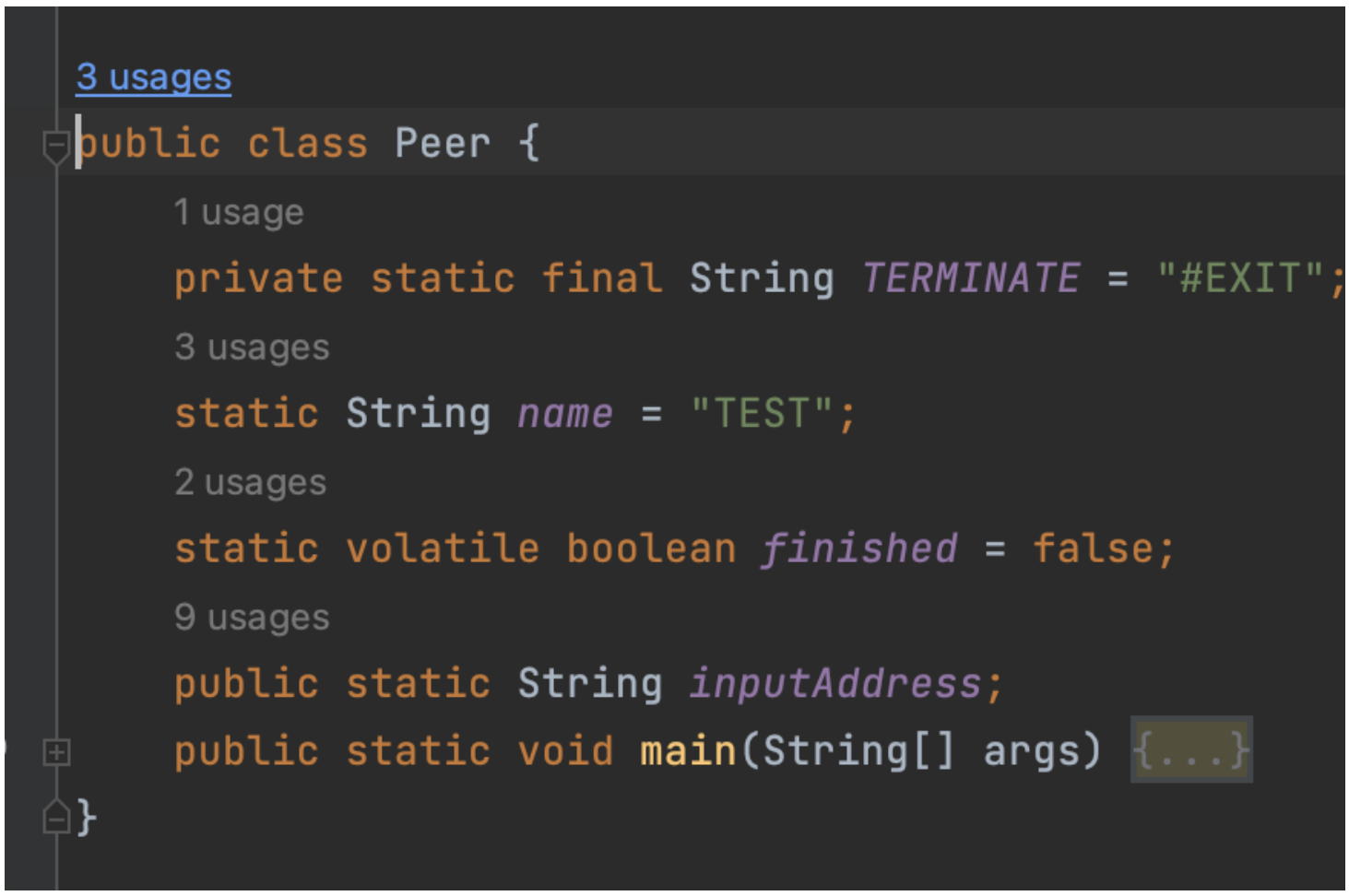
Assignment#1_2019072351_김정훈

Last edited by **김정훈** 5 minutes ago

** Assignment #1 – UDP 의 Multicast 를 이용한 P2P 방식의 오픈 채팅 프로그램을 구현한다.**

과제 1 에서는 P2P(Peer to Peer)방식의 오픈 채팅 프로그램을 구현한다.



최상위 클래스명은 Peer, ReadThread 이다.

1. 채팅 프로그램 실행 Peer 프로그램 실행 인자로 port number 를 입력 받는다. 실행 예시: java Peer portNo

portNo 를 인자로 받기 위에 Peer 클래스 내에서 Main function 에서 알맞은 인자를 받게 하였다. 피어 프로그램 수행 시에 명령어 라인 인자로 multicast port number(portNo)를 지정한다. 피어는 프로그램이 실행되면 원하는 채팅방에 들어가거나 새로운 채팅방을 개설할 수 있다. 채팅 내에서 “#”으로 시작하는 문장은 아래에서 설명하는 동작을 수행하기 위한 용도로만, 즉 명령어로만, 사용됨 (“#”으로 시작하는 메시지는 전달할 수 없음).

2. 채팅 연결 채팅에 참여하기 위해서 #JOIN 명령어를 구현 #JOIN (참여할 채팅방의 이름) (사용자 이름)

```
Scanner sc = new Scanner(System.in);
//every EXIT means another possible JOIN
while (true) {
    String tmp = sc.nextLine();
    if (tmp.startsWith("#JOIN")) {
        String[] buffer = tmp.split( regex: " ");
        inputAddress = buffer[1];
        name = buffer[2];
        break;
    }
}
```

채팅을 시작하는 Peer 는 네트워크 내 중복되지 않는 Multicast address 를 이용해서 새로운 채팅방을 생성 또는 기존 채팅방에 참여

포트 번호는 프로그램 실행에서 입력 받은 번호 사용

Multicast address 는 225.0.0.0 ~ 225.255.255.255 범위

입력 받은 채팅방 이름을 “SHA-256” 해시를 이용해서 Multicast address 225.x.y.z 로 변환(x, y, z 값을 구함)

java.security.MessageDigest 참고

해시 값(byte 배열)의 가장 마지막 3 개의 byte 를 이용해서 값을 구함

```
import java.net.*;
import java.io.*;
import java.nio.ByteBuffer;
import java.security.NoSuchAlgorithmException;
import java.util.*;
```

SHA256.java 파일을 미리 준비해둠과 동시에 NoSuchAlgorithmsException 를 추가하여 원활히 돌아갈 수 있도록 하였습니다.

한정된 범위 내에서 다른 채팅방 이름 사이에 같은 Multicast address 가 발생할 가능성이 존재하지만 본 과제 구현에서는 무시함

해당 채팅에 참여하고 싶은 Peer 는 채팅방 이름을 동일한 방법으로 구한 Multicast address 를 이용해서 채팅방에 참여할 수 있음

```java
try {
    //encryption process
    SHA256 sha256 = new SHA256();
    int port = Integer.parseInt(args[0]);
    inputAddress = sha256.encrypt(inputAddress);
    inputAddress = inputAddress.substring( beginIndex: inputAddress.length() - 6);
    int x = Integer.parseInt(inputAddress.substring(0, 2),  radix: 16);
    int y = Integer.parseInt(inputAddress.substring(2, 4),  radix: 16);
    int z = Integer.parseInt(inputAddress.substring( beginIndex: 4),  radix: 16);
    String hashAddress = "225." + x + "." + y + "." + z;
```

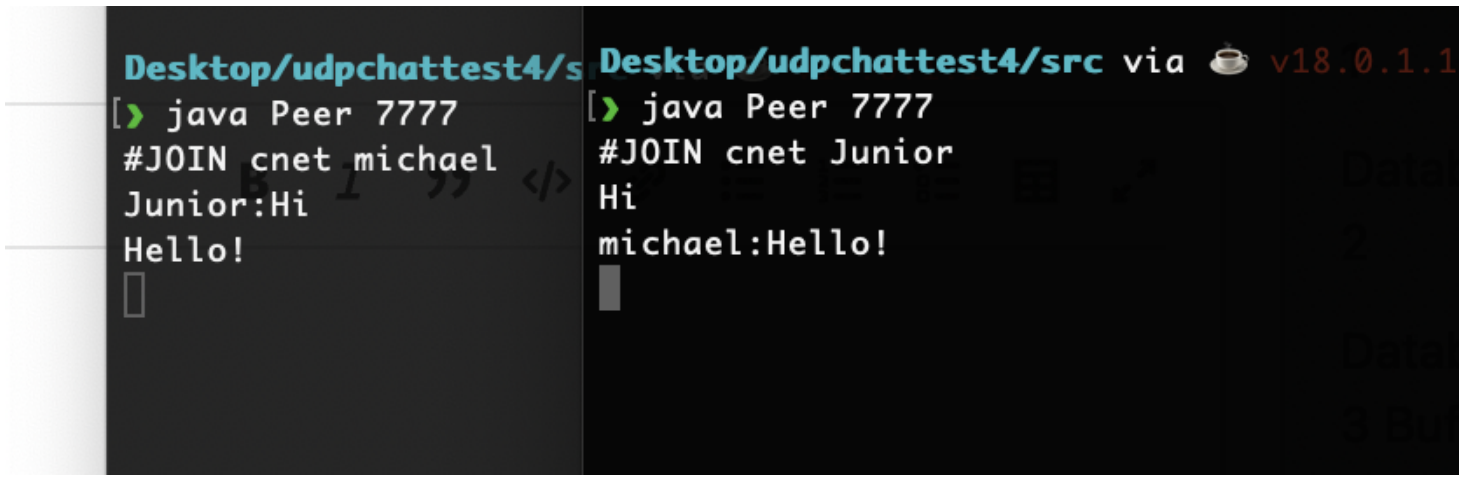
뒤에서 6글자를 가져와서, SHA256 복호화 하는 법으로 각 다른 채팅방 이름에 따라 Multicast Address 가 생성이 된다.

사용자 이름은 메시지 시작 부분에 포함되어서 전달되어서, 수신한 피어가 메시지를 화면에 출력할 때 송신자를 알 수 있도록 한다.

3. 채팅 Multicast 를 이용해서 채팅방 내에 있는 모든 Peer 에게 메시지를 전달

채팅 메시지를 chunk 단위(512 byte)로 나누어서 전송 출력 예시)

피어 B 는 “cnet” 채팅방에 조인했을 때 아래와 같이 피어 A 와 C 로부터의 메시지 수신을 화면에 출력한다. 아래의 출력 예는 피어 A 의 사용자 이름은 “PeerA”이고 피어 C 의 사용자 이름은 “PeerC”인 경우이다. PeerA: Hello PeerC: Bye



4. 채팅방 나가기 채팅방에서 나가기 위한 #EXIT 명령어를 구현

#EXIT 현재 채팅방에서 떠남

구현 시 주의 사항

Peer 는 채팅 메시지를 전송하는 동작과 채팅 메시지를 읽는 동작이 동시에 동작할 수 있도록 Thread 를 이용해서 구현해야 한다.

```java
//Thread for concurrent usage.
Thread t = new Thread(new ReadThread(socket, group, port));
```

Thread Object 를 사용하여 커넥션을 담당하는 담당인 쓰레드를 생성하도록 한다.

```
message = name + ":" + message;
int count = 0;

byte[] buffer = null;
buffer = message.getBytes();
boolean allsent = false;

count =((buffer.length)/512)+1;

if(count > 2) {
    for (int i = 0; i <= count; i++) {
        if(i==count){
            byte[] packet = Arrays.copyOfRange(buffer, from: i * 512, buffer.length);
            DatagramPacket datagram = new
                    DatagramPacket(packet, length: 512, group.getAddress(), port);
            socket.send(datagram);
            break;
        }
        byte[] packet = Arrays.copyOfRange(buffer, from: i * 512, to: i * 512 + 512);
        DatagramPacket datagram = new
                DatagramPacket(packet, length: 512, group.getAddress(), port);
        socket.send(datagram);
    }
}
```

작성중인 메세지가 512 바이트 이상이라면, 버퍼 크기 / 512 + 1 으로 세그먼트를 내눠 for 문으로 순차적으로 송신하고 있다.

버퍼 자체는 512 바이트 이상이기에, copyOfRange 을 사용하여 512단위의 패킷을 소킷을 통해 보내는 방식을 차용하고 있다.

```
ReadThread(MulticastSocket socket, InetSocketAddress group, int port) {
    this.socket = socket;
    this.group = group;
    this.port = port;
}

@Override
public void run() {
    while (!Peer.finished) {
        byte[] buffer = new byte[ReadThread.MAX_LEN];
        DatagramPacket datagram = new
                DatagramPacket(buffer, buffer.length, group.getAddress(), port);
        String message;
        try {
            socket.receive(datagram);
            message = new
                    String(buffer, offset: 0, datagram.getLength(), charsetName: "UTF-8");
            if (!message.startsWith(Peer.name))
                System.out.println(message);
        } catch (IOException e) {
            System.out.println("Socket closed!");
        }
    }
}
```

수신 이외에 송신을 담당할 ReadThread 내부 클래스를 따로 만들어서, 들어오는대로 512 바이트 기준 데이터그램을 받아, 이를 stringify 하여 상대 , 즉 클라이어는 입장에서 메세지를 볼 수 이쓴 것이다.

** 사용 방법 **

javac Peer.java 로 테스트 하는 머신에서 컴파일을 한다.

java Peer (Port Number)로 해당 포트에 클라이언트로 접근한다.

#JOIN (chatroomname) (username) 으로 원하는 채팅방에 원하는 유저네임(중복 불가능) 으로 활동 할 수 있다.

나가고 싶다면 **#EXIT** 으로 채팅방 나온 후, 다시 **#JOIN (chatroomname) (username)** 으로 원하는 다른 방에 다른 이름으로 참여할 수 있다.