OutputStreamWriter(out, "UTF-8")), true);
                        writer.println(UserID + "님이 접속하셨습니다.");

                    }
                    System.out.println(UserID + "님이 접속하셨습니다."); //해당
서버창에 출력
                    //서버 -> 클라이언트
                    while ((msg = read.readLine()) != null) { //메세지가
null 이 아닐때까지
                        if (msg.equals("quit")) {
                            wr.println(msg); //quit 이면 해당 메세지를 보낸
client 의 창에만 출력
                            break;
                        }

                        System.out.println(msg);
                        for (int i = 0; i < list.size(); i++) {
                            //quit 가 아닐 경우 모든 클라이언트에게 메세지 전송
                            OutputStream out =
list.get(i).getOutputStream();
                            PrintWriter writer = new PrintWriter(new
BufferedWriter(new OutputStreamWriter(out, "UTF-8")), true);
                            writer.println(msg);

                        }
                    }


                } catch (IOException e) {
                    throw new RuntimeException(e);
                } finally {
                    System.out.println(UserID + "님이 방을 나갔습니다.");
                    for (int i = 0; i < list.size(); i++) {
                        OutputStream out;
                        // 만약 나갔을 경우 모든 클라이언트에게 퇴장 문구 전송.
                        try {
                            out = list.get(i).getOutputStream();
                            PrintWriter writer = new PrintWriter(new
BufferedWriter(new OutputStreamWriter(out, "UTF-8")), true);
                            writer.println(UserID + "님이 방을 나갔습니다.");
                        } catch (IOException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }

                }
            }
        }

        ).start();
    }
}
    catch (Exception e) {
        throw new RuntimeException(e);
    }
```

```
    }
}
```

2. Client

```
import java.io.*;
import java.net.*;
import java.util.Scanner;


public class Client {
    static BufferedReader read;
    static PrintWriter prt;


    public static void main(String[] args) throws IOException,
InterruptedException {


        try {
            // 서버 연결
            System.out.println("서버에 접속중입니다.");
            Socket clientSocket = new Socket("localhost", 8080);
            System.out.println("서버와 연결 완료.");
            System.out.print("이름을 입력해주세요: ");
            Scanner sc = new Scanner(System.in);
            String name = sc.nextLine();

            /*보내는 sender 쓰레드
            * 사용자에게 입력을 받아서 해당 메세지를 스트림에 보낸다.
            * 만약 'quit'라면 해당 메세지를 스트림에 올린 후 쓰레드를 끝낸다. */

            Thread s_Thread = new Thread() {
                    @Override
                    public void run() {
                    try {
                        OutputStream out = clientSocket.getOutputStream();
                        PrintWriter writer = new PrintWriter(new
BufferedWriter(new OutputStreamWriter(out, "UTF-8")), true);
                        writer.println(name); //이름을 출력해줌
                        while (true) {
                            String msg = sc.nextLine();
                            if (msg.equals("quit")) {
                                writer.println(msg);
                                break;
                            }
                            writer.println(name + ": " + msg);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }
            };
            s_Thread.start();

            /* 받는 reciever 쓰레드
             * 버퍼에 올려진 메세지를 받아 해당 클라이언트의 채팅창에 출력시킨다.
             * 마찬가지로 메세지가 quit 이면 더이상 메세지를 읽지 않는다.
             */
```

```
            Thread r_Thread = new Thread() {
                @Override
                public void run() {
                    try{
                        InputStream input;//읽는 stream
                        BufferedReader read;// input 내용을 buffer로 받아옴
                        while(true){
                            String msg = null;
                            input = clientSocket.getInputStream();
                            read = new BufferedReader(new
InputStreamReader(input, "UTF-8"));
                            if ((msg = read.readLine())!=null){
                                if (msg.equals("quit")){
                                    break;}

                                System.out.println(msg);

                            }


                        }
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }
            };

            r_Thread.start();

            s_Thread.join();
            r_Thread.join();

            //쓰레드의 종료 순서가 엇갈리지 않도록 join을 통해 종료시킨다.

            clientSocket.close();




        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

<실행화면 및 실행 방법>

1. 서버 실행

- src 파일로 이동 후 Server.java를 컴파일하고 실행시킨다.

서버 시작을 알리는 문구를 출력한다.

```
PS D:\2-2\컴퓨터네트워크\Socket_programming> cd src
PS D:\2-2\컴퓨터네트워크\Socket_programming\src> javac Server.java
PS D:\2-2\컴퓨터네트워크\Socket_programming\src> java Server
서버가 시작되었습니다.
```

2. 클라이언트 접속

- 서버에 접속하여 연결을 하였다는 문구를 출력시킨다.

최초의 메시지로 클라이언트의 이름을 입력하고 접속 문구를 출력한다.

```
PS D:\2-2\컴퓨터네트워크\Socket_programming> cd src
PS D:\2-2\컴퓨터네트워크\Socket_programming\src> javac Client.java
PS D:\2-2\컴퓨터네트워크\Socket_programming\src> java Client
서버에 접속중입니다.
서버와 연결 완료.
이름을 입력해주세요: minju
minju님이 접속하셨습니다.
```

3. 여러명이 접속했을 경우

- 처음 접속한 사람에게는 이후 접속자들의 접속 문구가 뜬다. 또한 서버에서는 모든 접속자들의 접속 문구가 보이게 된다.

```
서버가 시작되었습니다.
서버에 연결중입니다.
minju님이 접속하셨습니다.
서버에 연결중입니다.
juyeon님이 접속하셨습니다.
서버에 연결중입니다.
myjava님이 접속하셨습니다.
서버에 연결중입니다.
cse님이 접속하셨습니다.
```

4. 클라이언트 –(메시지)-> 서버

- 메시지를 보낼 경우 해당 클라이언트 창에서 입력한 문구와 채팅이 함께 보인다.

```
서버에 접속중입니다.
서버와 연결 완료.
이름을 입력해주세요: minju
minju님이 접속하셨습니다.
juyeon님이 접속하셨습니다.
myjava님이 접속하셨습니다.
cse님이 접속하셨습니다.
hi my name is minju
minju: hi my name is minju
```

```
cse님이 접속하셨습니다.
minju: hi my name is minju
juyeon: my Im juyeon
myjava: Im myjava
Im cse
cse: Im cse
```

5. 클라이언트 -/-> 서버

- 'quit'를 보내면 클라이언트와 서버간의 접속이 끊기게 된다.

  이때 모든 클라이언트들에게 해당 클라이언트의 퇴장 여부를 알린다.

```
cse님이 접속하셨습니다.
minju: hi my name is minju
juyeon: my Im juyeon
myjava: Im myjava
Im cse
cse: Im cse
quit
PS D:\2-2\컴퓨터네트워크\Socket_programming\src>
```

```
cse님이 방을 나갔습니다.
myjava님이 방을 나갔습니다.
juyeon님이 방을 나갔습니다.
quit
```

다른 사람들이 나갔을 때 다른 클라이언트에게 보이는 창

6. 모든 클라이언트가 퇴장했을 때

```
PS D:\2-2\컴퓨터네트워크\Socket_programming> cd src
PS D:\2-2\컴퓨터네트워크\Socket_programming\src> javac Server.java
PS D:\2-2\컴퓨터네트워크\Socket_programming\src> java Server
서버가 시작되었습니다.
서버에 연결중입니다.
minju님이 접속하셨습니다.
서버에 연결중입니다.
juyeon님이 접속하셨습니다.
서버에 연결중입니다.
myjava님이 접속하셨습니다.
서버에 연결중입니다.
cse님이 접속하셨습니다.
minju: hi my name is minju
juyeon: my Im juyeon
myjava: Im myjava
cse: Im cse
cse님이 방을 나갔습니다.
myjava님이 방을 나갔습니다.
juyeon님이 방을 나갔습니다.
minju님이 방을 나갔습니다.
```