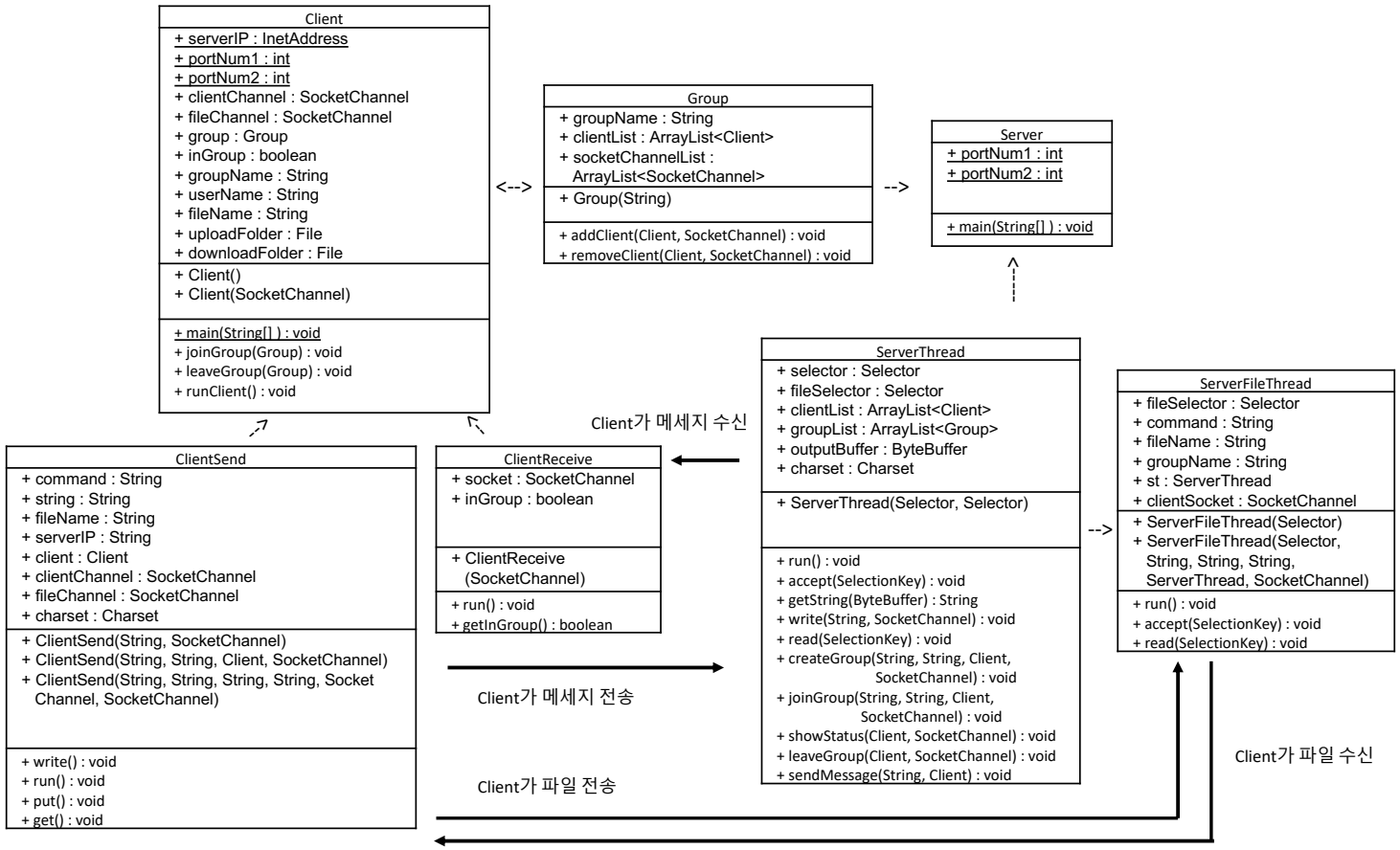
Computer Networks : Programming Assignment #2

컴퓨터소프트웨어학부 20210250205 강태욱

1. 프로그램 구조

a. 프로그램 구조도



b. 프로그램 설명

i. 이 프로그램은 TCP 와 NIO 방식의 파일 공유, 단체 채팅 프로그램이다.

ii. 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.

iii. server 에서는 하나의 Thread 와 Channel 을 이용해 각각의 client 와 동시에 소통한다. 이때, 각각의 Client 는 Server 와 Channel 을 통해 Non-Blocking 방식으로 데이터를 주고받는다. 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. NIO 방식의 채팅 프로그램을 구현하는 과정에서 시간 부족 등의 이유로 file transfer 기능을 완성하지 못하였습니다. “#PUT” 명령어는 Client 에서 작동하나, Server 에서 이를 받아 file 을 전달받는 기능을 수행하지 못하며, 이 명령어를 수행하면 프로그램에 오류가 발생합니다.
    2. “#PUT (file name)” 명령어는 (file name)을 이름으로 가지는 file 을 client 가 참여하고 있는 group 에게 업로드한다는 의미이다.
    3. “#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 non-blocking IO 와 TCP protocol 을 통해 server 에게 전송한다.

viii. server 는 client 로부터 받은 정보를 바탕으로 그에 맞는 작업을 수행 후 non-blocking IO 를 통해 결과 메세지를 client 에게 전달한다.

ix. client 는 server 로부터 메세지를 전달받는 thread 를 가지며, 이는 server 에게 정보를 전달하는 thread 와 분리되어 동작한다.

x. client 가 프로그램을 실행하면 ServerThread thread 의 clientList 에 client 의 정보가 등록된다. 이때, Client 객체를 attach 하여 추후 client 로부터 정보를 전달받을 때 이를 사용하도록 한다.

xi. 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 메세지를 전송하고 group 을 등록되지 않는다. client 가 어떠한 group 에 참여하고 있는 상태에서는 새로운 group 을 만들 수 없다.

xii. 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 메세지를 전송하고 client 는 등록되지 않는다. client 가 어떠한 group 에 참여하고 있는 상태에서는 새로운 group 에 참여할 수 없다.

xiii. "#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.

xiv. 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 메세지를 출력한다.

xv. '#'로 시작하지 않는 문자열은 채팅을 통해 전달하고자 하는 메세지로, 이 프로그램의 client 들은 thread 와 non-blocking IO 를 통해 메세지를 전송하고 수신한다.

xvi. 올바르지 않은 명령어, group 에 가입하지 않은 상태로 group 을 탈퇴하거나 메세지를 입력하는 등 비정상적 행동에 대해서는 그에 맞는 에러 메세지를 출력한 후, 사용자로부터 다시 입력을 받는다.

2. 코드 상세 설명 – Client Class

a. variables

i. public static String 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 SocketChannel clientChannel : server 로 message 를 전송하는 socket channel

v. public SocketChannel fileChannel : server 로 file 을 전송하는 socket channel

vi. public Group group : client 가 참여하고 있는 group

vii. public Boolean inGroup : client 의 group 참여 여부를 알려줌
viii. public String groupName : 명령어 인자로 받은 생성/참여하고자 하는 group 의 이름
ix. public String userName : 명령어 인자로 받은 group 에 참여하고자 하는 client 의 이름
x. public String filename : 명령어 인자로 받은 전송/수신할 file 의 이름
xi. public file uploadFolder : server 로 전송할 file 을 담은 폴더
xii. public file downloadFolder : server 로부터 수신할 file 을 담을 폴더

b. Constructors

i. public Client () : default constructor
ii. public Client (SocketChannel clientChannel) : clientChannel 을 client 의 this.clientChannel 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 SocketChannel 을 생성한 후 이를 clientChannel 에 할당한다.
ii. serverIP, portNum2 를 이용해 SocketChannel 을 생성한 후 이를 fileChannel 에 할당한다.
iii. 사용자로부터 문자열을 입력받을 BufferedReader br 을 생성한다.
iv. server 로부터 정보를 수신할 ClientReceive cr 을 선언하고 이를 실행한다.
v. put 할 file 을 담을 client_upload 폴더와 get 한 file 을 담은 client_download 폴더를 생성한다.
vi. vii.부터 xvii.까지의 반복문을 종료 전까지 수행한다.
vii. command line 으로부터 문자열을 입력받는다.
viii. 만약 문자열의 길이가 0 이면, 사용자가 어떠한 문자를 입력하지 않은 것이므로 오류 메시지("----- Please enter contents -----")를 출력하고 다시 문자열을 입력받는다.

ix. 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”, 입력받은 명령어를 담은 str, clientChannel 을 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.
6. 만약 그룹 생성에 성공했다면 client 객체의 userName 을 userName, 객체의 groupName 을 groupName 으로 지정해 client 의 userName 과 groupName 정보를 저장한다.

x. 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”, 입력받은 명령어를 담은 str, clientChannel 을 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.
6. 만약 그룹 가입에 성공했다면 client 객체의 userName 을 userName, 객체의 groupName 을 groupName 으로 지정해 client 의 userName 과 groupName 정보를 저장한다.

xi. 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, 명령어를 담은 str, serverIP, clientChannel, fileChannel 을 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 put 관련 메세지를 server 에 전송하고 file 을 업로드한다.
xii. 만약 입력받은 문자열이 “#GET”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. “#GET”, fileName, 명령어를 담은 str, serverIP, clientChannel, fileChannel 을 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 get 관련 메세지를 server 에 전송하고 file 을 다운로드한다.
xiii. 만약 입력받은 문자열이 “#STATUS” 라면
    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. “#STATUS”, clientChannel 을 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 status 요청 메세지를 server 에게 전송한다.
xiv. 만약 입력받은 문자열이 “#EXIT” 라면
    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. “#EXIT”, clientChannel 을 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 exit 요청 메세지를 server 에게 전송한다.
    3. inGroup 을 false 로 지정해 client 가 어떠한 group 에 참여하고 있지 않다는 상태를 저장한다.
xv. 만약 입력받은 문자열이 “#QUIT” 이라면

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. “#QUIT”, clientChannel 을 인자로 가지는 ClientSend cs thread 를 만들고 이를 실행해 exit 요청 메세지를 server 에게 전송한다.
3. Thread.sleep(100)을 통해 group exit 메세지를 전달받을 시간을 만든다.
4. ClientRecevie cr 을 null 로 만들어 server 로부터 정보를 더이상 전달받지 않도록 한다.
5. client_upload 와 client_download 폴더와 그 내용물을 삭제한다.
6. 종료 메세지와 함께 client 프로그램을 종료한다.

xvi. 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 메세지(“Not allowed command. Try again -----“)를 출력한다.
2. 사용자로부터 다시 문자열을 입력받는다.

xvii. 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 담은 str, client 객체, clientChannel 을 인자로 가지는 ClientSend cs thread 를 생성하고 이를 실행한다.

3. 코드 상세 설명 – ClientSend Class extends Thread

a. variables

i. public String command : Thread 를 통해 server 로 전달할 명령어

ii. public String string : Thread 를 통해 server 로 전달할 명령어 전체 또는 채팅 문장

iii. public String fileName : Thread 를 통해 server 로 전달할 file 의 이름

iv. public String serverIP : file transfer 을 수행할 socket channel 의 IP

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, SocketChannel clientChannel)는 this.clientChannel = clientChannel, this.command = command 의 역할을 수행한다.

c. public void write(String str) : Server 로 문자열을 전달하는 method

i. UTF-8 의 인코딩 방식을 Charset charset 에 저장한다.

ii. charset 을 통해 명령어 혹은 메세지를 인코딩해 이를 outputBuffer 에 넣는다.
iii. outputBuffer 의 내용을 clientChannel 을 통해 Server 로 보낸다.
iv. outputBuffer 의 내용을 지운다.

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

i. 만약 command 가 "#CREATE" 혹은 "#JOIN"이면
    1. write(string)을 통해 command 전체를 server 로 전달한다.
ii. 만약 command 가 "#PUT"이면
    1. file 전송에 관한 역할을 수행하는 put() method 를 실행한다.
iii. 만약 command 가 "#GET"이면
    1. file 수신에 관한 역할을 수행하는 get() method 를 실행한다.
iv. 만약 command 가 "#SEND"이면
    1. write("#SEND" + string)을 통해 command 와 메세지를 server 로 전달한다.
v. 만약 command 가 "#STATUS" 혹은 "#EXIT"이면
    1. write(command)를 통해 command 를 server 로 전달한다.
vi. 만약 command 가 "#QUIT"이면
    1. write(command)를 통해 command 를 server 로 전달한다.

e. 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. 명령어 전체를 server 로 전송한다.
iv. file 의 정보를 읽는 FileInputStream fis 를 생성한다.
v. channel 을 통해 데이터를 server 로 전송하는 FileChannel outChannel 을 생성한다.
vi. 64KB 의 크기를 가지는 ByteBuffer buffer 를 생성한다.
vii. 반복문을 통해 한번에 buffer 의 크기만큼 file 을 server 로 전송한다. 반복문이 한번 수행될 때 마다 '#'을 출력하여 64KB 씩 file 이 전송됨을 나타낸다.
viii. fis, fileChannel 을 닫는다.

f. public void get() : file 수신에 관한 역할을 수행하는 method // 미구현

i. 명령어 전체를 Server 로 전송한다.

4. 코드 상세 설명 – ClientReceive Class extends Thread

a. variables

i. public SocketChannel socket : server 로부터 정보를 전달받을 socket channel

ii. public Boolean inGroup : client 의 group 소속 여부를 저장

b. Constructors

i. public ClientReceive (SocketChannel socket) : ClientReceive Class 객체의 socket 을 인자로 전달받은 socket 으로 설정한다.

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

i. UTF-8 의 인코딩 방식을 Charset charset 에 저장한다.

ii. 문자열을 전달받은 ByteBuffer inputBuffer 를 지정한다.

iii. iv.부터 vii.까지의 반복문을 수행한다.

iv. server 로부터 수신된 문자열을 inputBuffer 에 저장한 후 이를 decoding 해 message 에 저장한다.

v. 만약 message 가 "#TRUE"이면

1. CREATE 혹은 JOIN 이 성공적으로 일어난 것이므로 inGroup 을 true 로 변경하고 inputBuffer 의 내용을 지운다. (CREATE 혹은 JOIN 이 성공적으로 수행되면 server 가 client 에게 "#TRUE"를 전송하도록 프로그램을 작성했다.)
2. Server 로부터 문자열을 다시 전달받는다.

vi. 만약 message 가 "#FALSE"이면

1. CREATE 혹은 JOIN 에 실패한 것이므로 inGroup 을 false 로 변경하고 inputBuffer 의 내용을 지운다. (CREATE 혹은 JOIN 에 오류가 발생하면 server 가 client 에게 "#FALSE"를 전송하도록 프로그램을 작성했다.)
2. Server 로부터 문자열을 다시 전달받는다.

vii. message 를 출력하고 inputBuffer 의 내용을 지운다.

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

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. clientChannel 로부터 데이터를 가져올 Selector selector 를 만든다.

iii. fileChannel 로부터 데이터를 가져올 Selector fileSelector 를 만든다.

iv. Server 의 hand-shaking 역할을 할 ServerSocketChannel chatServerChannel 로 만들고, 이를 NIO 방식으로 지정한다.

v. Server 의 역할을 수행할 chatSocket 을 만들고, portNum1 의 port number 를 이에 할당한다.

vi. accept 의 역할을 수행할 selector 를 chatServerChannel 에 등록한다.

vii. fileServer 의 hand-shaking 역할을 할 ServerSocketChannel fileServerChannel 로 만들고, 이를 NIO 방식으로 지정한다.

viii. fileServer 의 역할을 수행할 fileSocket 을 만들고, portNum2 의 port number 를 이에 할당한다.

ix. accept 의 역할을 수행할 fileSelector 를 fileServerChannel 에 등록한다.

x. Server 의 단일 Thread 인 st 를 선언하고 이를 실행한다.

6. 코드 상세 설명 – ServerThread Class extends Thread

a. variables

i. public Selector selector : client 에게 데이터를 전달받는 Selector

ii. public Selector fileSelector : client 에게 파일을 전달받는 Selector

iii. public ArrayList<Client> clientList : server 가 관리하는 client 의 목록

iv. public ArrayList<Group> groupList : server 가 관리하는 group 의 목록

v. public ByteBuffer outputBuffer : Client 로 전달할 정보를 담은 ByteBuffer

vi. public Charset charset : 문자열을 decoding, encoding 할 Charset

b. Constructor

i. public ServerThread(Selector selector, Selector fileSelector)

1. 인자로 전달받은 selector 와 fileSelector 를 객체의 selector 와 fileSelector 로 전달한다.

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

i. outputBuffer 를 만들고, UTF-8 방식의 Charset 을 charset 에 할당한다.

ii. iii.부터 v.까지의 반복문을 수행한다.

iii. client 로부터 데이터를 수신한다. NIO 방식이기에 file transfer 가 없더라도 fileSelctor.selector()를 이 곳에 선언하여도 blocking 이 발생하지 않는다.

iv. selector 를 통해 전달받은 SelectionKey 를 selectionKeys Set 에 저장한다.

v. Iterator 를 통해 전달받은 key 들에 대한 작업을 순차 시행한다.

1. 만약 key 가 acceptable 하면, accept(key) method 를 실행한다.
2. 만약 key 가 readable 하면, read(key) method 를 실행한다.
3. 작업이 끝난 key 를 삭제한다.

d. public void accept(SelectionKey key) throws IOException : key 를 accpet 하는 method

i. key 를 accpet 하고 그 후 작업을 수행하는 SocketChannel socketChannel 을 만들고, 이를 NIO 방식으로 지정한다.

ii. 해당 channel 과 소통할 Client newClient 를 만들고, 이를 clientList 에 넣는다.

iii. 정보를 읽어오는 selectionKey 를 만들고, 이에 위에서 만든 newClient 를 attach 해 데이터를 수신하는 과정에서 newClient 를 사용할 수 있도록 한다.

e. public String getString(ByteBuffer) : 전달받은 ByteBuffer 를 decoding 해 String 으로 만드는 method

f. public void write(String str, SocketChannel socket) : client 로 문자열을 전송하는 method

i. UTF-8 의 인코딩 방식을 Charset charset 에 저장한다.

ii. charset 을 통해 str 을 인코딩해 이를 outputBuffer 에 저장한다.

iii. socket 을 통해 client 로 outputBuffer 를 전송한다.

iv. outputBuffer 를 비운다.

g. public void read(SelectionKey key) throws IOException : key 를 read 하는 method

i. key 로부터 Client 객체를 전달받아 이를 client 에 저장한다.

ii. 수신할 데이터를 저장할 inputBuffer, 전달받은 명령어인 command, group 의 이름 groupName, client 의 이름 userName, file 의 이름 fileName, 메세지 message 를 지정한다.

iii. key 를 통해 client 와 소통하는 clientSocket 을 만들고 client 로부터 데이터를 수신해 이를 inputBuffer 에 저장한다.

iv. inputBuffer 의 데이터를 getString 을 통해 문자열로 바꾸고 이를 공백을 기준으로 나누어 str 에 저장한다. getString 을 통해 전달받은 문자열은 “CREATE room1 client1” 등의 형식이다.

v. command 를 str 의 첫번째 요소로 지정한다.

vi. 만약 command 가 “#CRAETE”이면

1. str[1]을 groupName, str[2]를 userName 으로 지정한 후 createGroup(groupName, userName, client, clientSocket) method 를 실행한다.

vii. 만약 command 가 "#JOIN"이면

1. str[1]을 groupName, str[2]를 userName 으로 지정한 후 joinGroup(groupName, userName, client, clientSocket) method 를 실행한다.

viii. 만약 command 가 "#PUT" 혹은 "#GET"이면

1. fileSelector, command, fileName, client.groupName, ServerThread 객체 그 자체, clientSocket 을 인자로 가지는 SeverFileThread ft 를 생성하고 이를 실행한다.
2. 만약 command 가 "#PUT"이면 file 이 업로드되었음을 알리는 메세지 ("----- (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 메세지를 전송한다.
3. fileSocket 을 닫는다.

ix. 만약 command 가 "#STATUS"라면

1. showStatus(client, clientSocket) method 를 실행한다.

x. 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 key 를 더이상 받지 않도록 해 client 와 server 사이의 소통을 끊는다.

xi. 만약 command 가 "#SEND"이라면

1. client 로부터 문자열을 전달받고 client 의 이름과 함께 이를 message 에 저장한다.
2. sendMessage method 를 실행한다.

h. public void createGroup(String groupName, String userName, Client client, SocketChannel clientChannel) : group 생성의 역할을 수행하는 method

i. 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 메세지인 "#FALSE"와 오류 메세지 ("----- A group with name '(group name)' already exists -----") 를 client 에게 전송한다.

ii. groupName, userName, socket 을 인자로 가지는 Client client 를 생성한다.

iii. gourpName 을 인자로 가져 그룹의 이름을 groupName 으로 하는 group 을 생성한다.
iv. client.userName 을 userName, client.groupName 을 groupName, client.group 을 group 으로 지정해 client 의 이름을 설정한다.
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 메세지인 "#TRUE"를 전송하고 그룹 생성 메세지를 group 내의 모든 client 들에게 전송한다.

i. public void joinGroup(String groupName, String userName, Client client, SocketChannel clientChannel) throws Exception : group 참가의 역할을 수행하는 method

i. client.userName 을 userName 으로 지정해 client 의 이름을 설정한다.
ii. 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 메세지인 "#FALSE"와 오류 메세지 ("----- There is a user with the same name in the group -----")를 client 에게 전송하고 method 를 종료한다.
iii. client.userName 을 userName, client.groupName 을 groupName, client.group 을 group 으로 지정해 client 의 이름을 설정한다.
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 메세지인 "#TRUE"를 전송하고 그룹 참가 메세지를 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 메세지를 client 에게 전송한다.

j. public void showStatus()

i. group 의 이름을 담은 String status 를 만든다.
ii. 반복문을 통해 group 내의 모든 client 의 이름을 status 에 저장한다.
iii. head 의 길이만큼 '-'를 status 를 저장하고 이를 client 에게 전송한다.

k. public void leaveGroup(Group group)

i. group 을 떠났다는 메세지를 group 내의 모든 client 에게 전송한다.
ii. client 의 group 을 xGroup 으로 설정한다.

iii. client 에게 할당된 group 을 해제하고, group 에서 client 를 삭제한다.

iv. 만약 xGroup 내의 client 가 없다면

1. iterator 를 통해 해당 group 을 groupList 삭제한다.
2. 그 group 의 storage 폴더와 그 내용물을 삭제한다.

v. group 과 client 를 null 로 만들어 group 과 client 를 초기화시킨다.

vi. inGroup 을 false 로 만들어 해당 ServerThread 가 관리하는 client 가 어떠한 group 에도 참여하고 있지 않음을 알린다.

l. public void sendMessage(String message)

i. 만약 ServerThread 에 할당된 client 가 group 에 참여중이라면 아래의 과정을 수행한다.

ii. 반복문을 통해 client 자기 자신을 제외한 group 내의 모든 client 들에게 message 를 전송한다.

7. 코드 상세 설명 – ServerFileThread Class

a. variables

i. public Selector fileSelector : file 에 관한 정보를 얻을 Selector

ii. public String command : 해당 Class 의 객체가 수행할 명령을 담은 명령어

iii. public String fileName : file transfer 에 사용될 file 의 이름

iv. public String groupName : file transfer 가 이루어지는 group 의 이름

v. public ServerThread st :

vi. public SocketChannel clientSocket : client 에게 file 을 수신, 전송할 때 사용

b. Constructor

i. public ServerFileThread(Selector fileSelector, String command, String fileName, String groupName, ServerThread st, SocketChannel clientSocket) : 인자들을 Class 객체의 variable 로 저장하는 constructor

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

i. outputBuffer 를 만들고, UTF-8 방식의 Charset 을 charset 에 할당한다.

ii. iii.부터 v.까지의 반복문을 수행한다.

iii. client 로부터 데이터를 수신한다. NIO 방식이기에 file transfer 가 없더라도 fileSelctor.selector()를 이 곳에 선언하여도 blocking 이 발생하지 않는다.

iv. selector 를 통해 전달받은 SelectionKey 를 selectionKeys Set 에 저장한다.

v. Iterator 를 통해 전달받은 key 들에 대한 작업을 순차 시행한다.

1. 만약 key 가 acceptable 하면, accept(key) method 를 실행한다.
2. 만약 key 가 readable 하면, read(key) method 를 실행한다.

3. 작업이 끝난 key 를 삭제한다.

d. public void accept(SelectionKey key) throws Exception : key 를 accpet 하는 method

i. key 를 accpet 하고 그 후 작업을 수행하는 SocketChannel fileSocketChannel 을 만들고, 이를 NIO 방식으로 지정한다.

e. public void read(SelectionKey key) throws Exception : key 를 read 하는 method

i. key 를 통해 client 와 소통하는 fileSocket 을 만든다.

ii. 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. fo 로부터 file 의 데이터를 수신할 inChannel 을 만든다.
4. 64KB 버퍼인 ByteBuffer buffer 를 생성하고, 반복문을 통해 버퍼의 크기만큼 데이터를 client 로부터 계속 수신하여 file 을 다운로드 받는다.
5. fo 를 닫는다.

iii. 만약 command 가 "#GET"이라면 // 미구현

1. (groupName)_storage 폴더 안에 fileName 을 file 의 이름으로 가지는 file 을 생성한다.

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<SocketChannel> sockeChanneltList : group 에 참여하는 client 의 socket channel 목록

b. Constructor

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

c. public void addClient(Client client, SocketChannel socketChannel) : group 에 client 를 추가하는 method

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

9. 프로그램 실행 방법

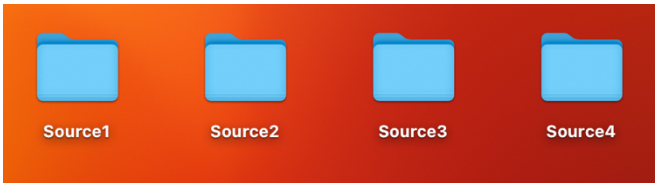
a. 프로그램 작성자 테스트 환경 : MacBook Air M1, terminal 에서 실행

b. 실행 방법은 TCP 를 사용한 Assignment 2 와 완전히 동일하다.

c. 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.

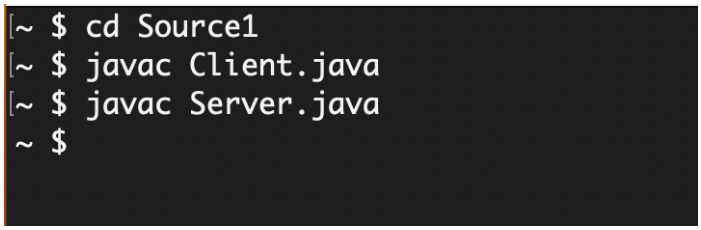
   ii. 다음은 네 명의 Client 가 채팅 프로그램을 사용하는 예시이다.



      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

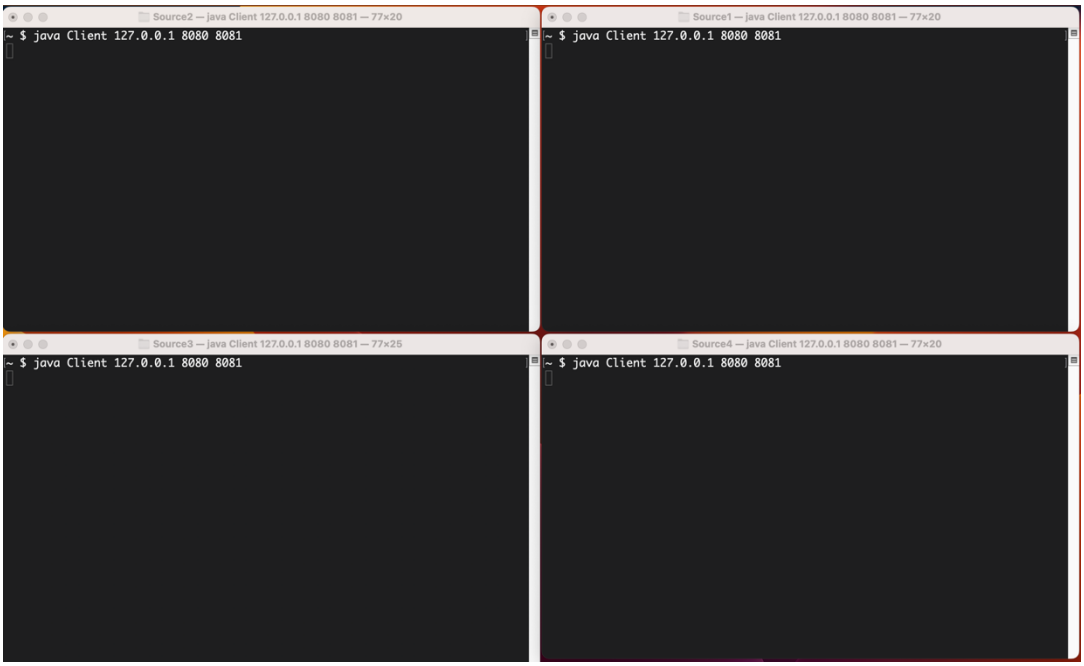


   iv. 4 개의 폴더 중 한 폴더에서 다음 명령어를 통해 Server 프로그램을 실행한다.

      1. java Server (portNum1) (portNum2)
      2. 본 문서에서는 portNum1 을 8080, portNum2 를 8081 로 설정하였다.

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. 프로그램 실행 중 사용하는 모든 채팅 관련 명령어는 Assignment 2 와 모두 동일하다. 파일 전송 관련 명령어는 오류가 발생하거나, 실행되지 않는다.

1. 아래의 사진은 이 프로그램을 사용해 채팅을 진행한 예시이다.