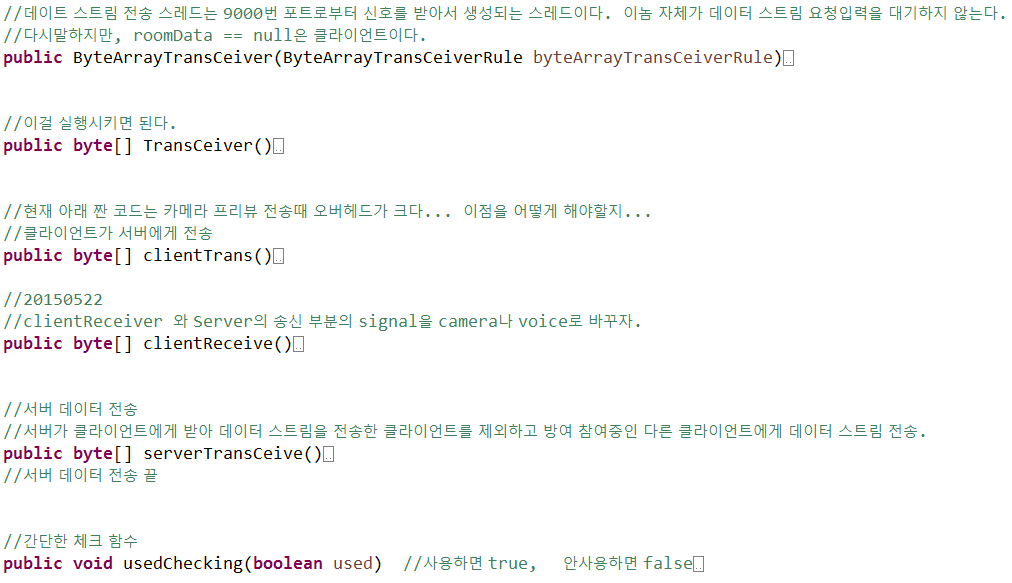
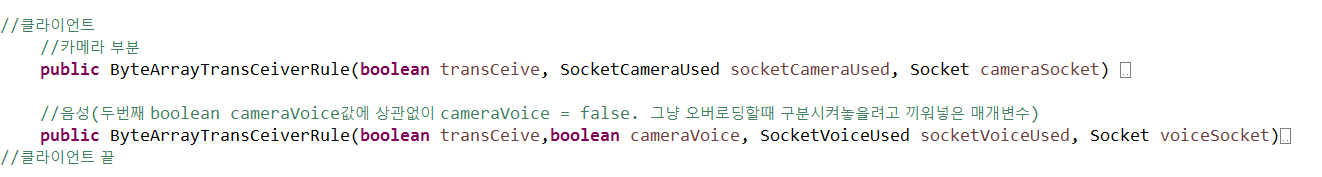
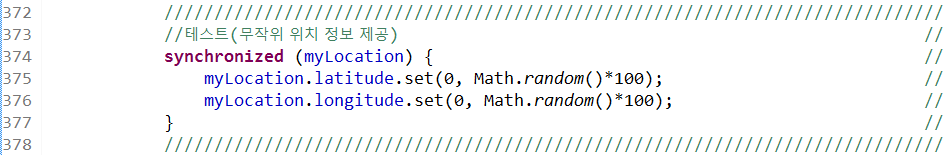
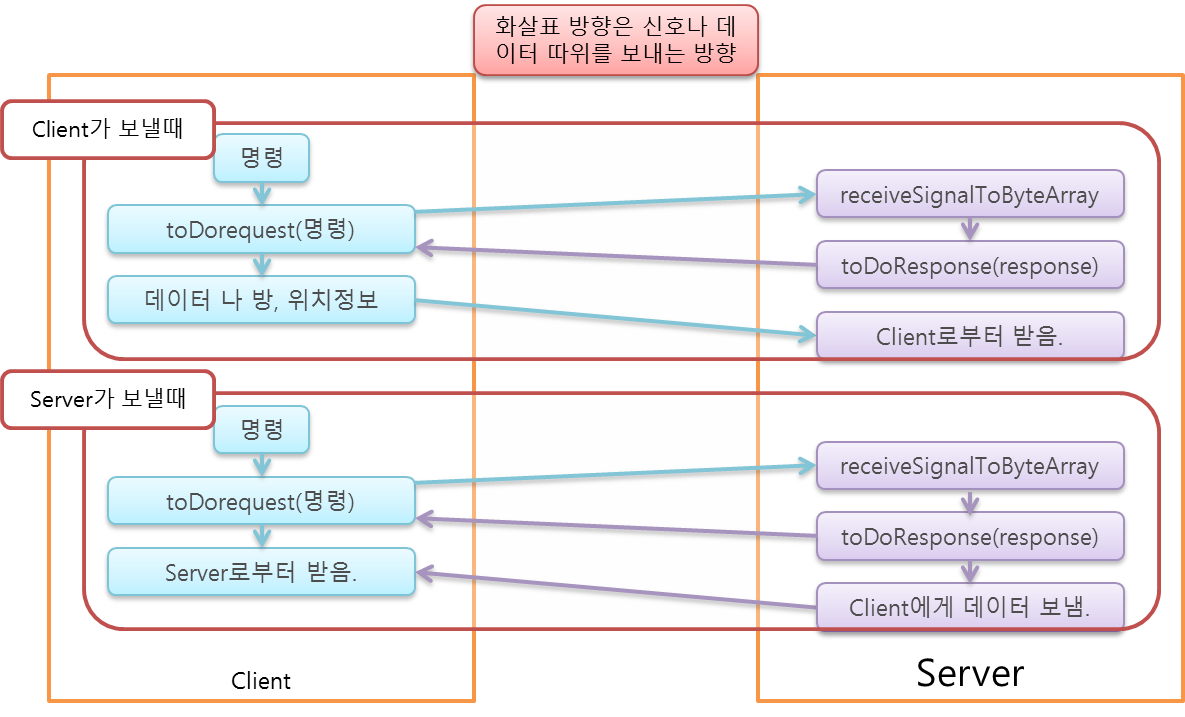
|  |  |
| --- | --- |
| 안드로이드에 이식할 때 관련되는 부분 설명서.  (혹시 GitHub 사용하면  <Https://github.com/sonsangjun/SocketTest.git> 에서 git clone (명령어임)로 받을 수 있음 받아주세요.  아니면 이클립스로 받아 주세요.) | |
| 안드로이드에 소스 포함시킬 때  **Server.java, ServerThread.java, MonitorThread.java을 제외한**  모든 파일을 포함 시키세요!! | |
|  | ByteArrayTransCeiver.java, ByteArrayTransCevierRule.java  └ 카메라, 음성 전송  Client.java └ 예제 클라이언트, ,Thread를 상속받고 있다. 안드로이드에서 클라이언트 만들 때 이걸 참조해 작성하면 된다..  LocationManage.java └ 위치 정보를 관리하는 클래스.  SignalData.java  └ 서버와 클라이언트 간의 통신약속을 구현한 클래스 그냥 어떤 메소드가 있는지만 확인하면 된다. 내부로직까지는 몰라도 된다.  SocketBroadCastThread.java, SocketBroadCastUsed.java └ 서버에서 날아오는 메시지 (채팅이나, 음성 전송등)를 받아 클라이언트에게 알려준다. (Thread) Used.java는 BroadCast 스레드를 죽일때만 사용한다.  SocketCameraUsed.java, SocketVoiceUsed.java SocketEventUsed.java └ 클라이언트가 서버로부터 데이터스트림을 받을 때 그 스트림의 성격에 따라 Used.java에서 사용여부를 체크해 락(Lock)(잠근다)을 건다. 이건 소켓이 이중 접속을 막기 위해서.  그외, Used.java에선 byte[] message를 통해 데이터 스트림을 공유 가능하다. SocketPushThread.java에서 받은 바이트 스트림 역시 공유가 가능하다. eventUsed는 eventSocket 락걸때 사용한다. 안드로이드에서 버튼을 자꾸 누르는 행위를 막을때도 사용할 듯 싶다.  Top.java ValueCollections.java └ 클라이언트 와 서버 클래스 객체를 만들어 스레드를 실행시킨다. (자세한 건 해당 소스 참고)  ValueCollection.java은 자주 사용하는 변수나, 스위치 같은 변수를 선언한 곳이다. ValueCollection을 수정하면 코드 전반적으로 영향을 미친다.  SocketPushThread.java SocketPushUsed.java └ Thread는 데이터 스트림 받기를 대기하는 스레드 이다. 서버에서 push(야 데이터 받아라)가 날라오면 이 스레드가 데이터 스트림을 받는다. 이 스레드가 데이터를 받는동안 Used.java에서 락을 건다. 다른 놈이 PushSocket을 못사용하게 막는것이다.  안드로이드 개발에선 위에 체크한 클래스만 살펴보면 된다. |

각 클래스 설명 (위의 순서대로 설명)  
  
ByteArrayTransCeiver.java  
  
  
  
ByteArrayTransCeiver()는 생성자. ByteArrayTransCeiverRule는 생성자의 매개변수(파라메터)다.  
TransCeiver() 해당 클래스에서 이걸 호출해야한다. 다른 함수는 이 함수의 엑스트라라 호출 할 필요가 없다.  
  
ClientTrans() 는 클라이언트가 서버로 전송하는 기능.  
ClientReceive() 는 클라이언트가 서버로부터 데이터를 받는 기능.  
ServerTransCeive() 는 클라이언트에게 받은 데이터를 다른 클라이언트에게 뿌리는 기능.  
usedChecking() 은 각 소켓의 사용에 대해 true(잠그고) false(열고) 는 기능.  
  
여기는 이런 함수가 있다는 것만 알고 있으면 된다.  
  
  
  
  
  
  
  
  
  
  
  
  
ByteArrayTransCeiverRule.java  
  
  
변수부분은 설명이 적혀있어 거의 생략.  
SocketVoiceUsed를 보면, 이것은 안드로이드 클래스와 공유하려면 (SocketCameraUsed 역시 해당된다)  
  
socketVoiceUsed = 받은 데이터;   
 이런 식으로 대입하지 말고,  
synchronized(socketVoiceUsed) {  
socketVoiceUsed.message = new byte[받은데이터.length];  
for(int i=0; 받은데이터.length; i++)  
 socketVoiceUsed.message[i] = 받은데이터[i];  
}  
 같은 코드를 집어넣어야 할거 같다.( 이런 식의 코드는 pushThread.java 추가하면 될 듯. 그 쪽에서 다시 언급함)  
  
클라이언트에서 해당 클래스 변수 선언시 생성자에 들어가야 하는 매개변수 목록. 고민할 거 없이 넣어달라는데로 넣어주면 된다.

Client.java

|  |  |
| --- | --- |
| 변수 명 | 설명 |
| **int** portNum = value.portNum;  **int** waitTime = value.waitTime;  String ServerIP = value.ServerIP; | 연결에 필요한 기본적인 변수값으로 ValueCollection을 통해 받는다.(port번호,대기시간(waitTime), ServerIP) |
| **byte**[] cameraByteArray = **null**;  **byte**[] voiceByteArray = **null**; | byte[] 두개는 데이터 스트림이 들어갈 자리다. |
| String roomName = **new** String(value.unname);  String yourName = **new** String(value.yourName); | roomName은 참가한 방이름 yourName은 너 너 이름. |
| SignalData signal; | signal은 서버와 통신하기 위해 약속을 선언한다. |
| ByteArrayTransCeiver byteArrayTransCeiver; //데이터 스트림 송수신 역할  ByteArrayTransCeiverRule cameraTransCeiver; //카메라 프리뷰 데이터 송수신  ByteArrayTransCeiverRule voiceTransCeiver; //음성 데이터 송수신 | |
| IntegerToByteArray integerToByteArray; | integerToByteArray는 int형 데이터크기를 byte[]로 변환시켜 준다. 이것은 서버에게 데이터 사이즈를 보내기 위해서 변환과정을 거치는 것이다. |
| Socket pushSocket;  Socket broadCastSocket;  Socket eventSocket;  Socket cameraSocket;  Socket voiceSocket; | socket변수는 서버와 통신할 각 소켓이다. 소켓의 포트번호는 각각 pushSocket ( 8998 ) broadCastSocket ( 8999 ) eventSocket (9000) cameraSocket ( 9001)  voiceSocket (9002) 이다. |
| BufferedInputStream eventInput;  BufferedOutputStream eventOutput; | signal을 전송하기 위해 입력을 받고(input) 신호를 출력(output) 하는 버퍼스트림을 선언한다. 초기화시 클라이언트에게 ID를 받을 때 사용한다. |
| SocketBroadCastThread socketBroadCastThread = **null**;  SocketPushThread socketPushThread = **null**; | BroadCastThread는 서버에게 메시지를 받고, PushThread는 서버에게 데이터 스트림을 받는다. |
| SocketPushUsed socketPushUsed = **new** SocketPushUsed();  SocketBroadCastUsed socketBroadCastUsed = **new** SocketBroadCastUsed();  SocketEventUsed socketEventUsed = **new** SocketEventUsed();  SocketCameraUsed socketCameraUsed = **new** SocketCameraUsed();  SocketVoiceUsed socketVoiceUsed = **new** SocketVoiceUsed(); | 각 소켓에 대해 락을 거는 클래스 |
| RoomDataToArray myLocation;  RoomDataToArray locationList = **null**; | myLocation은 당신의 위치를 저장하는 클래스 locationList는 서버에서 주기적으로 받는 방에 참여한 사람들의 위치정보List. |
| RoomData roomData; //방 목록을 받아오기 위해 선언  RoomManage roomManage; //방 관리하는 클래스 | |

메소드 설명  
public test\_II()  
테스트 메소드이다. 현재는 호출 자체가 안되니. 삭제해도 상관없다. (투명인간 취급)  
  
  
public void run();  
스레드 start()할 때 실행되는 메소드. 클라이언트의 초기화를 담당한다. 안드로이드에서 작성시 onCreate()같은 초기화 부분 구현할 때 참고하면 된다.  
  
receiveClientID()  
run()안에서 실행되며, 서버로부터 ID를 할당받는다.  
  
public void clientTerminate()  
run()안에서 실행되며 초기화 후에 본격적으로 실행되는 클라이언트 부분이다.  
   
 public void clientTerminate() 안에서 작동하는 메소드이다.  
 exitClient()  
 클라이언트가 연결이 끊어지거나, 서버와 연결을 종료하고 싶을 때 호출되는 메소드   
   
  
**clientTerminate() 부분에 이런식의 코드가 있는데, 이건 실제 안드로이드 구현할 때 진짜 위치 정보를 줘야 한다. 실제 위치 정보는 LocationManage에 넣어주면 된다.**  
  
클라이언트의 동작 방식은  


|  |  |  |
| --- | --- | --- |
| toDoRequset(명령)는 | toDoResponse(명령) | 를 순서대로 선언한 것과 같다. |
| toCatchResponse(signal.response) |

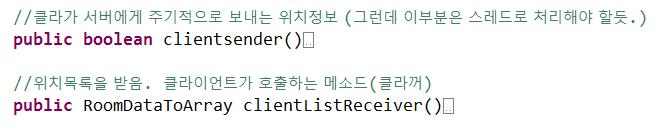
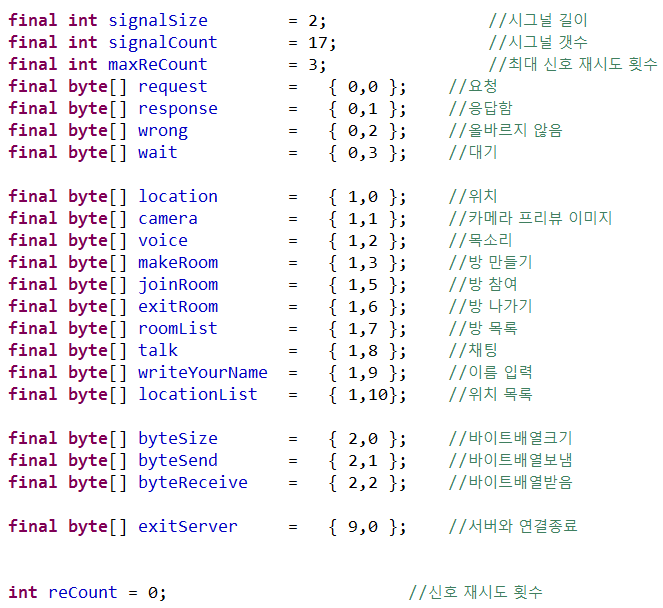
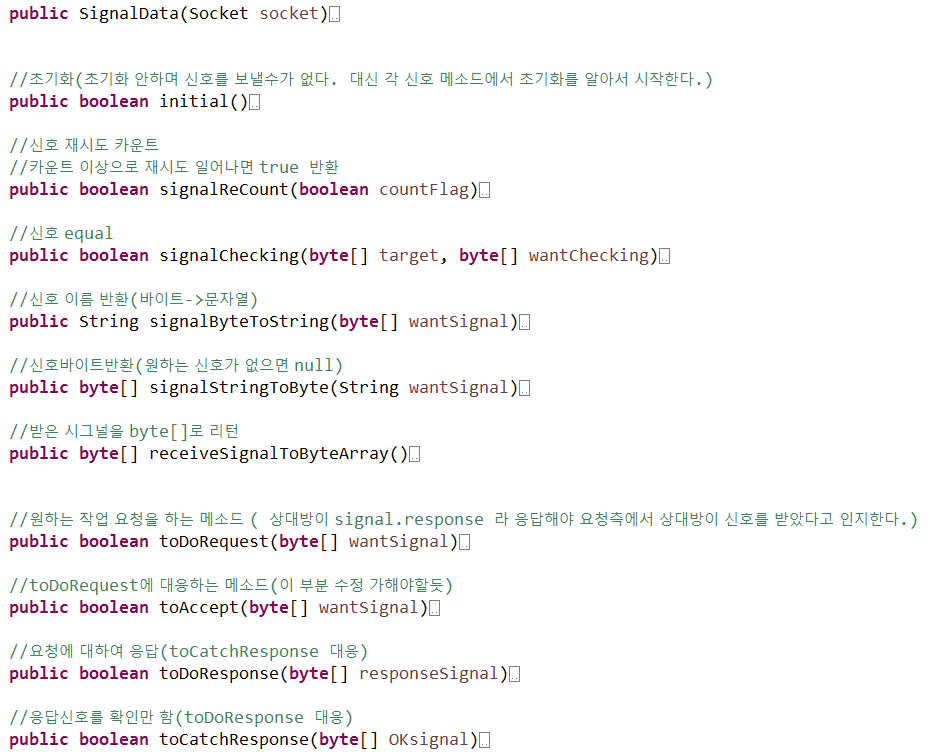
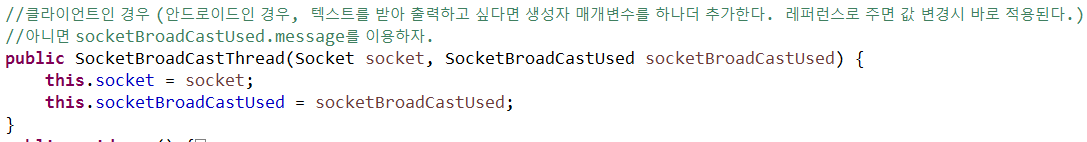
서버측에선 위와 같은 그림처럼 신호 약속을 하고 있으니 클라이언트는 위와 같은 도식도로 짜면 된다.  
예제 클라이언트의 동작방식은 Client의 clientTerminate()를 참고.  
  
LocationManage.java  
  
변수  
 ValueCollections value = **new** ValueCollections();

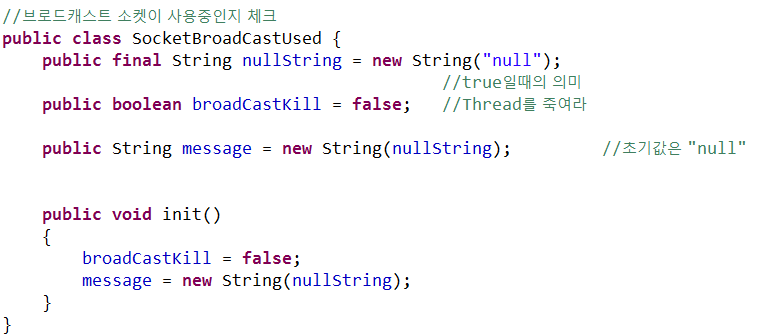
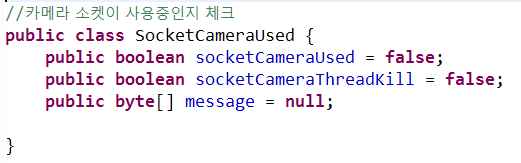
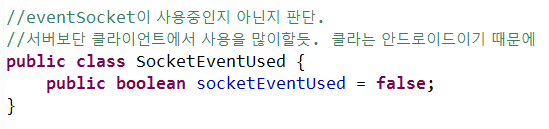
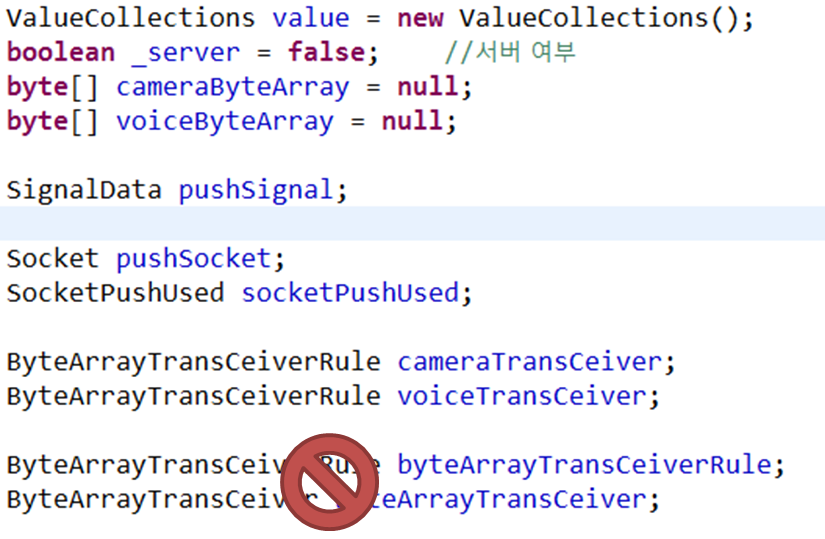
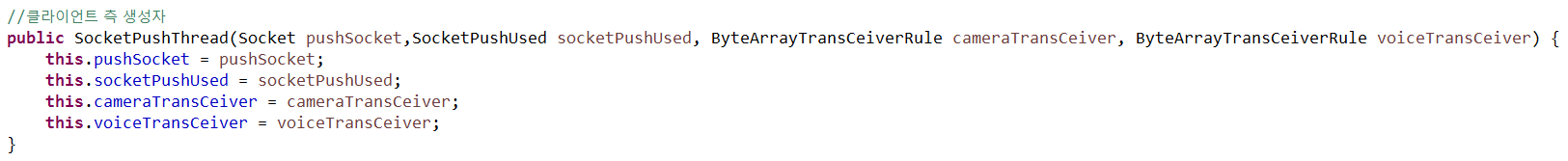
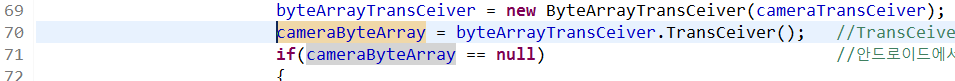
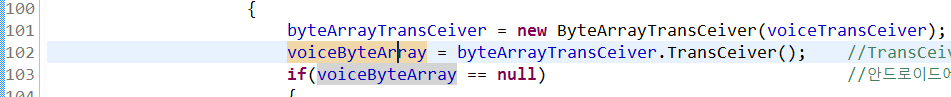
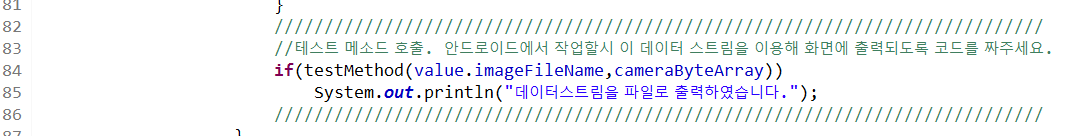
RoomData roomData = **null**;

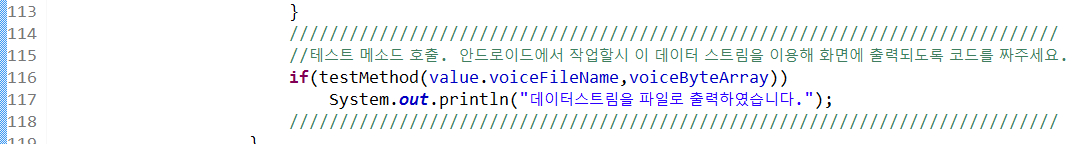
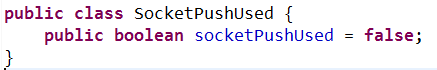
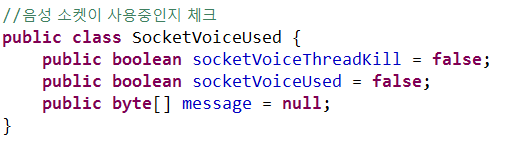
Socket eventSocket = **null**;

SocketEventUsed socketEventUsed;

RoomDataToArray locationList = **null**;

RoomDataToArray myLocation = **null**;  
Client.java와 겹치므로 그 부분 설명은 참고하자.  
  
클라이언트 생성자만 보면 된다.  
myLocation이나 locationList는 Client.java에서 선언후에 얘한테 넘겨줘야 한다. 안그러면… 안된다.  
  
서버측 메소드는 볼 필요없고  
  
클라이언트 메소드만 확인하면 된다.  
두번째 메소드가 반환하는 클래스는 서버로부터 받은 자기가 참여한 방 사람들의 위치 목록이다.  
(클래스 타입은 public RoomDataToArray(List<Integer> clientID, List<Double> latitude, List<Double> longitude) )  
  
  
  
  
  
  
  
  
  
  
  
  
  
SignalData.java  
  
  
  
  
  
메소드 중에 몇가지 부연 설명을 하면.  
  
toDoRequset는 서버에게 명령을 보내고 기다리는 응답은 response(byte[] {0,1}) 신호를 기다린다.  
toAccept는 서버에게 명령을 받고 자기가 wantSignal한거라면 응답으로 response를 보낸다.  
즉, 둘은 대응된다.  
  
toDoRequest(명령)는 toDoResponse(명령), toCatchResponse(response)로 분리할 수 있다.  
toAccept 역시 순서만 반대로( toCatchResponse(명령) -> toDoResponse(response)) 쓸 수 있다.  
 -> 굳이 분리해서 쓰는 이유라면, 신호를 보내고, 꼭 false를 반환 받지 않으면 무한 루프 같은 상황에 빠지는  
 경우를 막기 위해 쓰는 경우가 있다. 그외 if문 두번 쓰고 싶으면 쓰는거고.  
  
신호를 주고 받는 기능을 하는 메소드는 모두 boolean을 반환한다.  
boolean값이   
true인 경우는 당연히 신호전송이 잘 된경우고,  
false인 경우는 서버와 연결이 끊어졌거나, 서버가 올바르지 않은 신호라고 응답한다거나,  
Input/Output 예외가 발생(IOException)한 것이다.  
  
신호 🡨🡪 바이트 변환 예시  
signalByteToString(signal.response) 리턴값 “response”  
signalStringToByte(“response”) 리턴값 byte[] = {0,1};  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
SocketBroadCastThread.java  
  
변수는 Client.java 설명과 비슷하다. (생략)  
  


생성자는 클라이언트 것만 사용한다. 매개변수는 잘 알다시피 두개 넣으면 된다.  
락 걸기 위해 SocketBroadCastUsed를 넣는다.  
  
메소드는 run()뿐인데, 설명은 82Line부터 설명한다. ( 그 앞은 서버용이라 필요없다. )  
  
temp가 서버로부터 날아오는 메시지를 임시 저장하게 될 녀석이다.   
아무것도 안들어있기에 84Line에서 초기화 했다.  
86Line에서 서버로부터 메시지를 받으려고 버퍼와 서버와연결된 소켓을 이어준다.  
91Line에서 다른놈이 BroadCast못 사용하게 true로 막아버린다. 의미없는 코드지만 혹시 모르니 써둔다.  
(synchronized(클래스) 블록에 있으면 다른 스레드에서 ‘클래스’ 수정을 할 수 없다.)   
(synchronized는 동기화라 부르기도 한다.)  
  
98Line에서 소켓과 연결된 버퍼를 통해 메시지를 받는다.  
101Line에서 value.waitTime 동안 대기하고,  
106Line에서 socketBroadCastUsed를 동기화 시키고, 받은메시지를 출력한다.  
**이때, 안드로이드는 이 메시지를 화면에 출력할 수 있게 처리해야한다.**  
이후는 안봐도 된다.  
  
SocketBroadCastUsed.java  
  
  
  
SocketCameraUsed.java  
  
  
SocketEventUsed.java  
  
  
SocketPushThread.java  
  
  
위에 나온 변수만 클라이언트가 쓴다. (아래 두줄 빼고)  
  
카메라나 음성을 구분하기 위해서 cameraTransCeiver, voiceTransCeiver를 구분해 놓았다.  
그부분이 데이터 스트림이 들어간다. (내부에 socketVoiceUsed, socketCameraUsed가 있다.)  
  
메소드는  
run(), TestMethod() 두개가 있다.  
TestMethod()는 개발단계에서 데이터 스트림을 정상적으로 받았는지 확인하는 함수로서,  
개발이 끝나가면 빼야한다.  
  
  
  
실질적으로 run() 한 개라 봐도 무방하다.  
클라이언트 코드는 45Line ~ 136Line 이다.  
데이터를 받는 부분은  
  
camera (70Line, TransCeiver()가 byte[] 반환함)  
  
  
voice(102Line, TransCeiver()가 byte[] 반환함)  
  
  
문제는 byte[]은 이 스레드가 while(true) 반복을 시작하면 데이터가 사라지는 거 같다. 그래서 생성자 매개변수로 카메라나 음성을 받는 클래스를 하나 씩 더 받고, 그 클래스의 byte[] 배열에 for문을 이용하여 데이터를 집어 넣는 대안이 있다. (해당 문서 3page, ByteArrayTransCeiverRule.java 참고)  
  


  
데이터 스트림을 올바르게 받는지 확인하는 메소드(아까 언급한거)인데, 개발 중에 필요없으면 삭제하자.  
  
  
SocketPushUsed.java  
  
  
PushSocket 사용중이면 true, 미사용중이면 false  
  
SocketPushUsed.java  
  
  
  
  
  
Top.java  
  
main()가 위치한 테스트 클래스.  
해당 프로젝트를 run(컴파일,디버그)(실행) 시키면 Top의 main()가 호출되어 ValueCollection.java값에 따라  
서버 or 클라이언트로 작동되어 실행된다.  
안드로이드 개발할땐 거의 필요없는 클래스. 어차피 클라이언트 개발만 할 것이기 때문이다.  
  
  
Top.java

|  |  |
| --- | --- |
| 변수 명 | 설명 |
| **final** String upLine = **new** String("────");  **final** String downLine = **new** String("──"); | 콘솔창에서 깔끔하게 출력하기 위해 수평선을 만들었다. |
| **final** String yourName = **new** String("null");  **final** String unname = **new** String("unname"); //방에 참여안했을때, unname으로 할당 | |
| //final String ServerIP = "168.131.151.169"; //연구실 컴  //final String ServerIP = "168.131.153.170"; //DB서버 IP  **final** String ServerIP = "221.156.9.145"; //외부에서 테스트할때 IP  //final String ServerIP = "192.168.0.3"; //사설IP 서버 | |
| **final** **int** waitTime = 100; //정말 기다려야 하는 시간  **final** **int** coolTime = 10000; //위치 정보등을 받는 주기(interval)  **final** **int** portNum = 9000; //eventSocket 포트 번호, 이 번호가 8998 ~ 9002 의 중간번호  **final** **int** packetSize = 1; //한번에 얼마나 보낼지 (기본 1Byte)(이 변수 수정하지 말아요!) | |
| **final** **double** basicLatitude = 35.159773; //기본 위도  **final** **double** BasicLongitude = 126.851568; //기본 경도 | |
| **final** **boolean** \_Server = **false**; //코드가 서버로 작동하는 경우 true  **final** **boolean** \_ClientByteArray = **false**; //클라이언트가 데이터를 제공하는지.(Top.java테스트)  //데이터 스트림 전송 테스트를 하고 싶으면 아래 두 파일을  //20150503\_CommonSender폴더에 두고 이 값을 true해라.  **final** String imageFileName = "test.jpeg"; //카메라 이미지 전송이 파일 전송과 비슷하므로(테스트 변수)  **final** String voiceFileName = "test.mp3"; //음성 테스트 파일 | |

여기에 자주 사용하는 변수 값이나, 테스트 값을 설정해 놓았다.   
서버가 바뀔경우 위의 서버IP목록같이 한줄을 더 추가하고, 나머지 서버 IP는 주석처리 하면된다.  
waitTime이나 coolTime은 값 변경시 클라이언트의 퍼포먼스에 살짝 영향을 줄 수 있다.