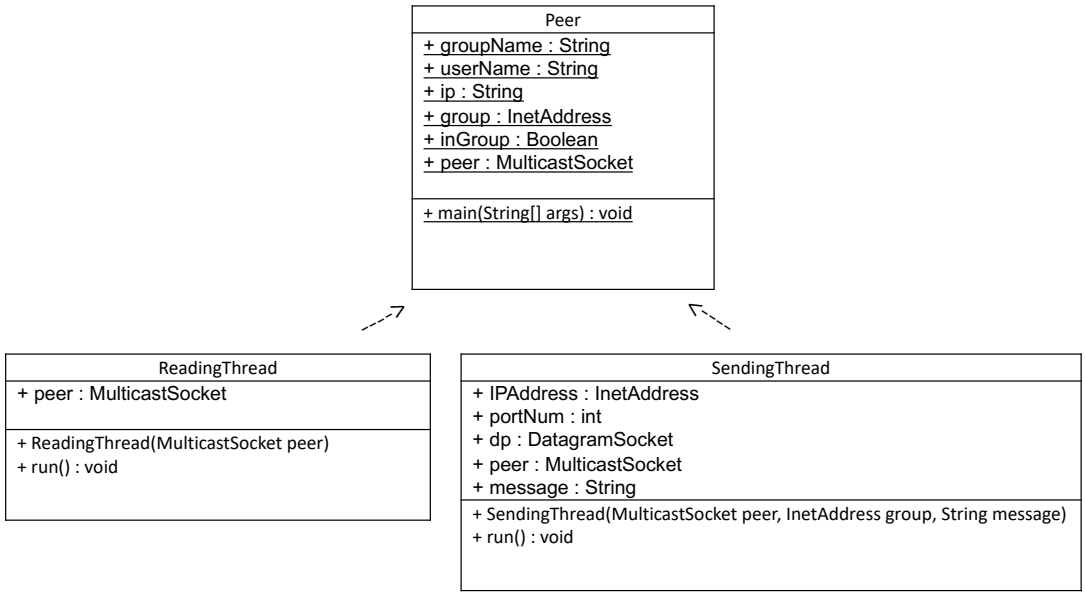
Computer Networks : Programming Assignment #1

컴퓨터소프트웨어학부 2021025205 강태욱

1. 프로그램 구조

a. 프로그램 구조도



b. 프로그램 설명

i. 이 프로그램은 Multicast 를 이용해 P2P 방식의 채팅 프로그램이다.

ii. 이 프로그램은 “#JOIN”, “#EXIT”, “#QUIT” 명령어를 가진다. “#JOIN (group name) (user name)” 명령어는 (group name)의 이름을 가지는 group 에 사용자 이름 (user name)으로 접속한다는 의미이다. “#EXIT” 명령어는 참여하고 있는 group 을 떠날 때 사용한다. “#QUIT” 명령어는 프로그램을 종료하기 위해 사용한다.

iii. group 에 새로운 peer 가 가입하면 Thread 를 통해 해당 group 에 있는 모든 peer 에게 새로운 사용자가 group 에 가입했음을 알린다.

iv. peer 는 한 group 에 가입한 상태로 다른 group 에 가입할 수 있다. 이때, 기존에 가입하고 있던 group 에서는 자동으로 나가게 된다.

v. peer 가 group 을 떠나면 Thread 를 통해 본인을 제외한 해당 group 에 있는 모든 peer 에게 해당 peer 가 group 을 떠났음을 알린다. group 을 떠난 peer 본인에게는 관련 메세지를 전달하지 않는다. “#EXIT” 명령어를 통하지 않고 group 을 떠난 경우는 group 의 peer 들에게 탈퇴 관련 메세지를 전달하지 않는다.

vi. '#'로 시작하지 않는 문자열은 채팅을 통해 전달하고자 하는 메세지로, 이 프로그램의 client 인 peer 들은 Thread 를 통해 메세지를 전송하고 수신한다.

vii. 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 – Peer Class

a. variable

i. public static String groupName : peer 가 참가할 group 의 이름을 저장한다.

ii. public static String userName : peer 가 group 에 참여할 때 사용할 이름을 저장한다.

iii. public static InetAddress group : peer 가 참가할 group 의 ip 를 InetAddress 형식으로 저장한다.

iv. public static Boolean inGroup : peer 가 group 에 참가중인지 여부를 저장한다. false 일 경우, peer 는 group 에 참여하지 않고 있다는 뜻이다.

v. public static MulticastSocket peer : Thread 를 통해 데이터를 전송하고 수신할 Multicast Socket 이다.

b. public static void main(String[] args) : 프로그램을 실행할 main method

i. 사용자로부터 문자열을 입력받도록 하는 BufferedReader br 을 정의한다.

ii. 프로그램 실행 시 얻는 첫번째 인자를 int 로 바꾸어 portNum 에 저장한다.

iii. portNum 을 socket 의 port number 로 가지는 Multicast Socket 인 peer 를 만든다.

iv. 반복문을 통해 입력 "#QUIT"이 들어올 때 까지 과정 v.부터 xii.를 반복한다.

v. br 를 통해 사용자로부터 문자열을 입력받는다.

vi. 만약 사용자가 '\n'(즉, 아무것도 입력하지 않고 enter key 를 치면) 만을 입력하면, 에러 메세지를 출력하고 다시 문자열을 입력받는다.

vii. 만약 입력받은 문자열이 "#JOIN (group name) (user name)"이면,

1. (group name)을 groupName 에 저장하고, (user name)을 userName 에 저장한다.
2. 만약 peer 가 이미 다른 채팅방에 참여하고 있으면 해당 채팅방을 떠난다.
3. SHA-256 을 이용해 groupName 을 hashing 한 후 이를 digest 에 저장한다.
4. digest 의 30 번째 byte 를 int x, 31 번째 byte 를 int y, 32 번째 byte 를 int z 에 저장한다.
5. x, y, z 를 이용해 225.x.y.z 와 같은 ip 형태로 만들어 group 에 저장한다.
6. peer.joinGroup(group)을 통해 peer 를 group 에 가입시킨다.

7. peer 가 어떠한 group 에 참여하고 있으므로 inGroup 을 true 로 변경한다.
8. groupName 과 userName 을 이용해 welcomeMessage 를 만들고, SendingThread 를 이용해 peer 들에게 해당 메세지를 전송한다.
9. ReadingThread 를 만들고, 이를 실행해 peer 들로부터 데이터를 받도록 한다.

viii. 만약 입력받은 문자열이 "#EXIT"이면,

1. peer 가 참여중인 group 이 없다면, 에러 메세지("----- You are not in any chat room. Try again -----")를 출력하고 사용자로부터 다시 문자열을 입력받는다.
2. userName 과 groupName 을 이용해 farewellMessage 를 만들고, SendingThread 를 이용해 이를 연결된 peer 들에게 전송한다. group 을 나가는 peer 자기 자신은 farewellMessage 를 전달받지 않는다.
3. ReadingThread rt 를 null 로 만들어 peer 들로부터 데이터를 더 이상 받지 않도록 만든다.
4. peer.leaveGroup(group)을 통해 peer 를 group 에서 탈퇴한다.
5. peer 가 더 이상 어떠한 group 소속이 아니므로 inGroup 을 false 로 설정한다.

ix. 만약 입력받은 문자열이 "#QUIT"이면,

1. 만약 peer 가 어떤 group 에 참여중이라면, 그 group 을 떠난다.
2. peer.close()를 통해 socket 을 닫는다.
3. 종료 메세지("-----Quit program -----")를 출력하고 프로그램을 종료한다.

x. 만약 입력받은 문자열의 첫 글자가 '#'이지만 "#JOIN", "#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.

xi. 만약 입력받은 문자열이 명령어가 아니고, peer 가 group 에 참여중인 경우

1. 입력받은 문자열 앞에 userName + " : "을 붙어 message 로 저장한다.
2. SendingThread 를 통해 같은 group 에 있는 모든 peer 들에게 message 를 전송한다.

xii. 만약 입력 문자열이 명령어가 아니고, peer 가 group 에 참여하고 있지 않은 경우

1. 에러 메세지("----- You are not in any chat room. Try again -----")를 출력한다.
2. 사용자로부터 다시 문자열을 입력받는다.

3. 코드 상세 설명 – SendingThread Class

a. variable

i. public InetAddress IPAddress : data 를 전송하는 peer 의 ip 를 저장한다.

ii. public int portNum : data 를 전송받을 peer 들의 port number 이다.

iii. public DatagramPacket dp : SendingThread 를 통해 전송할 data 이다.

iv. public MulticastSocket peer : data 를 전송할 Multicast Socket 이다.

v. public String message : SendingThread 를 통해 전송할 문자열이다.

b. public SendingThread(MulticastSocket peer, InetAddress group, String message)

i. SendingThread Class 의 Constructor 이다.

ii. parameter 로 전달받은 MulticastSocket peer 를 this.peer 에 저장한다.

iii. parameter 로 전달받은 InetAddress group 을 IPAddress 에 저장한다.

iv. parameter 로 전달받은 String message 를 this.message 에 저장한다.

c. public void run()

i. Thread 를 실행해 data 를 group 내의 peer 들에게 전송하도록 하는 method 이다.

ii. byte[] data = new byte[512]를 통해 512byte 의 chunk size 를 가지는 data 를 만든다.

iii. chunk size 만큼의 data 를 전송하는 DatagramPacket dp 를 만든다.

iv. dp 의 주소를 group 의 ip 인 IPAddress 로 설정한다.

v. dp 의 port number 를 데이터를 전송받는 peer 의 port number 인 portNum 으로 설정한다.

vi. peer 들에게 전송할 message 를 byte 로 만들어 byteMessage 에 저장한다.

vii. dp 가 전송할 데이터를 byteMessage 로 설정한다.

viii. peer.send(dp)를 통해 group 에 연결된 peer 들에게 데이터를 전송한다. 이 과정에서 synchronized(this)를 통해 thread 를 동기화한다.

4. 코드 상세 설명 – ReadingThread Class

a. variable

i. public MulticastSocket peer : data 를 수신할 Multicast Socket 이다.

b. public ReadingThread(MulticastSocket peer)

i. ReadingThread Class 의 constructor 이다.

ii. parameter 로 전달받은 MulticastSocket peer 를 this.peer 에 저장한다.

c. public void run()

i. peer 가 열린 동안 다음 반복문을 수행한다.

ii. 수신할 데이터를 저장할 공간인 512byte 크기의 data 를 만든다.

iii. 512byte 의 data 를 수신할 DatagramSocket dp 를 만든다.

iv. peer.receive(dp)를 통해 group 내의 peer 가 전송한 데이터를 수신한다. 이때, 메세지를 전송한 peer 본인 또한 group 에 참여하고 있기 때문에 자기 자신이 전송한 메세지를 수신한다. 이 과정에서 synchronized(this)를 통해 thread 를 동기화한다.

v. dp 의 data 를 String 으로 바꾸어 출력한다.

5. 프로그램 실행 방법

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

b. source 폴더에서

i. javac Peer.java 명령어를 통해 Peer.class 파일을 생성한다.

ii. javac SendingThread.java 명령어를 통해 SendingThread.class 파일을 생성한다.

iii. javac ReadingThread.java 명령어를 통해 ReadingThread.class 파일을 생성한다.

```
~ $ ls
Peer.java               ReadingThread.java      SendingThread.java
~ $ javac Peer.java
Note: Peer.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
~ $ javac SendingThread.java
~ $ javac ReadingThread.java
~ $ ls
Peer.class              ReadingThread.class     SendingThread.class
Peer.java               ReadingThread.java      SendingThread.java
~ $
```

c. java Peer (port number) 명령어를 통해 프로그램을 실행한다.

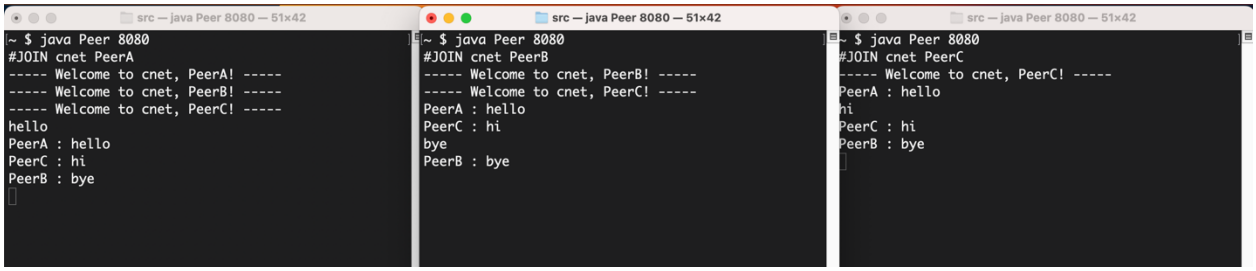
i. 본 설명 문서에서는 임의의 port number 8080 을 이용해 프로그램을 실행했다.

ii. peer (client)들은 모두 같은 port number 를 가져야 한다.

d. “JOIN (group name) (user name)” 명령어를 통해 채팅 group 에 가입한다.

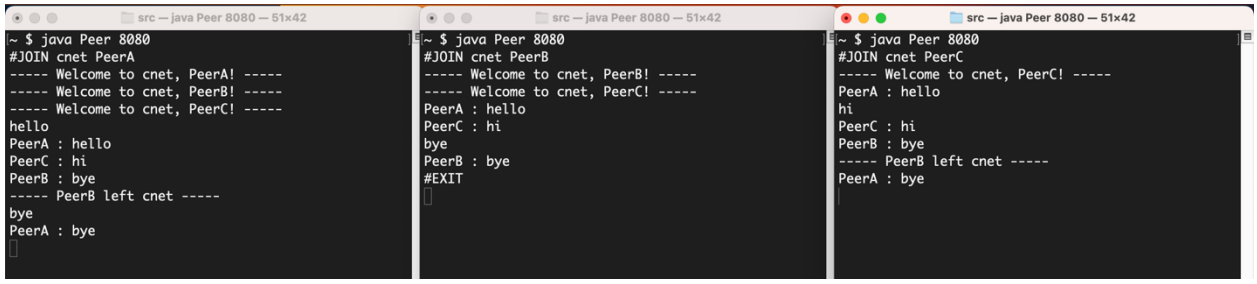
i. 본 설명문서에서는 임의의 group name “cnet”, user name “PeerA”, “PeerB”, 그리고 “PeerC”를 사용했다. 세 peer 모두 같은 port number 를 가진다.

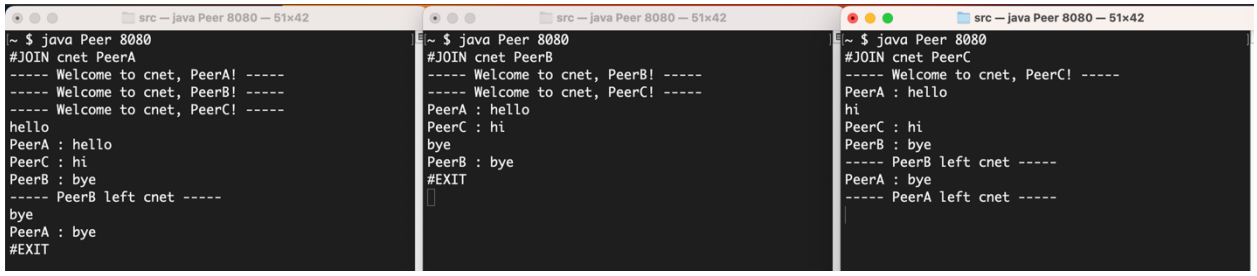
e. 보내고자 하는 메세지를 각자의 입력창을 통해 입력한다.



f. "#EXIT" 명령어를 통해 채팅 group 을 떠난다.

i. 아래의 사진을 통해 채팅 group 을 떠난 peer 에게는 메세지가 전달되지 않는 것을 볼 수 있다.





g. "#QUIT" 명령어를 통해 프로그램을 종료한다.