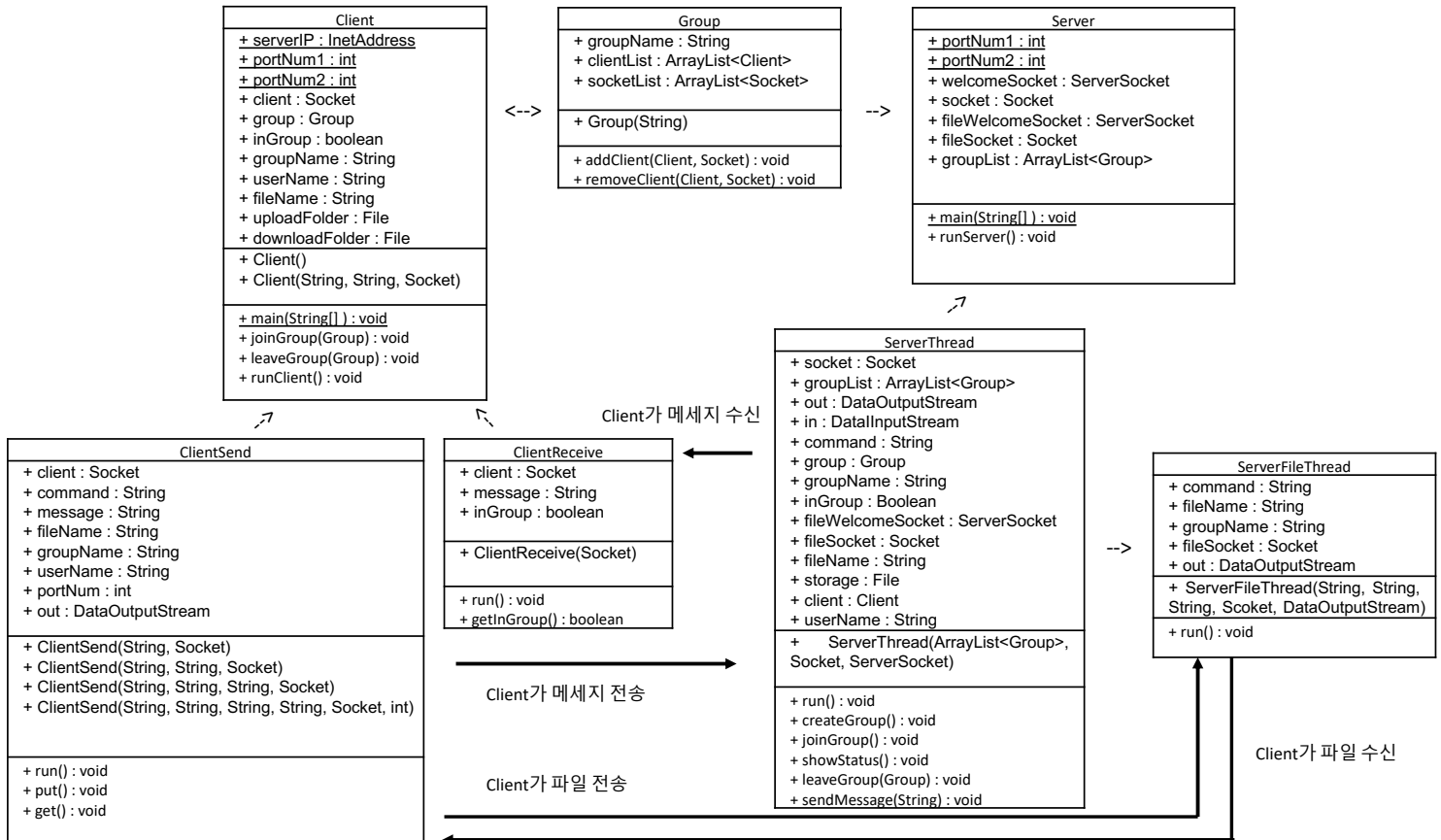


Computer Networks : Programming Assignment #2

컴퓨터소프트웨어학부 20210250205 강태욱

1. 프로그램 구조

a. 프로그램 구조도



b. 프로그램 설명

- 이 프로그램은 TCP 방식의 파일 공유, 단체 채팅 프로그램이다.
- 이 프로그램은 하나의 server 를 가지며, 여러 개의 client 프로그램을 통해 client 끼리 소통할 수 있다. 즉, client 프로그램은 서로 독립된 경로를 가지는 source 폴더를 통해 실행되어야 한다.
- server 에서는 Thread 를 이용해 각각의 client 와 동시에 소통한다. 이때, ServerThread 의 배열 중 하나의 ServerThread 는 하나의 Client 와 소통한다. 즉, 하나의 Client 에 하나의 ServerThread 가 1:1 로 할당되어 server 와 client 를 서로

연결한다. server 프로그램에서는 각각의 ServerThread 가 하는 일을 출력해 server 에서 어떠한 작업이 이루어지고 있는지 확인할 수 있도록 한다.

- iv. client 프로그램을 실행하면 .java file 혹은 .class file 이 위치한 경로에 client_upload, client_download 폴더를 생성한다.
- v. 이 프로그램은 채팅방과 관련하여 “#CREATE”, “#JOIN”, “#STATUS”, “#EXIT”, “#QUIT” 명령어를 가진다.
 - 1. “#CREATE (group name) (client name)” 명령어는 (group name)의 이름을 가지는 group 을 만들고, 사용자 이름 (client name)으로 해당 채팅방에 접속한다는 의미이다. 사용자는 중복된 이름의 group 을 생성할 수 없다.
 - 2. “#JOIN (group name) (client name)” 명령어는 (group name)의 이름을 가지는 group 에 사용자 이름 (client name)으로 접속한다는 의미이다. 사용자는 참가하고자 하는 group 내의 client 들과 중복된 client name 을 가질 수 없다.
 - 3. “#STATUS” 명령어는 client 가 참여하고 있는 group 의 이름과 해당 group 에 참여하고 있는 client 들의 이름을 보여준다.
 - 4. “#EXIT” 명령어는 client 가 참여하고 있는 group 을 떠나기 위해 사용된다.
 - 5. “#QUIT” 명령어는 client program 을 종료하기 위해 사용된다.
- vi. 이 프로그램은 파일 전송 및 수신과 관련하여 “#PUT”, “#GET” 명령어를 가진다.
 - 1. “#PUT (file name)” 명령어는 (file name)을 이름으로 가지는 file 을 client 가 참여하고 있는 group 에게 업로드한다는 의미이다.
 - 2. “#GET (file name)” 명령어는 (file name)을 이름으로 가지는 file 을 client 가 참여하고 있는 group 으로부터 다운로드한다는 의미이다.
- vii. client 는 입력 받은 문자열에 맞는 정보를 thread 와 TCP protocol 을 통해 server 에게 전송한다.
- viii. server 는 client 로부터 받은 정보를 바탕으로 그에 맞는 작업을 수행 후 thread 를 통해 결과 메시지를 client 에게 전달한다.
- ix. client 는 server 로부터 메시지를 전달받는 thread 를 가지며, 이는 server 에게 정보를 전달하는 thread 와 분리되어 동작한다.
- x. client 가 group 을 만들면 서버의 groupList 에 group 이 등록되며, client 에게 group 이 만들어짐을 알린다. 이때, groupList 에 같은 이름을 가지는 group 이 이미 존재하면 server 는 실패 메시지를 전송하고 group 을 등록되지 않는다. client 가 어떠한 group 에 참여하고 있는 상태에서는 새로운 group 을 만들 수 없다.

- xi. client 가 group 에 join 하면 group 의 clientList 에 peer 가 등록되며, 자신을 포함한 group 에 있는 모든 client 에게 해당 client 가 group 에 참여함을 알린다. 같은 이름을 가지는 client 가 group 에 이미 존재하면, server 는 실패 메시지를 전송하고 client 는 등록되지 않는다. client 가 어떠한 group 에 참여하고 있는 상태에서는 새로운 group 에 참여할 수 없다.
- xii. “#EXIT” 혹은 “#QUIT” 명령어를 통해 client 가 group 을 떠나면 자신을 포함한 group 에 있는 모든 client 에게 해당 client 가 group 을 떠났음을 알린다. 만약 group 을 떠난 client 가 해당 group 의 유일한 client 였다면, client 의 탈퇴 후 해당 group 을 groupList 에서 삭제한다. 이후 다른 client 가 삭제된 group 과 동일한 이름을 가지는 group 을 생성할 수 있다.
- xiii. client 가 client_upload 폴더 내의 file 을 put 하면, (group_name)_storage 폴더에 해당 file 이 저장된다. file 의 put 이 완료되면 group 내의 모든 client 에게 해당 file 이 put 되었음을 알린다. 만약 client_upload 폴더 내에 해당 file 이 없다면 실패 메시지를 출력한다.
- xiv. client 가 (group_name)_storage 폴더 내의 file 을 get 하면, client_download 폴더에 해당 file 이 저장되며 file 을 get 한 client 에게만 file 을 수신했다는 메시지를 출력한다. 만약 (group name)_storage 폴더 내에 해당 file 이 없다면 실패 메시지를 출력한다.
- xv. ‘#’로 시작하지 않는 문자열은 채팅을 통해 전달하고자 하는 메시지로, 이 프로그램의 client 들은 thread 를 통해 메시지를 전송하고 수신한다.
- xvi. 올바르지 않은 명령어, group 에 가입하지 않은 상태로 group 을 탈퇴하거나 메시지를 입력하는 등 비정상적 행동에 대해서는 그에 맞는 에러 메시지를 출력한 후, 사용자로부터 다시 입력을 받는다.

2. 코드 상세 설명 – Client Class

a. variables

- i. `public static InetAddress serverIP` : client 와 정보를 주고받을 server 의 IP Address
- ii. `public static int portNum1` : 채팅과 관련된 정보를 주고받을 때 사용하는 port number
- iii. `public static int portNum2` : 파일 전송 및 수신에 사용되는 port number
- iv. `public Socket client` : server 로 message 를 전송하는 socket
- v. `public Group group` : client 가 참여하고 있는 group
- vi. `public Boolean inGroup` : client 의 group 참여 여부를 알려줌
- vii. `public String groupName` : 명령어 인자로 받은 생성/참여하고자 하는 group 의 이름
- viii. `public String userName` : 명령어 인자로 받은 group 에 참여하고자 하는 client 의 이름
- ix. `public String filename` : 명령어 인자로 받은 전송/수신할 file 의 이름
- x. `public File uploadFolder` : server 로 전송할 file 을 담은 폴더
- xi. `public File downloadFolder` : server 로부터 수신할 file 을 담은 폴더

b. Constructors

- i. `public Client ()` : default constructor
- ii. `public Client (String groupName, String userName, Socket socket)` : groupName, userName, socket 을 각각 client 의 groupName, userName, this.client 로 만드는 constructor

c. `public void joinGroup(Group group)`

- i. this.group 을 group 으로 지정해 client 가 속한 group 을 지정해주는 method

d. `public void leaveGroup(Group group)`

- i. this.group 을 null 로 지정해 client 의 group 을 제거하는 method

e. `public static void main(String[] args)` : Client Class 의 main method

- i. 프로그램 실행 명령어의 첫번째 인자를 serverIP, 두번째 인자를 portNum1, 세번째 인자를 portNum2 로 입력받아 저장한다.
- ii. default constructor 를 통해 client 을 만들고 runClient() method 를 실행한다.

f. `public void runClient()` : server 로부터 message 를 수신하는 thread 를 실행하며 message 를 입력받고 그에 맞는 thread 를 실행하는 method

- i. serverIP, portNum1 을 이용해 Socket 을 생성한 후 이를 client 에 할당한다.
- ii. 사용자로부터 문자열을 입력받을 BufferedReader br 을 생성한다.
- iii. server 로부터 정보를 수신할 ClientReceive cr 을 선언하고 이를 실행한다.
- iv. put 할 file 을 담은 client_upload 폴더와 get 한 file 을 담은 client_download 폴더를 생성한다.
- v. vi.부터 xvi.까지의 반복문을 종료 전까지 수행한다.

- vi. command line 으로부터 문자열을 입력받는다.
- vii. 만약 문자열의 길이가 0 이면, 사용자가 어떠한 문자를 입력하지 않은 것이므로 오류 메시지(“----- Please enter contents -----”)를 출력하고 다시 문자열을 입력받는다.
- viii. 만약 입력받은 문자열이 “#CREATE”으로 시작한다면
 - 1. 만약 inGroup 이 true 라면 client 가 어떤 group 에 참여하고 있다는 뜻이므로 오류 메세지 (“----- You must leave current group to create another group -----”)를 출력하고 다시 문자열을 입력받는다.
 - 2. 입력받은 문자열의 2 번째 인자를 groupName, 3 번째 인자를 userName 에 저장한다.
 - 3. “#CREATE”, groupName, userName, client 를 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 group 생성 메세지를 server 에 전송한다.
 - 4. Thread.sleep(100)을 통해 프로그램을 잠시 멈추어 ClientReceive 가 server 로부터 정보를 완전히 전달받을 시간을 제공한다.
 - 5. cr.getInGroup()을 통해 server 로부터 받은 group 생성 여부를 inGroup 에 저장한다.
- ix. 만약 입력받은 문자열이 “#JOIN”시작한다면
 - 1. 만약 inGroup 이 true 라면 client 가 어떤 group 에 참여하고 있다는 뜻이므로 오류 메세지 (“----- You must leave current group to create another group -----”)를 출력하고 다시 문자열을 입력받는다.
 - 2. 입력받은 문자열의 2 번째 인자를 groupName, 3 번째 인자를 userName 에 저장한다.
 - 3. “#JOIN”, groupName, userName, client 를 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 group 참가 메세지를 server 에 전송한다.
 - 4. Thread.sleep(100)을 통해 프로그램을 잠시 멈추어 ClientReceive 가 server 로부터 정보를 완전히 전달받을 시간을 제공한다.
 - 5. cr.getInGroup()을 통해 server 로부터 받은 group 생성 여부를 inGroup 에 저장한다.
- x. 만약 입력받은 문자열이 “#PUT”으로 시작한다면
 - 1. 만약 inGroup 이 false 라면 client 가 어떠한 group 에도 참여하고 있지 않다는 뜻이므로 오류 메세지 (“You are not in any chat room. Try again -----”)을 출력하고 다시 문자열을 입력받는다.

2. 입력받은 문자열의 2 번째 인자를 fileName 에 저장한다.
 3. “#PUT”, fileName, groupName, userName, client, portNum2 를 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 put 관련 메시지를 server 에 전송하고 file 을 업로드한다.
- xi. 만약 입력받은 문자열이 “#GET”으로 시작한다면
1. 만약 inGroup 이 false 라면 client 가 어떠한 group 에도 참여하고 있지 않다는 뜻이므로 오류 메시지 (“You are not in any chat room. Try again -----”)을 출력하고 다시 문자열을 입력받는다.
 2. 입력받은 문자열의 2 번째 인자를 fileName 에 저장한다.
 3. “#GET”, fileName, groupName, userName, client, portNum2 를 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 get 관련 메시지를 server 에 전송하고 file 을 다운로드한다.
- xii. 만약 입력받은 문자열이 “#STATUS” 라면
1. 만약 inGroup 이 false 라면 client 가 어떠한 group 에도 참여하고 있지 않다는 뜻이므로 오류 메시지 (“You are not in any chat room. Try again -----”)을 출력하고 다시 문자열을 입력받는다.
 2. “#STATUS”, client 를 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 status 요청 메시지를 server 에게 전송한다.
- xiii. 만약 입력받은 문자열이 “#EXIT” 라면
1. 만약 inGroup 이 false 라면 client 가 어떠한 group 에도 참여하고 있지 않다는 뜻이므로 오류 메시지 (“You are not in any chat room. Try again -----”)을 출력하고 다시 문자열을 입력받는다.
 2. “#EXIT”, client 를 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 exit 요청 메시지를 server 에게 전송한다.
 3. inGroup 을 false 로 지정해 client 가 어떠한 group 에 참여하고 있지 않다는 상태를 저장한다.
- xiv. 만약 입력받은 문자열이 “#QUIT” 이라면
1. 만약 inGroup 이 false 라면 client 가 어떠한 group 에도 참여하고 있지 않다는 뜻이므로 오류 메시지 (“You are not in any chat room. Try again -----”)을 출력하고 다시 문자열을 입력받는다.
 2. “#QUIT”, client 를 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 exit 요청 메시지를 server 에게 전송한다.

3. ClientRecevie cr 을 null 로 만들어 server 로부터 정보를 더이상 전달받지 않도록 한다.
 4. client_upload 와 client_download 폴더와 그 내용물을 삭제한다.
 5. client 프로그램을 종료한다.
- xv. 만약 입력받은 문자열의 첫 글자가 '#'이지만, "#CREATE", "#JOIN", "#PUT", "#GET", "#STATUS", "#EXIT", "#QUIT" 이외의 다른 문자열이 입력된 경우
1. 사용자가 잘못된 명령어를 입력한 경우이기에 에러 메시지("Not allowed command. Try again -----")를 출력한다.
 2. 사용자로부터 다시 문자열을 입력받는다.
- xvi. 만약 입력받은 문자열이 명령어가 아니고, client 가 group 에 참여중인 경우
1. 입력받은 문자열 앞에 userName + ":" 을 붙여 message 로 저장한다.
 2. "#SEND", message 를 인자로 가지는 ClientSend cs thread 를 생성하고 이를 실행한다.

3. 코드 상세 설명 – ClientSend Class extends Thread

a. variables

- i. public Socket client : server 로 정보를 전송하는데 사용되는 socket
- ii. public String command : Thread 를 통해 server 로 전달할 명령어
- iii. public String message : Thread 를 통해 server 로 전달할 채팅 문장
- iv. public String fileName : Thread 를 통해 server 로 전달할 file 의 이름
- v. public String groupName : Thread 를 통해 server 로 전달할 group 의 이름
- vi. public Sting userName : Thread 를 통해 server 로 전달할 client 의 이름
- vii. public String portNum : file transfer 을 수행할 socket 의 port number
- viii. public DataOutputStream out : server 로 데이터를 전송하는 역할 수행

b. Constructors

- i. ClientSend Class 의 Constructor 은 모두 인자로 받은 요소를 ClientSend Class 의 객체가 가지는 variable 값으로 setting 하는 역할을 수행한다.
- ii. 예를 들어, public ClientSend(String command, Socket client)는 this.client = client, this.command = command 의 역할을 수행한다.

c. public synchronized void run() : Thread 의 기능을 수행하는 method

- i. client.getOutputStream()을 인자로 가지는 DataOutputStream out 을 만든다.
- ii. 만약 command 가 "#CREATE" 혹은 "#JOIN"이면
 1. out.writeUTF(command)를 통해 command 를 server 로 전달한다.

2. out.writeUTF(groupName)을 통해 groupName 를 server 로 전달한다.
3. out.writeUTF(userName)을 통해 userName 를 server 로 전달한다.
- iii. 만약 command 가 “#PUT”이면
 1. file 전송에 관한 역할을 수행하는 put() method 를 실행한다.
- iv. 만약 command 가 “#GET”이면
 1. file 수신에 관한 역할을 수행하는 get() method 를 실행한다.
- v. 만약 command 가 “#SEND”이면
 1. out.writeUTF(command)를 통해 command 를 server 로 전달한다.
 2. out.writeUTF(message)를 통해 message 를 server 로 전달한다.
- vi. 만약 command 가 “#STATUS” 혹은 “#EXIT”이면
 1. out.writeUTF(command)를 통해 command 를 server 로 전달한다.
- vii. 만약 command 가 “#QUIT”이면
 1. out.writeUTF(command)를 통해 command 를 server 로 전달한다.
 2. client 프로그램이 실행을 종료하므로 client.close()를 통해 client socket 을 닫는다.

d. public void put() : file 전송에 관한 역할을 수행하는 method

- i. client_upload 폴더 안의 fileName 을 file 의 이름으로 가지는 file 에 대한 File file 을 생성한다.
- ii. 만약 fileName 을 file 의 이름으로 가지는 file 이 존재하지 않는다면 오류 메세지 (“--- -- (fileName) not exists! -----”)를 출력한다.
- iii. out 을 통해 command 와 fileName 을 server 로 전송한다.
- iv. file 의 정보를 읽는 FileInputStream fis 를 생성한다.
- v. client 의 IP Address 와 portNum 을 이용해 file transfer 를 독자적으로 수행하는 Socket socket 을 생성한다.
- vi. socket 을 통해 데이터를 server 로 전송하는 OutputStream os 를 생성한다.
- vii. 64KB 의 크기를 가지는 버퍼 buffer 를 생성한다.
- viii. 반복문을 통해 한번에 buffer 의 크기만큼 file 을 server 로 전송한다. 반복문이 한번 수행될 때 마다 ‘#’을 출력하여 64KB 씩 file 이 전송됨을 나타낸다.
- ix. fis, os, socket 을 close 한다.

e. public void get() : file 수신에 관한 역할을 수행하는 method

- i. out 을 통해 command 와 fileName 을 server 로 전송한다.

- ii. client 의 IP Address 와 portNum 을 이용해 file transfer 를 독자적으로 수행하는 Socket socket 을 생성한다.
- iii. socket 과 연결된 server 로부터 데이터를 수신하는 InputStream is 를 생성한다.
- iv. client_download 폴더 안에 fileName 을 file 의 이름으로 가지는 file 에 대한 File file 을 생성한다.
- v. file 의 정보를 disk 에 쓰는 FileOutputStream fo 를 생성한다.
- vi. 64KB 의 크기를 가지는 버퍼 buffer 를 생성한다.
- vii. 반복문을 통해 한번에 buffer 의 크기만큼 file 을 server 로부터 수신한다. 반복문이 한번 수행될 때 마다 '#'을 출력하여 64KB 씩 file 이 수신됨을 나타낸다.
- viii. fis, os, socket 을 close 한다.

4. 코드 상세 설명 – ClientReceive Class extends Thread

a. variables

- i. public Socket client : server 로부터 정보를 전달받을 client socket
- ii. public String message : server 로부터 전달받은 문자열
- iii. public Boolean inGroup : client 의 group 소속 여부를 저장

b. Constructors

- i. public ClientReceive (Socket client) : ClientReceive Class 객체의 client 를 인자로 전달받은 client 로 설정한다.

c. public void run() : Thread 의 기능을 수행하는 method

- i. client 로 전달된 데이터를 읽는 DataInputStream in 을 생성한다.
- ii. iii.부터 vi.까지의 반복문을 수행한다.
- iii. server 로부터 수신된 문자열을 message 에 저장한다.
- iv. 만약 message 가 "#TRUE"이면
 - 1. 실제 server 가 전달하고자 한 message 는 "#TRUE" 다음의 message 이므로 다음 문자열을 읽어 message 에 저장한다.
 - 2. CREATE 혹은 JOIN 이 성공적으로 일어난 것이므로 inGroup 을 true 로 변경한다. (CREATE 혹은 JOIN 이 성공적으로 수행되면 server 가 client 에게 "#TRUE"를 전송하도록 프로그램을 작성했다.)
- v. 만약 message 가 "#FALSE"이면
 - 1. 실제 server 가 전달하고자 한 message 는 "#FALSE" 다음의 message 이므로 다음 문자열을 읽어 message 에 저장한다.

2. CREATE 혹은 JOIN 에 실패한 것이므로 inGroup 을 false 로 변경한다.
(CREATE 혹은 JOIN 에 오류가 발생하면 server 가 client 에게 “#FALSE”를 전송하도록 프로그램을 작성했다.)

vi. message 를 출력한다.

d. public Boolean getInGroup() : inGroup 을 return 하는 method

- i. Client Class 의 객체가 자신의 group 참여 여부를 설정하기 위해 사용되는 method 이다.

5. 코드 상세 설명 – Server Class

a. variables

- i. public static int portNum1 : 채팅과 관련된 역할을 수행하는 socket 의 port number
- ii. public static int portNum2 : file transfer 과 관련된 역할을 수행하는 socket 의 port number
- iii. public ServerSocket welcomeSocket : client socket 의 hand-shaking socket
- iv. public Socket socket : welcomeSocket 을 통해 client 와 server 가 연결된 뒤 그 후의 역할을 수행하는 Socket
- v. public ServerSocket : fileWelcomeSocket : file transfer 와 관련된 socket 의 hand-shaking socket
- vi. public Socket fileSocket : fileWelcomeSocket 을 통해 file transfer 와 관련된 socket 과 server 가 연결된 뒤 그 후의 역할을 수행하는 Socket
- vii. public ArrayList<Group> groupList : 채팅이 이루어지는 group 의 목록을 저장한 ArrayList

b. public static void main(String[] args) : Server Class 의 main method

- i. 프로그램 실행 시 받은 첫번째 인자를 portNum1, 두번째 인자를 portNum2 에 저장한다.
- ii. Server Class 의 객체 server 를 생성한다.
- iii. runServer() method 를 실행한다.

c. public void runServer() : client 와의 연결을 수행한 후 수신한 문자열에 대한 작업을 수행하는 Thread 를 할당하는 method

- i. 100 개의 ServerThread Class 객체를 가지는 ServerThread[] st 를 선언한다.
- ii. st 배열의 index 를 나타내는 threadCnt 를 선언한다.
- iii. portNum1 을 port number 로 가지는 채팅에 관한 ServerSocket welcomeSocket 을 생성한다.

- iv. portNum2 를 port number 로 가지는 file transfer 에 관한 ServerSocket
fileWelcomeSocket 을 생성한다.
- v. vi.부터 ix.까지의 반복문을 수행한다.
- vi. socket = welcomeSocket.accept()를 통해 hand-shaking process 를 수행한 후 이후
작업을 socket 에게 넘긴다.
- vii. groupList, socket, fileWelcomeSocket 을 인자로 가지는 ServerThread 를 생성해
threadCnt 번째 st 에게 할당한다.
- viii. st 를 start 하고, threadCnt 를 1 늘린다.
- ix. threadCnt 가 100 이상이라면 threadCnt 를 0 으로 초기화 해 배열을 처음부터
사용할 수 있도록 설정한다. 즉, 이 server 프로그램은 한번에 최대 100 명의 client 가
접속할 수 있다.

6. 코드 상세 설명 – ServerThread Class extends Thread

a. variables

- i. public Socket socket : client 에게 데이터를 수신하고 전송하는 socket
- ii. public ArrayList<Group> groupList : server 가 관리하는 group 의 목록
- iii. public DataOutputStream out : socket 에서 데이터를 실제로 전송하는 역할을 하는
DataOutputStream
- iv. public DataInputStream in : socket 에서 데이터를 실제로 수신하는 역할을 하는
DataInputStream
- v. public String command : client 에서 전송한 명령어
- vi. public Group group : ServerThread 객체가 관리하는 client 의 group
- vii. public String groupName : ServerThread 객체가 관리하는 client 의 group 이름
- viii. public boolean inGroup : ServerThread 객체가 관리하는 client 의 group 참여 여부
저장
- ix. public ServerSocket fileWelcomeSocket : file-transfer socket 의 hand-shaking socket
- x. public Socket fileSocket : file-transfer 를 수행하는 socket
- xi. public String fileName : file-transfer 에 사용될 file 의 이름
- xii. public File storage : group 에서 client 로부터 받을 file 을 저장하는 폴더
- xiii. public Client client : ServerThread 에 할당된 client Socket
- xiv. public String userName : ServerThread 에 할당된 client 의 client name

b. Constructor

- i. public ServerThread(ArrayList<Group> groupList, Socket socket, ServerSocket
fileWelcomeSocket)

1. ServerThread 객체의 groupList 를 인자로 전달받은 groupList 로 지정한다.
이때, shallow-copy 를 하기 때문에 groupList 의 변화 사항은 Server 의 groupList 에도 반영된다.
2. ServerThread 객체의 socket 을 인자로 전달받은 socket 으로 지정한다.
3. ServerThread 객체의 fileWelcomeSocket 을 인자로 지정받은 fileWelcomeSocket 으로 지정한다.

c. public synchronized void run() : ServerThread 를 실행하는 method

- i. ii.부터 x.까지의 반복문을 수행한다.
- ii. socket 으로부터 데이터를 전달받는 DataInputStream in 을 생성한다.
- iii. socket 으로부터 명령어를 전달받아 command 에 저장한다.
- iv. socket 으로부터 전달하는 DataOutputStream out 을 생성한다.
- v. 만약 command 가 “#CREATE”이면
 1. createGroup() method 를 실행한다.
- vi. 만약 command 가 “#JOIN”이면
 1. joinGroup() method 를 실행한다.
- vii. 만약 command 가 “#PUT” 혹은 “#GET”이면
 1. fileWelcomeSocket 으로부터 hand-shaking process 를 수행한 후 이후 작업을 수행하는 fileSocket 을 할당한다.
 2. socket 으로부터 file 의 이름을 전달받아 fileName 에 이를 저장한다.
 3. command, fileName, groupName, fileSocket, out 을 인자로 가지는 SeverFileThread ft 를 생성하고 이를 실행한다.
 4. 만약 command 가 “#PUT”이면 file 이 업로드되었음을 알리는 메시지 (“----- (fileName) Uploaded -----”)를 group 에 있는 모든 client 에게 전송한다. 이때, file 을 put 한 client 와 나머지 client 들에게는 서로 다른 method 를 이용해 메시지를 전송한다.
 5. fileSocket 을 닫는다.
- viii. 만약 command 가 “#STATUS”라면
 1. showStatus() method 를 실행한다.
- ix. 만약 command 가 “#EXIT” 혹은 “#QUIT”이라면
 1. 만약 client 가 어떠한 group 에 참여하고 있다면 해당 group 을 나온다.
 2. 만약 command 가 “#QUIT”이라면 socket 을 닫는다.
- x. 만약 command 가 “#SEND”이라면

1. client 로부터 문자열을 전달받아 이를 message 에 저장한다.
2. sendMessage method 를 실행한다.

d. public void createGroup() throws Exception : group 생성의 역할을 수행하는 method

- i. server 로부터 두 문자열을 전달받아 첫번째 문자열을 groupName, 두번째 문자열을 userName 에 저장한다.
- ii. 중복된 이름을 가지는 group 을 생성할 수 없으므로, 만약 groupList 에 groupName 과 같은 group 의 이름을 가지는 group 이 있다면 “#CREATE” 명령어를 전달한 client 에게 group 생성에 실패했다는 뜻의 메시지인 “#FALSE”와 오류 메시지를 (“----- A group with name ‘(group name)’ already exists -----”) 를 client 에게 전송한다.
- iii. groupName, userName, socket 을 인자로 가지는 Client client 를 생성한다.
- iv. groupName 을 인자로 가져 그룹의 이름을 groupName 으로 하는 group 을 생성한다.
- v. Client Class 의 joinGroup method 를 이용해 client 를 group 에 가입시킨다.
- vi. Group Class 의 addClient method 를 이용해 group 에 client 을 등록한다.
- vii. groupList 에 group 을 등록한다.
- viii. inGroup 을 true 로 지정해 group 이 성공적으로 create 되었음을 저장한다.
- ix. (groupName)_storage 의 이름을 가진 폴더를 생성한다.
- x. client 에게 group 생성에 성공했음을 알리는 메시지인 “#TRUE”를 전송하고 그룹 생성 메시지를 group 내의 모든 client 들에게 전송한다.

e. public void joinGroup() throws Exception : group 참가의 역할을 수행하는 method

- i. server 로부터 두 문자열을 전달받아 첫번째 문자열을 groupName, 두번째 문자열을 userName 에 저장한다.
- ii. groupName, userName, socket 을 인자로 가지는 Client client 를 생성한다.
- iii. group 내에 중복된 이름을 가진 client 가 있을 수 없으므로, 참가하고자 하는 group 안에 중복된 이름을 가진 client 가 있다면 “#JOIN” 명령어를 전달한 client 에게 group 참가에 실패했다는 뜻의 메시지인 “#FALSE”와 오류 메시지 (“----- There is a user with the same name in the group -----”)를 client 에게 전송하고 method 를 종료한다.
- iv. client 에게 group 을 할당하고, group 에 client 를 등록한다.
- v. (groupName)_storage 의 이름을 가진 폴더를 folder 에 지정한다.
- vi. inGroup 을 true 로 지정해 group 에 성공적으로 join 했음을 저장한다.
- vii. client 에게 group 참가에 성공했음을 알리는 메시지인 “#TRUE”를 전송하고 그룹 참가 메시지를 group 내의 모든 client 들에게 전송한다.

- viii. 위의 과정을 수행하여도 group 에 참가하지 못한 경우 groupName 의 group 이 존재하지 않는 경우이므로 “#FALSE”와 오류 메시지를 client 에게 전송한다.

f. public void showStatus() throws Exception

- i. group 의 이름을 출력하는 String head 를 만든다.
- ii. client 에게 head 를 전송한다.
- iii. 반복문을 통해 group 내의 모든 client 의 이름을 client 에게 전송한다.
- iv. head 의 길이만큼 ‘\n’를 저장해 출력한다.

g. public void leaveGroup(Group group) throws Exception

- i. group 을 떠났다는 메시지를 group 내의 모든 client 에게 전송한다.
- ii. client 에게 할당된 group 을 해제하고, group 에서 client 를 삭제한다.
- iii. 만약 group 내의 client 가 없다면
 - 1. iterator 를 통해 해당 group 을 groupList 삭제한다.
 - 2. 그 group 의 storage 폴더와 그 내용물을 삭제한다.
- iv. group 과 client 를 null 로 만들어 group 과 client 를 초기화시킨다.
- v. inGroup 을 false 로 만들어 해당 ServerThread 가 관리하는 client 가 어떠한 group 에도 참여하고 있지 않음을 알린다.

h. public void sendMessage(String message) throws Exception

- i. 만약 ServerThread 에 할당된 client 가 group 에 참여중이라면 아래의 과정을 수행한다.
- ii. client 자기 자신을 제외한 group 내의 모든 client 들에게 문자열을 전송할 DataOutputStream sendOut 을 만든다.
- iii. 반복문을 통해 client 자기 자신을 제외한 group 내의 모든 client 들에게 message 를 전송한다.

7. 코드 상세 설명 – ServerFileThread Class

a. variables

- i. public String command : 해당 Class 의 객체가 수행할 명령을 담은 명령어
- ii. public String fileName : file transfer 에 사용될 file 의 이름
- iii. public String groupName : file transfer 가 이루어지는 group 의 이름
- iv. public Socket fileSocket : file transfer 가 이루어지는 Socket
- v. public DataOutputStream out : client 에게 file 을 전송할 때 사용

b. Constructor

- i. `public ServerFileThread(String command, String fileName, String groupName, Socket fileSocket, DataOutputStream out)` : 인자들을 Class 객체의 variable 로 저장하는 constructor

c. `public void run()` : file transfer 를 수행하는 method

- i. 만약 command 가 “#PUT”이라면
 - 1. (groupName)_storage 폴더 안에 fileName 을 file 의 이름으로 가지는 file 을 생성한다.
 - 2. file 을 생성할 FileOutputStream fo 를 만든다.
 - 3. fileSocket 으로부터 file 의 데이터를 수신할 InputStream in 을 만든다.
 - 4. 64KB 버퍼인 buffer 를 생성하고, 반복문을 통해 버퍼의 크기만큼 데이터를 client 로부터 계속 수신하여 file 을 다운로드 받는다.
 - 5. fo 와 in 을 닫는다.
- ii. 만약 command 가 “#GET”이라면
 - 1. (groupName)_storage 폴더 안에 fileName 을 file 의 이름으로 가지는 file 을 생성한다.
 - 2. 만약 file 이 존재하지 않는다면, client 에게 오류 메시지를 전달하고 method 를 종료한다.
 - 3. file 의 정보를 읽을 FileInputStream fis 를 생성한다.
 - 4. fileSocket 을 통해 client 로 정보를 전송할 OutputStream os 를 생성한다.
 - 5. 64KB 버퍼인 buffer 를 생성하고, 반복문을 통해 버퍼의 크기만큼 데이터를 client 에게 계속 전송한다.
 - 6. fis 와 os 를 닫는다.

8. 코드 상세 설명 – Group Class

a. variables

- i. `public String groupName` : group 의 이름
- ii. `public ArrayList<Client> clientList` : group 에 참여하는 client 의 목록
- iii. `public ArrayList<Socket> socketList` : group 에 참여하는 client 의 socket 목록

b. Constructor

- i. `public Group(String groupName)` : 인자로 받은 groupName 을 객체의 groupName 으로 지정한다.

c. `public void addClient(Client client, Socket socket)` : group 에 client 를 추가하는 method

- i. clientList 에 client 를 추가한다.

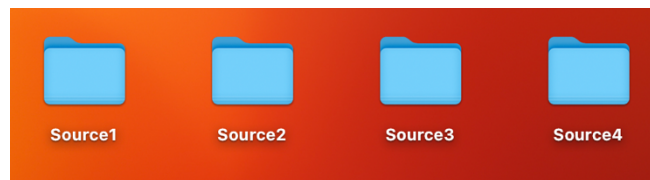
- ii. socketList 에 socket 을 추가한다.

d. public void removeClient(Client client, Socket socket) : group 에서 client 를 삭제하는 method

- i. clientList 에서 client 를 제거한다.
- ii. socketList 에서 socket 을 제거한다.

9. 프로그램 실행 방법

- a. 프로그램 작성자 테스트 환경 : MacBook Air M1, terminal 에서 실행
- b. 원활한 수행을 위해서는 한 source 폴더의 파일로 각각 하나의 Server 와 Client 를 실행해야 한다.
 - i. 예를 들어, 네 명의 Client 가 채팅 프로그램을 사용한다고 가정하면 4 개의 각기 다른 source 폴더에서 각각 Client 를 실행하고 한 폴더에서 Server 를 실행하면 된다.
 - ii. 다음은 네 명의 Client 가 채팅 프로그램을 사용하는 예시이다.



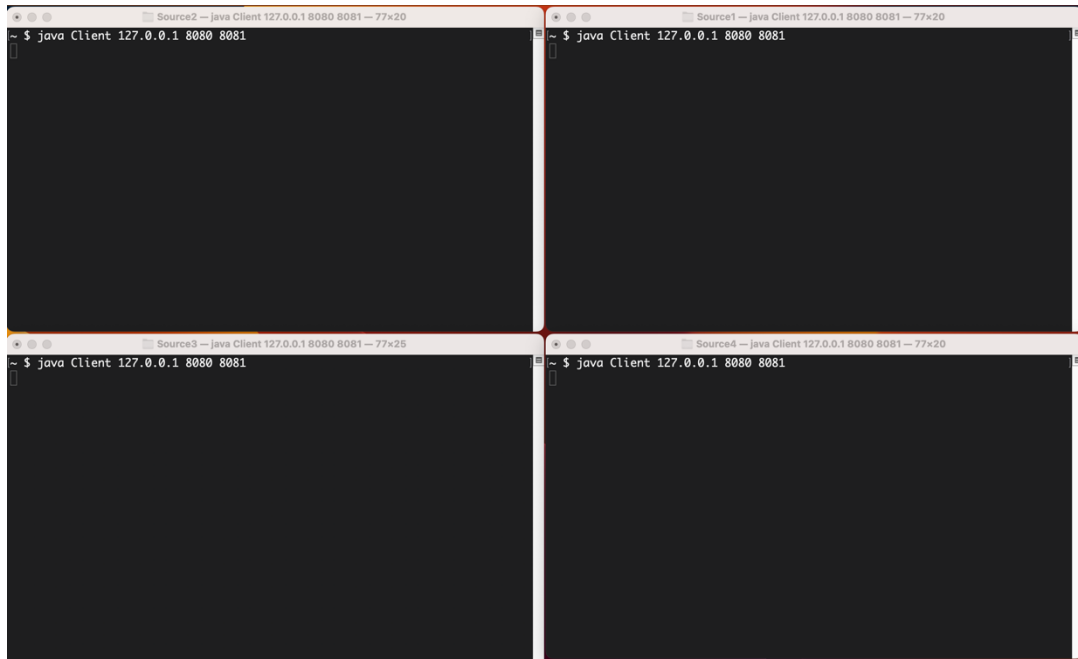
- 1. 다음과 같이 네 명의 Client 가 채팅 프로그램을 사용하기 위해선 4 개의 Source 를 담은 폴더가 필요하다.
- iii. 4 개의 Source 폴더 안에서 모두 다음 명령어를 통해 .Class File 을 만든다.
 - 1. javac Client.java : Client 프로그램 컴파일
 - 2. javac Server.java : Server 프로그램 컴파일

```
[~ $ cd Source1  
[~ $ javac Client.java  
[~ $ javac Server.java  
~ $
```

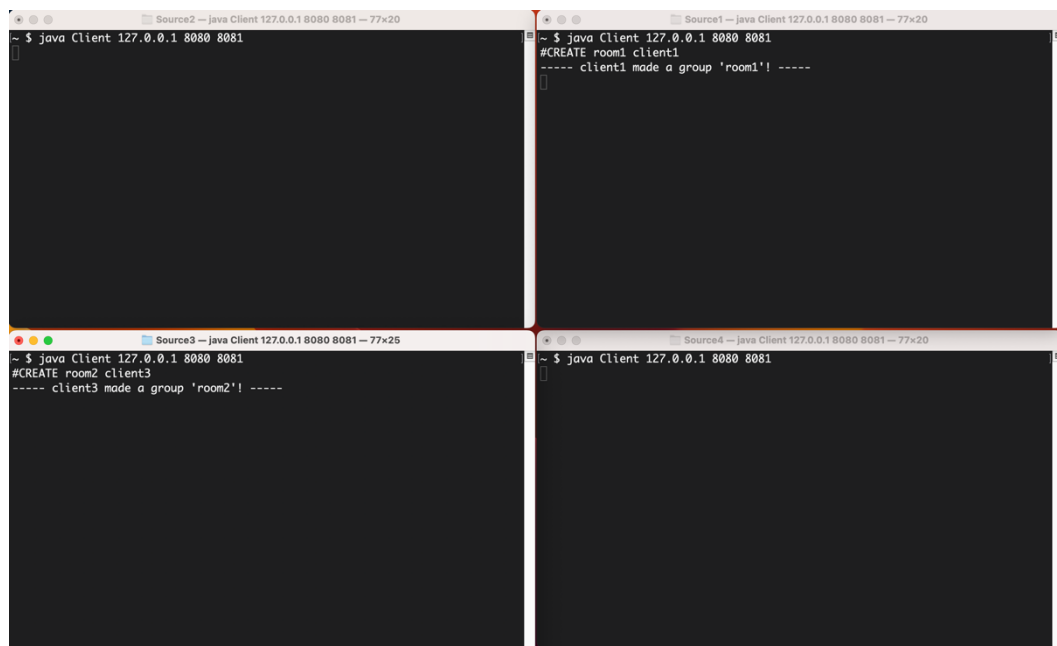
- iv. 4 개의 폴더 중 한 폴더에서 다음 명령어를 통해 Server 프로그램을 실행한다.
 - 1. java Server (portNum1) (portNum2)
 - 2. 본 문서에서는 portNum1 을 8080, portNum2 를 8081 로 설정하였다.

```
[~ $ java Server 8080 8081
```


- v. 4 개의 폴더 각각에서 다음 명령어를 통해 Client 프로그램을 실행한다.
1. java Client (ServerIP) (portNum1) (portNum2)
 2. 본 문서에서는 ServerIP 를 127.0.0.1 로 설정하였다. portNum 들은 Server 와 같아야 한다.

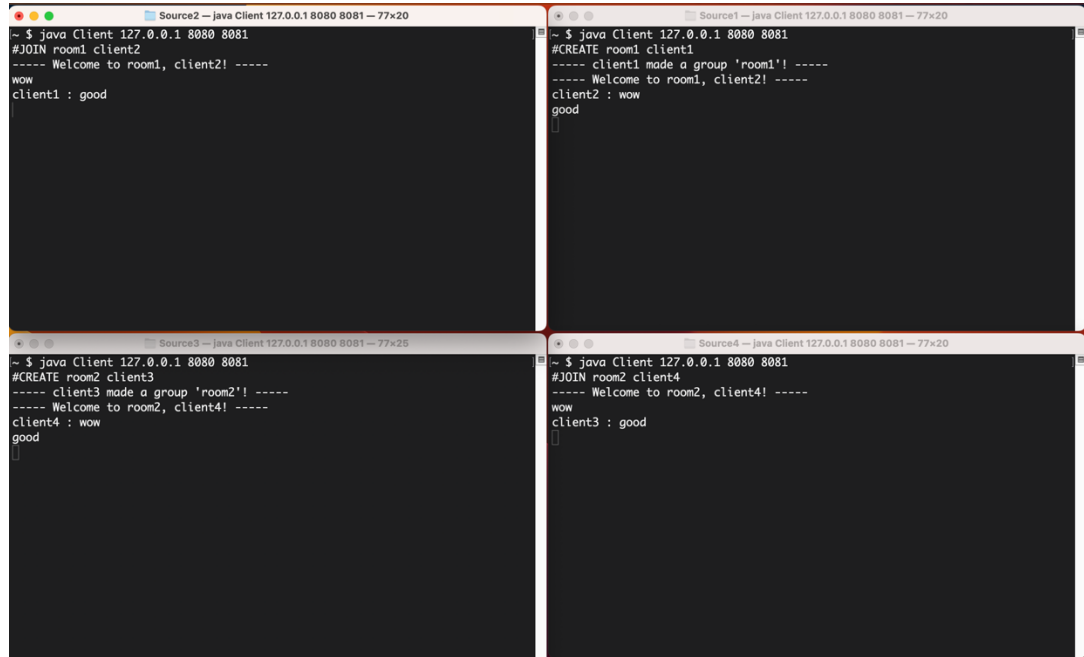


- vi. 다음 명령어를 통해 채팅방을 만든다.
1. #CREATE (groupName) (userName)
 2. 아래의 사진은 client1 이 room1, client3 가 room2 를 만든 예시이다.



vii. 다음 명령어를 통해 채팅방에 가입하고, 채팅을 시작한다.

1. #JOIN (groupName) (userName)
2. 아래의 사진은 client2 가 room1 에, client4 가 room2 에 가입한 예시이다. 각 채팅방의 채팅은 독립적으로 진행된다.



The image shows four terminal windows arranged in a 2x2 grid, each running a Java client. The top-left window (Source2) shows client2 joining room1 and receiving a welcome message. The top-right window (Source1) shows client1 creating room1 and then client2 joining, receiving a welcome message. The bottom-left window (Source3) shows client3 creating room2 and then client4 joining, receiving a welcome message. The bottom-right window (Source4) shows client4 joining room2 and receiving a welcome message. All windows show a prompt for the next command.

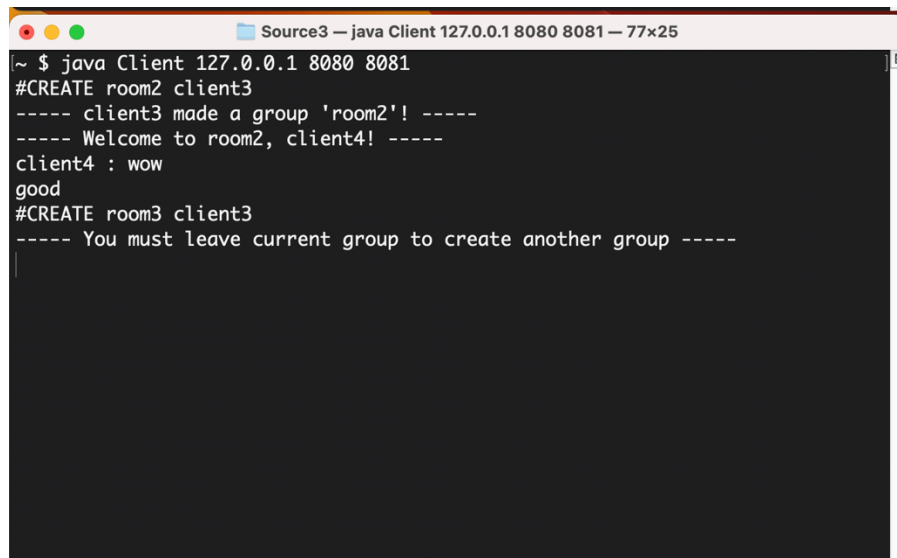
```
Source2 — java Client 127.0.0.1 8080 8081 — 77x20
~ $ java Client 127.0.0.1 8080 8081
#JOIN room1 client2
----- Welcome to room1, client2! -----
wow
client1 : good

Source1 — java Client 127.0.0.1 8080 8081 — 77x20
~ $ java Client 127.0.0.1 8080 8081
#CREATE room1 client1
----- client1 made a group 'room1'! -----
----- Welcome to room1, client2! -----
client2 : wow
good

Source3 — java Client 127.0.0.1 8080 8081 — 77x25
~ $ java Client 127.0.0.1 8080 8081
#CREATE room2 client3
----- client3 made a group 'room2'! -----
----- Welcome to room2, client4! -----
client4 : wow
good

Source4 — java Client 127.0.0.1 8080 8081 — 77x20
~ $ java Client 127.0.0.1 8080 8081
#JOIN room2 client4
----- Welcome to room2, client4! -----
wow
client3 : good
```

3. 채팅방에 가입한 상태이면 다른 채팅방에 가입하거나 채팅방을 만들 수 없다.



The image shows a single terminal window (Source3) where client3 attempts to create a new room (room3) while already being a member of room2. The system returns an error message: "You must leave current group to create another group".

```
Source3 — java Client 127.0.0.1 8080 8081 — 77x25
~ $ java Client 127.0.0.1 8080 8081
#CREATE room2 client3
----- client3 made a group 'room2'! -----
----- Welcome to room2, client4! -----
client4 : wow
good
#CREATE room3 client3
----- You must leave current group to create another group -----
```

4. 중복된 이름의 채팅방을 만들지 못하며, 어떤 채팅방에 가입할 때 중복된 이름으로 가입할 수 없다.

```
#CREATE room2 client2
----- A group with name 'room2' already exists -----
#JOIN room1 client1
----- There is a user with the same name in the group -----
```

- viii. “#STATUS” 명령어를 통해 현재 채팅방의 이름과 참가 인원을 알 수 있다.

```
#STATUS
----- room2 -----
client3
client4
-----
```

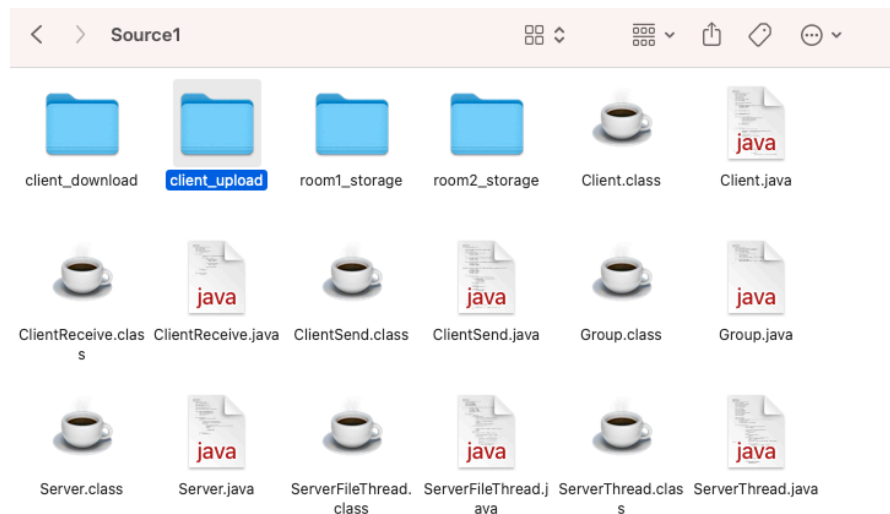
- ix. “#EXIT” 명령어를 통해 현재 채팅방을 떠날 수 있다. 채팅방을 떠난 후 새 채팅방을 만들거나 기존 채팅방에 참여하는 것은 자유롭다.

1. 만약 채팅방에 있는 모든 Client 가 채팅방을 떠나면 해당 채팅방과 그 채팅방의 storage 폴더는 완전히 삭제되므로 다른 사용자는 삭제된 이름의 채팅방을 새로 만들 수 있다. 예를 들어, room1 채팅방의 모든 Client 가 빠져나가면 다른 Client 는 room1 을 만들 수 있다.

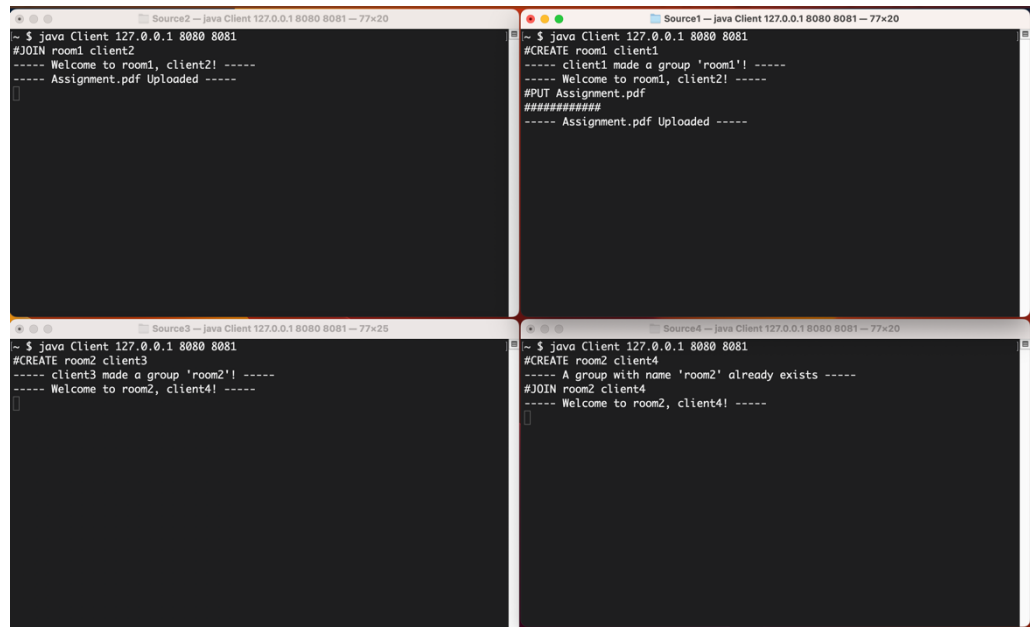
```
#EXIT
----- client1 left room1 -----
#CREATE room5 client1
----- client1 made a group 'room5'! -----
```

- x. 다음 명령어를 통해 Client 가 파일을 put 할 수 있다.

1. #PUT (fileName)
2. file 을 put 하기 위해서는 Client 의 Source 폴더 안의 Client_upload 폴더에 put 하고자 하는 file 을 넣어야 한다.



3. 아래 사진은 client1 이 room1 에 Assignment.pdf file 을 PUT 한 예시이다.
room1 의 멤버들에게는 파일이 업로드되었음을 알린다.



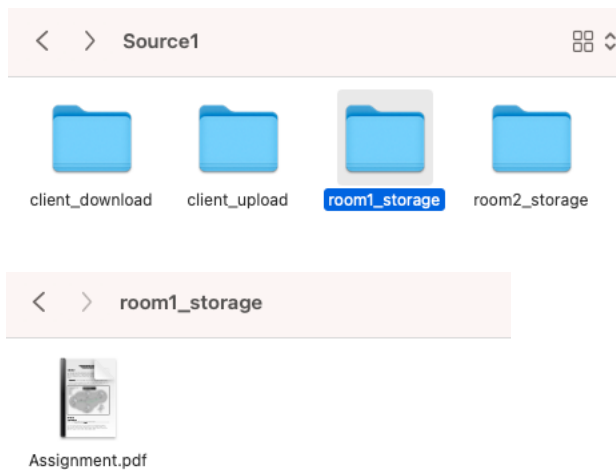
```
Source2 - java Client 127.0.0.1 8080 8081 - 77x20
~ $ java Client 127.0.0.1 8080 8081
#JOIN room1 client2
----- Welcome to room1, client2! -----
----- Assignment.pdf Uploaded -----

Source1 - java Client 127.0.0.1 8080 8081 - 77x20
~ $ java Client 127.0.0.1 8080 8081
#CREATE room1 client1
----- client1 made a group 'room1!' -----
----- Welcome to room1, client2! -----
#PUT Assignment.pdf
#####
----- Assignment.pdf Uploaded -----

Source3 - java Client 127.0.0.1 8080 8081 - 77x25
~ $ java Client 127.0.0.1 8080 8081
#CREATE room2 client3
----- client3 made a group 'room2!' -----
----- Welcome to room2, client4! -----

Source4 - java Client 127.0.0.1 8080 8081 - 77x20
~ $ java Client 127.0.0.1 8080 8081
#CREATE room2 client4
----- A group with name 'room2' already exists -----
#JOIN room2 client4
----- Welcome to room2, client4! -----
```

4. put 한 파일은 Server 프로그램의 Source 폴더 안의 (groupName)_storage 에 저장된다.



- xi. 다음의 명령어를 통해 Client 는 Server 에 저장된 파일을 get 할 수 있다.
1. #GET (fileName)
 2. 아래 사진은 client2 가 room1 로부터 Assignment.pdf file 을 GET 한 예시이다. room2 에 참여중인 client3 는 Assignment.pdf file 을 GET 할 수 없다.

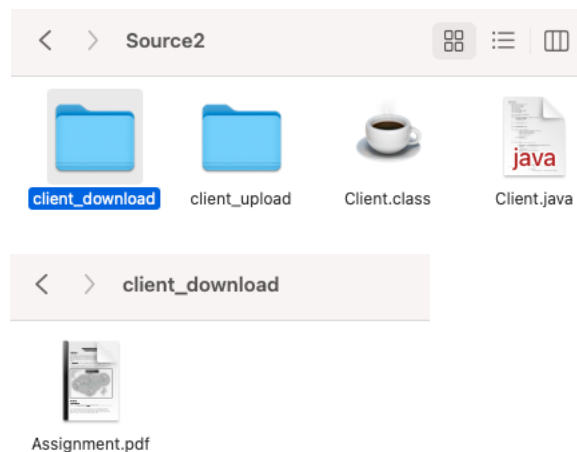
```
Source2 - java Client 127.0.0.1 8080 8081 - 77x20
~ $ java Client 127.0.0.1 8080 8081
#JOIN room1 client2
----- Welcome to room1, client2! -----
----- Assignment.pdf Uploaded -----
#GET Assignment.pdf
#####
----- Assignment.pdf Downloaded -----

Source1 - java Client 127.0.0.1 8080 8081 - 77x20
~ $ java Client 127.0.0.1 8080 8081
#CREATE room1 client1
----- client1 made a group 'room1!' -----
----- Welcome to room1, client2! -----
#PUT Assignment.pdf
#####
----- Assignment.pdf Uploaded -----

Source3 - java Client 127.0.0.1 8080 8081 - 77x20
~ $ java Client 127.0.0.1 8080 8081
#CREATE room2 client3
----- client3 made a group 'room2!' -----
----- Welcome to room2, client4! -----
#GET Assignment.pdf
----- Assignment.pdf not exists! -----

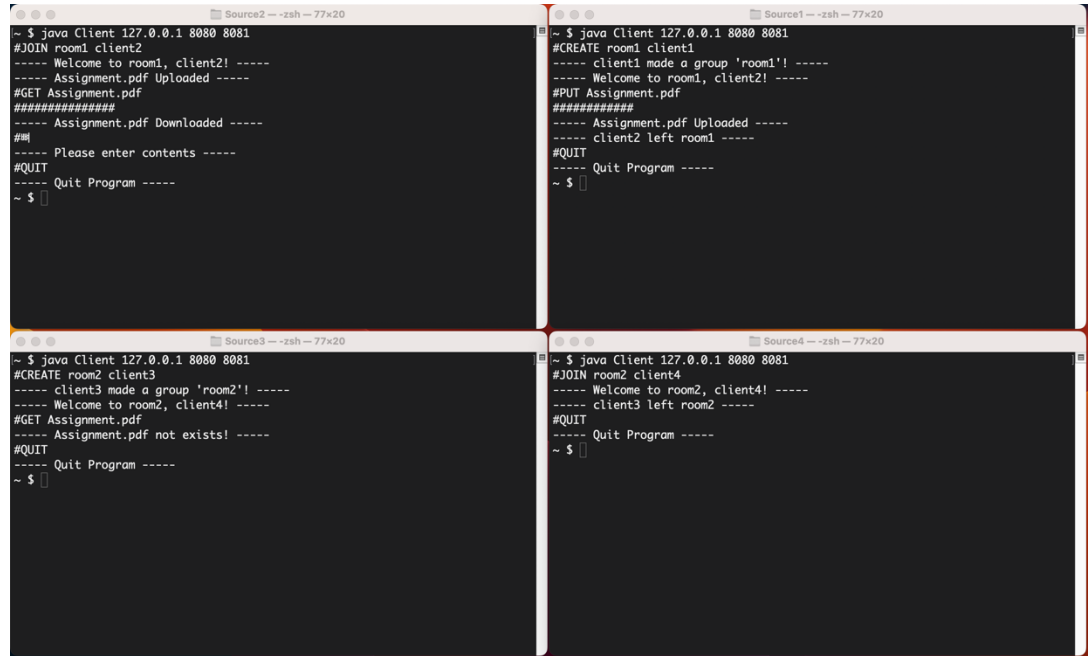
Source4 - java Client 127.0.0.1 8080 8081 - 77x20
~ $ java Client 127.0.0.1 8080 8081
#JOIN room2 client4
----- Welcome to room2, client4! -----
```

3. Client 가 GET 한 파일은 Client 프로그램을 실행한 Source 폴더 내의 client_download 폴더에 저장된다.



xii. “QUIT” 명령어를 통해 프로그램을 종료할 수 있다.

1. Client 가 채팅방에 참여중인 상태에서 #QUIT 명령어가 입력되면, 채팅방을 먼저 떠난 후 프로그램을 종료한다.
2. Client 가 프로그램을 종료하면 Client_upload 폴더와 Client_download 폴더는 같이 삭제된다.



```
Source2 -- -zsh -- 77x20
~ $ java Client 127.0.0.1 8080 8081
#JOIN room1 client2
----- Welcome to room1, client2! -----
----- Assignment.pdf Uploaded -----
#GET Assignment.pdf
#####
----- Assignment.pdf Downloaded -----
###
----- Please enter contents -----
#QUIT
----- Quit Program -----
~ $

Source1 -- -zsh -- 77x20
~ $ java Client 127.0.0.1 8080 8081
#CREATE room1 client1
----- client1 made a group 'room1!' -----
----- Welcome to room1, client2! -----
#PUT Assignment.pdf
#####
----- Assignment.pdf Uploaded -----
----- client2 left room1 -----
#QUIT
----- Quit Program -----
~ $

Source3 -- -zsh -- 77x20
~ $ java Client 127.0.0.1 8080 8081
#CREATE room2 client3
----- client3 made a group 'room2!' -----
----- Welcome to room2, client4! -----
#GET Assignment.pdf
----- Assignment.pdf not exists! -----
#QUIT
----- Quit Program -----
~ $

Source4 -- -zsh -- 77x20
~ $ java Client 127.0.0.1 8080 8081
#JOIN room2 client4
----- Welcome to room2, client4! -----
----- client3 left room2 -----
#QUIT
----- Quit Program -----
~ $
```