2013-2학기 세종대학교 SSG 프로젝트 기술문서

Android 기반
익명 채팅 어플리케이션 및 서버 개발

2014.01.02

SSG

박영준(3) 송성빈(3)
박명수(1) 최인식(1)

목차

1. 개요

1.1 개발 동기 및 목적

학기 중 개설된 모든 강의에 대해 익명의 채팅 공간을 제공하여 강의평가, 교수평가 및 기타 강의에 대한 정보를 수강생들이 좀 더 자유롭고 실시간으로 주고받을 수 있도록 하는 것이 본 어플리케이션 개발의 목적이다.

1.2 개발 환경

분류	내용
OS	Windows7 32bit, Windows7 64bit
Development and debug tool	Eclipse + Android SDK
Language	Java

[표 1] 개발환경

1.3 개발 기간 및 일정

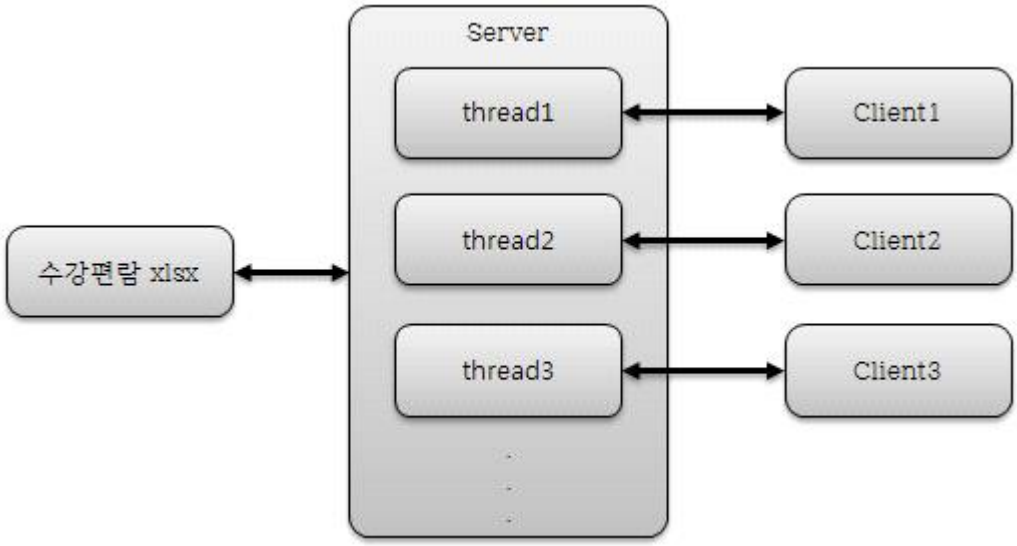
2013.09.23. ~ 2014.01.02.(102일)

Task	9월	10월	11월	12월
프로그램 설계 및 디자인				
브로드캐스트 채팅 구현				
서버와 수강편람 연동				
강의 목록 불러오기 기능				
강의별 채팅방 생성 기능				
각 강의별 채팅 구현				

[표 2] 개발 기간 및 일정

2. 소프트웨어 구조 및 설계

2.1 전체 소프트웨어 구조 및 동작 방식



[그림 1] 전체 프로그램 구조

클라이언트와 서버 두 가지 프로그램으로 구성되어 있으며, TCP/IP Socket통신을 사용하여 메시지 및 데이터를 주고받는다. 서버는 수강편람(엑셀파일)을 읽어서 강의 리스트를 생성하며, 클라이언트의 요청 시 클라이언트로 강의 리스트를 전송한다. 서버와 클라이언트 모두 멀티쓰레드로 구현되어있으며, 서버는 클라이언트가 접속할 때 마다 새로운 쓰레드를 생성하여 클라이언트와의 통신을 담당시킨다.

2.2 서버 구조 및 설계

서버가 실행되면 먼저 수강편람 파일로부터 강의정보를 파싱하여 리스트에 저장한다. 그 후 서버는 소켓을 생성하여 클라이언트의 접속을 기다린다. 클라이언트가 접속하면 새로운 소켓과 쓰레드를 만들어 클라이언트와의 통신에 할당한다. 클라이언트로부터 메시지를 수신하며, 수신된 메시지가 강의 목록 요청 메시지인 경우, 해당 클라이언트에 강의 목록을 전송한다. 수신된 메시지가 일반 채팅 메시지인 경우 연결된 모든 클라이언트에게 메시지를 전송한다. 서버를 구성하는 클래스는 다음과 같다.

클래스	기능
ChatServer	서버소켓을 생성하고 클라이언트의 접속을 기다린다. 클라이언트가 접속하면 클라이언트와 통신하기 위한 새로운 소켓과 쓰레드를 생성한다. 한 쓰레드로부터 받은 메시지를 생성된 모든 쓰레드에 전송하는 메시지 브로드캐스트 기능을 수행하며 클라이언트가 접속을 종료하면 생성했던 쓰레드를 삭제한다.
ServerThread	ChatServer 클래스의 내부 클래스로 클라이언트와 통신하는 역할을 수행한다. 클라이언트에 대한 입출력 스트림을 생성하며, 스트림으로부터 메시지를 읽어오거나 강의목록을 전송한다.
ExeclHandler	수강편람(엑셀 파일, *.xslx)을 읽어와 강의목록을 생성한다.
SubjectInfo	ExcelHandler로부터 파싱된 강의 정보를 저장하는 클래스

[표 3] 서버를 구성하는 클래스 및 기능

아래는 각 클래스별 메서드 및 생성자에 대한 설명이다.

■ ChatServer.java의 메서드

public ChatServer(int port)

```
public ChatServer(int port) {
    this.port = port;

    getSubjInfo();      // 강의 목록을 불러온다.

    // 쓰레드를 저장하는 백터 객체 생성
    v = new Vector<ServerThread>(10, 10);
    try {
            // 포트를 지정하여 서버소켓 생성
            serverSocket = new ServerSocket(port);
            System.out.println("서버가" + port + "에서 접속 대기중");

            // 클라이언트의 접속을 항상 받아들일 수 있도록 무한루프를 돌림
            while (true) {
                // 클라이언트의 접속을 기다림
                s = serverSocket.accept();

                // 클라이언트가 접속하면 쓰레드를 만들고
                // 쓰레드 객체의 생성자로 서버 소켓과 클라이언트 소켓을 넘겨줌
                ServerThread serverThread = new ServerThread(this, s);

                // 생성된 쓰레드 객체를 벡터에 추가한다
                this.addThread(serverThread);
```

```
                // 쓰레드 시작
                serverThread.start();
            }
        } catch (IOException ioe) {
            // 예외에 대한 처리
        }
}
```

ChatServer 클래스의 생성자이다. 서버 실행 시 가장 먼저 실행되는 코드이며 getSubjInfo() 메서드를 사용하여 강의 목록을 불러온다. 매개변수로 전달받은 port번호로 port 변수를 초기화 시킨다. 초기화된 port번호로 소켓을 생성한 뒤 클라이언트의 접속을 기다린다. 클라이언트가 접속하면 새로운 쓰레드를 생성하여 현재 ChatServer객체와 생성된 소켓(s)를 매개변수로 넘긴다. 생성된 쓰레드를 벡터 리스트에 추가한 뒤 쓰레드를 실행시킨다.

[표 4] ChatServer 생성자

public void getSubjInfo()

```
public void getSubjInfo() {
    System.out.println("강의 목록을 불러오는중...");
    subjInfoList = ExcelHandler.ExcelParser();

    for( int i = 0; i < subjInfoList.size(); i++ ) {
        System.out.print(subjInfoList.get(i).getSubjectName() + " / ");
        System.out.print(subjInfoList.get(i).getSubjectNo() + " / ");
        System.out.println(subjInfoList.get(i).getClassNo());
    }
    System.out.println(subjInfoList.size() + "개의 강의 불러오기 성공!\n");
}
```

ExcelHandler 클래스의 static 메서드인 ExcelParser()를 사용하여 엑셀로부터 강의 정보를 불러와 subjInfoList 객체에 담는다. subjInfoList는 ArrayList<SubjectInfo>로 정의된 리스트이다.

[표 5] getSubjInfo 메서드

public void addThread(ServerThread st)

```
public void addThread(ServerThread st) {
    v.addElement(st);
}
```

생성된 쓰레드(st)를 벡터(v)에 추가한다.

[표 6] addThread 메서드

public void removeThread(ServerThread st)

```
public void removeThread(ServerThread st) {
    v.removeElement(st);
}
```

쓰레드(st)를 벡터(v)에서 제거한다.

[표 7] removeThread 메서드

public void broadCast(String classId, String msg)

```
public void broadCast(String classId, String msg) {
    for( int i = 0; i < v.size(); i++ ) {
        // 벡터에 등록된 모든 쓰레드를 얻어온다.
        ServerThread st = v.elementAt(i);

        // classId가 일치하는 쓰레드만
        // if( st.classId.equals(classId))
        st.sendMessage(msg); // 각 쓰레드에 메시지 전송
    }
}
```

반복문을 사용하여 벡터에 등록된 모든 쓰레드에게 메시지를 전송한다. classId 변수의 경우 사용자가 입장한 채팅방을 구분하기 위한 용도로 사용할 예정이었으나 해당 기능을 구현하지 못해 지금은 사용되지 않는 변수이다.

[표 8] broadCast 메서드

public static void main(String[] args)

```
public static void main(String[] args) {
    new ChatServer(9999);
}
```

서버 프로그램의 main메서드로 ChatServer객체를 생성하여 서버를 실행한다.

[표 9] main 메서드

■ ServerThread.java의 메서드

public ServerThread(ChatServer server, Socket s)

```
public ServerThread(ChatServer server, Socket s) throws IOException {
    this.server = server;
    this.s = s; // 클라이언트와 통신할 수 있는 소켓 정보를 s에 저장함.

    // 클라이언트와 메시지를 주고 받기 위한 입출력 스트림을 가져온다
    // 객체 출력 스트림
    this.oos = new ObjectOutputStream( s.getOutputStream() );
    oos.flush();
    this.br = new BufferedReader( new InputStreamReader(s.getInputStream(),
        "EUC-KR")); // 입력 스트림
    this.pw = new PrintWriter( s.getOutputStream(), true); // 출력 스트림

    // 클라이언트 에서 IP 정보를 얻어 출력한다.
    System.out.println(s.getInetAddress() + "에서 접속함");
}
```

ServerThread 클래스의 생성자로, 매개변수로 전달받은 소켓으로부터 클라이언트와 통신하기 위한 입출력 스트림을 생성한다. ObjectOutputStream은 강의 리스트 객체를 클라이언트로 전송하기 위한 객체 출력 스트림이다. BufferedReader와 PrintWriter는 클라이언트와 메시지를 주고 받기 위한 입출력 스트림이다. 스트림 생성이 완료되면 접속한 클라이언트의 IP룰 출력하여 클라이언트의 접속을 알린다.

[표 10] ServerThread 생성자

public void sendMessage(String str)

```
public void sendMessage(String str) {
    pw.println(str);
}
```

생성자에서 생성된 PrintWriter객체(pw)를 통해 클라이언트에게 메시지(str)를 전송한다.

[표 11] sendMessage 메서드

public void run()

```
public void run() {
    //클라이언트로부터 받은 데이터를 클라이언트에게 송신한다.
    try {

        while((message = br.readLine()) != null ) {
            // 클라이언트로부터 강의 목록 요청이 들어오면 강의 목록을 전송한다.
```

```
                if(message.equals("REQUEST SUBJECT LIST"))
                    oos.writeObject(subjInfoList);
                // 일반 메시지의 경우 모든 클라이언트에게 뿌려줌
                else
                    server.broadCast("12345", message);
            }
        } catch (IOException ioe) {
            System.out.println(ioe);
            server.removeThread(this);
        } finally {
            System.out.println(s.getInetAddress() + "접속 연결 종료");
            server.removeThread(this);
            try {
                s.close();
            } catch (IOException ioe) {
                System.out.println(ioe);
            }
        }
}
```

쓰레드의 실행 메서드. 클라이언트의 메시지를 기다리며 메시지가 수신되면 메시지에 따라 처리한다. 수신 받은 메시지가 "REQUEST SUBJECT LIST" 문자열과 일치 하는 경우는 클라이언트가 강의 목록을 요청한 것이므로 ObjectOutputStream의 writeObject메서드를 사용하여 강의 리스트(subjInfoList)를 클라이언트에게 전송한다. 강의 목록 요청이 아닌 경우는 일반적인 대화 메시지인 경우밖에 없으므로 ChatServer 클래스의 broadCast메서드를 사용해 수신받은 메시지를 연결된 모든 클라이언트에게 전송한다.

[표 12] run 메서드

■ ExcelHandler.java의 메서드

public static ArrayList<SubjectInfo> ExcelParser()

```
public static ArrayList<SubjectInfo> ExcelParser() {
    File file = new File("./data/201302.xlsx");
    XSSFWorkbook wb = null;

    try {
        wb = new XSSFWorkbook(new FileInputStream(file));
    } catch (Exception e) {
        e.printStackTrace();
    }

    // 강의 정보를 담기위한 리스트
    ArrayList<SubjectInfo> subjInfoList = new ArrayList<SubjectInfo>();
    SubjectInfo si = null;

    for (Row row : wb.getSheetAt(0)) {
        si = new SubjectInfo();
        String[] temp = new String[32];
        int count = 0;

        for (Cell cell : row) {
            temp[count] = cell.getRichStringCellValue().getString();
            count++;
        }
        si.setSubjectName(temp[4]);       // 과목명
        si.setSubjectNo(temp[2]);         // 학수번호
        si.setClassNo(temp[3]);           // 분반
        subjInfoList.add(si);
    }
    return subjInfoList;
}
```

Apache POI 라이브러리를 사용해 수강편람(엑셀 파일)을 읽어온다. 과목명, 학수번호, 분반 셀의 값을 추출하여 SubjectList 객체에 저장한 뒤 리스트에 이 객체를 추가한다. 생성된 리스트를 반환한다. static 메서드이기 때문에 별도의 객체 생성 없이 접근이 가능하다. Java에는 기본적으로 엑셀파일(xls)를 열기위한 Jxl 라이브러리가 존재하지만 엑셀2007(xlsx)부터는 읽지 못한다. 그렇기 때문에 xlsx 파일을 읽을 수 있는 외부 라이브러리를 사용하게 되었으며, Apache POI 라이브러리는 http://poi.apache.org/ 에서 다운로드 받을 수 있다.

[표 13] ExcelParser 메서드

■ SubjectInfo.java

SubjectInfo.java

```
package ssg.chat;
import java.io.Serializable;

// 과목 정보를 저장할 클래스
public class SubjectInfo implements Serializable {

    private String subjectName;          // 과목명
    private String subjectNo;            // 학수번호
    private String classNo;              // 분반

    public String getSubjectName() {
        return subjectName;
    }

    public void setSubjectName(String sName) {
        this.subjectName = sName;
    }

    public String getSubjectNo() {
        return subjectNo;
    }

    public void setSubjectNo(String sNo) {
        this.subjectNo = sNo;
    }

    public String getClassNo() {
        return classNo;
    }

    public void setClassNo(String cNo) {
        this.classNo = cNo;
    }
}
```

강의 정보를 저장하기 위한 클래스로 과목명을 저장하는 subjectName필드와 학수번호를 저장하는 subjectNo필드, 분반을 저장하는 classNo 필드가 선언되어 있고 각 필드에 대한 getter, setter 메서드가 정의되어 있다. 서버에서 강의 목록을 전송할 때 해당 객체를 입출력해야 하므로 Serializable 인터페이스를 구현하여 클래스를 작성했다.

[표 14] SubjectInfo 클래스

2.3 안드로이드 클라이언트 구조 및 설계

클라이언트는 총 3개의 액티비티(화면)로 구성되어 있다.

액티비티	설명
ChattingRoomListActivity	사용자가 강의 목록으로부터 추가한 채팅방 리스트를 출력한다. 채팅방 중 하나를 선택하면 해당 채팅방으로 입장한다.
SubjectListActivity	현재 개설중인 모든 강의 목록을 출력한다. 원하는 강의를 선택하여 채팅방 리스트에 추가할 수 있다.
ChattingActivity	채팅방에 입장했을 때 보여지는 화면이다. 학수번호와 분반을 이용해 채팅방 고유의 ID가 생성되며 현재 ID와 같은 방에 있는 사용자들끼리 메시지를 주고 받는다. 접속한 사용자의 ID(0~9999)가 임의로 할당된다. 채팅방ID와 디바이스ID, 사용자ID, 메시지를 조합하여 서버로 전송한다. 다른 클라이언트가 보낸 메시지를 파싱하여 화면에 적절하게 출력한다.

[표 15] 액티비티 구성 및 설명

아래는 각 액티비티를 구성하는 java코드와 xml코드에 대한 설명이다.

■ ChattingRoomListActivity

(1) 자바 코드

ChattingRoomListActivity.java

```
public class ChattingRoomListActivity extends Activity implements
OnItemClickListener {

    ArrayList<SubjectInfo> subjInfoList;
    ArrayList<HashMap<String,String>> subjectList;
    HashMap<String,String> subjInfo;
    SimpleAdapter sAdapter;
    ListView subjectListView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.subject_list);

        // 액션바 설정
        ActionBar actionbar = getActionBar();
        actionbar.setTitle("SSG 익명 채팅");
        actionbar.setSubtitle("강의 목록");
```

```
}

// 채팅방 입장
@Override
public void onItemClick(AdapterView<?> parentView, View view, int position,
long id) {
    Intent intent = new Intent(ChattingRoomListActivity.this,
    ChattingActivity.class);
    intent.putExtra("subjectName", subjInfoList.get(position).getSubjectName());
    intent.putExtra("subjectNo", subjInfoList.get(position).getSubjectNo());
    intent.putExtra("classNo", subjInfoList.get(position).getClassNo());
    startActivity(intent);
}

// 메뉴 생성
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main_menu, menu);
    return super.onCreateOptionsMenu(menu);
}

// 메뉴 아이템 클릭시
public boolean onOptionsItemSelected(MenuItem item) {
    Intent intent;
    switch(item.getItemId()) {
    case R.id.add_subject:
        intent = new Intent(ChattingRoomListActivity.this,
                SubjectListActivity.class);
        startActivity(intent);
        return true;

    default:
        return false;
    }q
}

public void onResume() {
    super.onResume();

    subjInfoList = new ArrayList<SubjectInfo>();
    subjectList = new ArrayList<HashMap<String,String>>();

    // 파일로부터 채팅 목록을 읽어온다.
    FileInputStream fis = null;
    ObjectInputStream ois = null;

    try {
        fis = openFileInput("subject.lst");
```

```
                ois = new ObjectInputStream( fis );
                subjInfoList = (ArrayList<SubjectInfo>) ois.readObject();

                for( int i = 0; i < subjInfoList.size(); i++ ) {
                    subjInfo = new HashMap<String, String>();
                    subjInfo.put("title", subjInfoList.get(i).getSubjectName());
                    subjInfo.put("info", subjInfoList.get(i).getSubjectNo() + " / " +
                    subjInfoList.get(i).getClassNo());
                    subjectList.add(subjInfo);
                }

                ois.close();
                fis.close();

            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (StreamCorruptedException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }

            // 채팅 목록을 리스트뷰에 출력한다.
            sAdapter = new SimpleAdapter(this, subjectList,
            android.R.layout.simple_list_item_2,
            new String[]{"title", "info"},
            new int[]{android.R.id.text1, android.R.id.text2});

            subjectListView = (ListView)findViewById(R.id.subject_list);
            subjectListView.setAdapter(sAdapter);
            subjectListView.setOnItemClickListener(this);

            // 채팅 목록이 없을 때 보여줄 뷰 설정
            View emptyView = (TextView)findViewById(R.id.empty);
            subjectListView.setEmptyView(emptyView);
        }
    }
```

onCreate() : 액티비티가 실행되면 가장 먼저 호출되는 콜백 메서드로 여기서는 액션바의 타이틀 및 서브타이틀을 설정한다.

onResume() : onCreate() 메서드가 호출된 후 호출되는 메서드로 액티비티가 화면으로 보일 때 실행된다. 파일에 저장된 채팅방 리스트를 읽어와 리스트뷰에 연결시켜 리스트 형식으로 출력한다. 추가된 채팅방이 없는 경우 보여줄 뷰도 설정한다. 이 부분을 onCreate() 메서드가 아닌 onResume()에 작성한 이유는 강의 목록으로부터 채팅방이 추가된 후 다시 채팅방 리스트 액티비티로 돌아올 때 추가된 채팅방 목록을 바로바로 갱신

하여 보여주기 위해서이다.

onCreateOptionsMenu() : 상단 액션바에 메뉴를 추가하는 메서드이다. 여기서는 강의 목록 액티비티로 이동하기 위한 '채팅방 추가' 메뉴를 생성한다.

onOptionsItemSelected() : 메뉴를 선택했을 때 실행되는 메서드이다. 여기서는 '채팅방 추가' 메뉴 하나밖에 없으므로, 메뉴를 선택하면 강의목록 액티비티인 SubjectListActivity를 화면에 띄운다.
onItemClick() : 리스트뷰의 한 항목(item)을 선택했을 때 실행되는 메서드이다. 리스트뷰에 채팅방 목록이 출력되므로 여기서는 하나의 채팅방을 선택했을 때 실행되는 코드를 담고 있다. 선택한 채팅방 액티비티(ChattingRoomActivity)를 화면에 띄우며 Intent의 putExtra() 메서드를 통해 과목명, 학수번호, 분반 값을 띄우려는 액티비티로 전달한다.

[표 16] ChattingRoomListActivity

(2) XML 코드

subject_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/subject_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <TextView
        android:id="@+id/empty"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center"
        android:textColor="#5f5f5f"
        android:textSize="15sp"
        android:text="채팅 목록이 없습니다" />

</LinearLayout>
```

ChattingRoomListActivity의 화면을 구성하는 레이아웃 코드이다. 전체 레이아웃은 리니어 레이아웃으로 구성하고 그 안에 ListView와 TextView를 배치하였다. ListView는 채팅방 목록을 출력하기 위한 뷰이고 밑의 TextView는 리스트가 비었을 때 나타낼 뷰이다. 리스트가 존재하면 TextView는 화면에 보이지 않는다.

[표 17] subject_list.xml

■ SubjectListActivity

(1) 자바코드

SubjectListActivity.java

```
public class SubjectListActivity extends Activity implements OnItemClickListener {
    // 소켓 통신을 위한 객체
    Socket socket;
    BufferedWriter networkWriter;
    BufferedReader networkReader;
    ObjectInputStream ois;

    // 강의 목록을 담기위한 리스트
    ArrayList<SubjectInfo> subjInfoList;
    ArrayList<HashMap<String,String>> subjectList;
    ListView subjectListView;
    SimpleAdapter sAdapter;

    ProgressDialog pd;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.add_subject);

        ActionBar actionbar = getActionBar();
        actionbar.setTitle("강의 추가");
        actionbar.setDisplayHomeAsUpEnabled(true);

        // 서버에 강의 목록을 요청한다.
        new SetSocket().execute();
        new ReqSubjList().execute();

        subjInfoList = new ArrayList<SubjectInfo>();
        subjectListView = (ListView)findViewById(R.id.add_subject_list);
    }

    // 메뉴 생성
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.add_subject_menu, menu);
        MenuItem item  = menu.findItem(R.id.search_subject);
        SearchView searchView = (SearchView)item.getActionView();

        return super.onCreateOptionsMenu(menu);
    }

    // 메뉴 아이템 클릭시
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
        switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            return true;

        default:
            return false;
        }
    }

    /*
    * 서버와 연결을 맺기 위한 클래스
    */
    public class SetSocket extends AsyncTask<Void, Void, Void> {
        @Override
        protected Void doInBackground(Void... params) {
            try {
                // 소켓 연결
                socket = new Socket(getString(R.string.server_ip),
                                    Integer.parseInt(getString(R.string.server_port)));

                // 소켓으로부터 InputSteam과 OutputStream을 가져온다.
                // 한글이 깨지는것을 막기 위해 인코딩을 EUC-KR로 설정한다.
                networkWriter = new BufferedWriter(new
                OutputStreamWriter(socket.getOutputStream(), "EUC-KR"));
                networkReader = new BufferedReader(new
                InputStreamReader(socket.getInputStream(), "EUC-KR"));
                ois = new ObjectInputStream(socket.getInputStream());

                } catch (UnknownHostException e) {
                    e.printStackTrace();
                    Log.e("socket", e.getMessage());
                } catch (IOException e) {
                    Log.e("socket", e.getMessage());
                    e.printStackTrace();
                }
                return null;
            }
        }

    /*
     * 서버로부터 강의 목록을 요청한다.
     */
    class ReqSubjList extends AsyncTask<Void, Void, Void> {
        protected void onPreExecute() {
            pd = new ProgressDialog(SubjectListActivity.this);
            pd.setMessage("강의 목록을 가져오는 중입니다...");
            pd.setCancelable(true);
```

```
        pd.setIndeterminate(true);
        pd.show();
    }

    @SuppressWarnings("unchecked")
    @Override
    protected Void doInBackground(Void... params) {
        PrintWriter out = new PrintWriter(networkWriter, true);
        String requestMsg = "REQUEST SUBJECT LIST";
        out.println(requestMsg);

        try {
            subjInfoList = (ArrayList<SubjectInfo>)ois.readObject();
        } catch (OptionalDataException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return null;
    }

    @Override
    protected void onPostExecute(Void result){
        pd.dismiss();

        // 서버로부터 받은 리스트를 리스트뷰에 출력한다.
        HashMap<String,String> subjInfo;
        subjectList = new ArrayList<HashMap<String,String>>();

        for(int i = 0; i < subjInfoList.size(); i++ ) {
            subjInfo = new HashMap<String,String>();
            subjInfo.put("title", subjInfoList.get(i).getSubjectName());
            subjInfo.put("info", subjInfoList.get(i).getSubjectNo() + " / " +
            subjInfoList.get(i).getClassNo());
            subjectList.add(subjInfo);
        }

        sAdapter = new SimpleAdapter(SubjectListActivity.this, subjectList,
        android.R.layout.simple_list_item_2,
        new String[]{"title", "info"}, new int[]{android.R.id.text1,
        android.R.id.text2});

        subjectListView.setAdapter(sAdapter);

        subjectListView.setOnItemClickListener(SubjectListActivity.this);
```

```
    }
}

@Override
public void onItemClick(AdapterView<?> parent, View view, final int position,
long id) {
    String content = subjInfoList.get(position).getSubjectName();
    content += "\n학수번호 : " + subjInfoList.get(position).getSubjectNo();
    content += "\n분반 :  " + subjInfoList.get(position).getClassNo();

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("선택한 과목을 채팅목록에 추가하시겠습니까?");
    builder.setMessage(content);
    builder.setCancelable(true);
    builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            ArrayList<SubjectInfo> addedSubjList = new
            ArrayList<SubjectInfo>();

            // 기존에 추가한 채팅 목록이 있으면 파일로부터 기존 목록을 가져온다.
            FileInputStream fis;
            ObjectInputStream ois;
            try {
                fis = openFileInput("subject.lst");
                ois = new ObjectInputStream( fis );
                addedSubjList = (ArrayList<SubjectInfo>) ois.readObject();
            } catch (FileNotFoundException e1) {
                e1.printStackTrace();
            } catch (StreamCorruptedException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }

            // 목록에 새로운 채팅을 추가하여 파일에 저장한다.
            FileOutputStream fos;
            ObjectOutputStream oos;
            try {
                fos  = openFileOutput("subject.lst", MODE_PRIVATE);
                oos = new ObjectOutputStream( fos );
                addedSubjList.add(subjInfoList.get(position));
                oos.writeObject(addedSubjList);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
```

```
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    Toast.makeText(SubjectListActivity.this,
                    subjInfoList.get(position).getSubjectName()
                     + " 과목이 추가되었습니다", 0).show();

                    SubjectListActivity.this.finish();
                }
            });
            builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });
            AlertDialog ad = builder.create();
            ad.show();
        }
}
```

onCreate() : 서버로부터 강의 목록을 받아오기 위해 SetSocket 클래스로 서버와 소켓통신을 수립하고 ReqSubjList 클래스로 요청 메시지를 보낸다. 안드로이드에서는 메인쓰레드에서 네트워크 관련 작업을 할 수 없으므로 두 클래스 모두 안드로이드에서 제공하는 쓰레드의 일종인 AsyncTask 클래스를 상속받아 구현하였다. 별도의 쓰레드를 만들어 서버와 통신한다. 강의정보 목록을 담기 위한 subjInfoList를 초기화 한다.

onItemClick() : 강의 목록 중 특정 강의를 선택하면 실행되는 메서드이다. 선택된 강의를 채팅방 리스트에 추가할 것인지를 묻는 다이얼로그를 띄우며 'yes'를 선택하면 선택된 강의를 채팅방 목록에 추가한다. 채팅방 목록은 파일로 저장하여 어플리케이션을 종료해도 다음에 다시 실행하면 추가된 채팅방 목록이 유지되도록 했다.

SetSocket 클래스 : 서버와 통신하기 위한 소켓을 생성하고 입출력 스트림을 생성한다.

ReqSubjList 클래스 : 서버로부터 강의 목록을 요청하고 수신 받은 목록을 리스트에 저장한다. 저장된 리스트를 리스트뷰에 연결하여 강의 목록을 화면에 출력한다.

[표 18] SubjectListActivity.java

(2) XML 코드

add_subject.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
```

```
        android:id="@+id/add_subject_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

강의 목록을 나타내기 위한 리스트뷰 하나로 간단하게 구성되어있다.

[표 19] add_subject.xml

■ ChattingActivity

(1) 자바 코드

ChattingActivity.java

```
public class ChattingActivity extends Activity {
    Socket socket;                      // 서버와 통신하기 위한 소켓
    BufferedReader networkReader;
    BufferedWriter networkWriter;
    Handler mHandler;                   // 쓰레드와 통신하기 위한 핸들러

    EditText edMsg;                     // 사용자 입력 메시지
    Button btnSend;                     // 메시지 전송 버튼
    ListView lvChatListView;            // 채팅화면 리스트뷰

    ArrayList<MessageData> alMsgDataList;      // 채팅 내용을 담을 리스트
    // 채팅화면(리스트뷰)과 채팅 내용 리스트를 연결시켜줄 어뎁터
    MultiLayoutAdapter mlaChatListAdapter;


    TelephonyManager teleMgr;           // 디바이스의 ID를 얻기 위한 객체
    String deviceId;                    // 디바이스의 ID
    int userId;                         // 사용자의 ID
    String chatRoomId;                  // 채팅방 ID

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // 전달받은 데이터 받기
        Intent intent = getIntent();

        // 채팅방 ID 설정
        chatRoomId = intent.getStringExtra("subjectNo")
            + intent.getStringExtra("classNo");

        ActionBar actionbar = getActionBar();
        actionbar.setTitle(intent.getStringExtra("subjectName"));
```

```
actionbar.setSubtitle(intent.getStringExtra("subjectNo")
    + " / " + intent.getStringExtra("classNo"));

mHandler = new Handler(); // RecvThread와 통신하기위한 핸들러

// 디바이스의 ID를 가져온다.
teleMgr = (TelephonyManager)getSystemService(
    Context.TELEPHONY_SERVICE);
deviceId = teleMgr.getDeviceId();

// 유저 ID를 랜덤하게 생성
Random random = new Random();
userId = random.nextInt(9999);

edMsg = (EditText) findViewById(R.id.msg);                  // 메시지 입력 창
btnSend = (Button) findViewById(R.id.send);                 // 전송버튼
lvChatListView = (ListView) findViewById(R.id.chat_list);   // 채팅화면

// 리스트뷰 아이템 클릭시 색상이 바뀌지 않도록 빈 selector를 설정해준다.
lvChatListView.setSelector(R.drawable.listview_selector);

// 채팅 목록을 담을 Array
alMsgDataList = new ArrayList<MessageData>();

// ListView에 Array를 연결해주기위한 ArrayAdapter 생성
mlaChatListAdapter = new MultiLayoutAdapter(this, alMsgDataList);

// ListView에 ArrayAdapter 설정
lvChatListView.setAdapter(mlaChatListAdapter);

// 버튼 클릭 이벤트
btnSend.setOnClickListener(new OnClickListener() {

    public void onClick(View v) {
        if( !edMsg.getText().toString().equals("") ){
                // 버튼 클릭시 메시지를 보내는 AsyncTask(Thread) 실행
                new SendMsg().execute();
                // 메시지 전송 후 입력창을 비움
                edMsg.setText("");
        }
    }
});

/* 텍스트 변경 이벤트
        : 입력창에 텍스트가 없으면 전송버튼을 비활성화 하고
        : 텍스트가 있으면 활성화한다.
*/
edMsg.addTextChangedListener(new TextWatcher(){
```

```
            @Override
            public void afterTextChanged(Editable s) {
                if( edMsg.getText().toString().equals("") ){
                    btnSend.setEnabled(false);
                } else {
                    btnSend.setEnabled(true);
                }
            }

            @Override
            public void beforeTextChanged(CharSequence s, int start, int count,
                int after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before,
            int count) {
            }
        });

        // 서버와 소켓 연결
        new SetSocket().execute();
    }

    // 채팅창이 종료될 때 실행되는 onDestroy 메서드
    public void onDestroy(){
        super.onDestroy();
        // 소켓 연결 종료
        try {
            if( networkReader != null ) networkReader.close();
            if( networkWriter != null ) networkWriter.close();
            if( socket != null) socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /*
     * 서버와 연결을 맺기 위한 클래스
     */
    class SetSocket extends AsyncTask<Void, Void, Void> {
        @Override
        protected Void doInBackground(Void... params) {
            try {
                // 소켓 연결
                socket = new Socket(getString(R.string.server_ip),
                    Integer.parseInt(getString(R.string.server_port)));
```

```
                // 소켓으로부터 InputSteam과 OutputStream을 가져온다.
                // 한글이 깨지는것을 막기 위해 인코딩을 EUC-KR로 설정한다.
                networkWriter = new BufferedWriter(
                new OutputStreamWriter(socket.getOutputStream(), "EUC-KR"));
                networkReader = new BufferedReader(
                new InputStreamReader(socket.getInputStream(), "EUC-KR"));

            } catch (UnknownHostException e) {
                e.printStackTrace();
                Log.e("socket", e.getMessage());
            } catch (IOException e) {
                Log.e("socket", e.getMessage());
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(Void result){
            try {
                RecvThread rt = new RecvThread( socket );
                rt.start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    /*
     * 서버에 메시지를 보내기 위한 클래스
     */
    class SendMsg extends AsyncTask<Void, Void, Void> {
        @Override
        protected Void doInBackground(Void... params) {
            PrintWriter out = new PrintWriter(networkWriter, true);

            // 채팅방ID + 디바이스ID + 유저ID + 메시지 형태로 문자열을 구성한다.
            String return_msg = chatRoomId + deviceId
                + String.format("%04d", userId)
                + edMsg.getText().toString();

            out.println(return_msg);
            return null;
        }
    }

    /*
     * 서버로부터 메시지를 수신하고 UI 쓰레드에 메시지를 출력하기 위한 쓰레드
```

```
 */
class RecvThread extends Thread{
    BufferedReader brSocketInput; // 서버로부터 데이터를 수신받기 위한 스트림
    String line = null;      // 서버로부터 수신 받은 데이터를 저장하기 위한 변수
    String chatRoomId;       // 수신된 메시지의 채팅방 ID
    String senderDeviceId;   // 수신된 메시지의 디바이스의 ID
    String senderUID;        // 수신된 메시지의 유저 ID
    String message;          // 실제 메시지 내용
    StringBuilder sBuilder = new StringBuilder();

    public RecvThread(Socket s ) throws IOException{
        // 서버로부터 데이터를 수신 받기 위한 스트림 생성
        brSocketInput = new BufferedReader(
            new InputStreamReader( s.getInputStream(), "EUC-KR" ) );
    }

    public void run() {
        // 입력 스트림을 통해 데이터를 읽어와서 출력한다.
        try {
            while ((line = brSocketInput.readLine()) != null) {
                // 불필요한 문자 제거
                if( line.charAt(2) > '9' || line.charAt(2) < '0' )
                    line = line.substring(3);
                chatRoomId = line.substring(0, 9);   // 채팅방 ID 파싱

                // 채팅방 ID가 다르면 수신하지 않는다.
                if ( !this.chatRoomId.equals(ChattingActivity.this.chatRoomId) )
                    continue;

                senderDeviceId = line.substring(9, 24); // 송신자의 디바이스 ID 파싱
                senderUID = line.substring(24, 28);   // 유저 ID 파싱
                message = line.substring(28);          // 수신받은 메시지 파싱

                // 핸들러를 통해서 메시지를 채팅화면(리스트뷰)에 추가한다.
                mHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        if (senderDeviceId.equals(deviceId)) {
                            alMsgDataList.add(
                                new MessageData(1, senderUID, message));
                        } else
                            alMsgDataList.add(new MessageData(0, senderUID,
                                message));

                            mlaChatListAdapter.notifyDataSetChanged();
                    }
                });
```

```
                    // sBuilder.delete(0, sBuilder.length());
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    // 메시지 데이터 : 메시지의 내용과 타입을 저장
    class MessageData {
        int type;
        String userName;
        String msg;

        public MessageData( int _type, String _uid, String _msg ) {
            type = _type;
            userName = "유저" + Integer.parseInt(_uid);
            msg = _msg;
        }
    }
}
```

onCreate() : ChattingRoomListActivity로부터 putExtra() 메서드로 전달한 값을 받기 위해 getIntent() 메서드로 인텐트를 가져온다. 인텐트로부터 과목명, 학수번호, 분반 값을 가져와 액션바 타이틀로 출력한다. 이 중 학수번호와 분반 두 개의 스트링을 합쳐서 하나의 채팅방을 고유하게 식별하는 chatRoomId를 생성한다. TelephonyManager를 통해 디바이스의 ID를 가져오며 0부터 9999까지 정수 중 임의의 수로 유저ID로 할당한다. 대화 내용을 출력하기 위한 리스트뷰와 관련된 객체들을 초기화한다. SetSocket 클래스로 서버와 통신하기 위한 소켓을 생성한다.

onDestroy() : 액티비티가 종료될 때 실행되는 메서드로 열려있는 입출력 스트림과 소켓을 닫는다.

SetSocket 클래스 : 서버와 통신하기 위한 소켓을 생성하고 입출력 스트림을 생성한다. 이 후 서버가 보내는 메시지를 받기 위한 RecvThread 쓰레드를 실행시킨다.

SendMsg 클래스 : 생성된 출력 스트림을 통해 서버로 메시지를 보낸다. 보내는 메시지의 형태는 '채팅방ID + 디바이스ID + 유저ID + 메시지'로 구성된다.

RecvThread 클래스 : 서버로부터 메시지를 수신하고 수신된 메시지를 화면에 출력하기 위한 쓰레드. 서버로부터 수신받은 메시지를 채팅방ID, 디바이스ID, 유저ID, 데이터(메시지)로 나누며, 채팅방ID가 현재 입장한 채팅방ID와 같지 않으면 메시지를 출력하지 않는다. 화면 출력은 메인 쓰레드에서 담당하기 때문에 메인 쓰레드의 핸들러 객체를 통해 수신받은 메시지를 화면에 출력한다. 화면에 출력할 때는 디바이스ID를 체크하여 만약 수신

된 메시지의 디바이스ID가 자신의 디바이스ID와 같다면 노란색 메시지박스를 오른쪽에 출력하며, 다른 사람이 보낸 메시지는 유저ID와 함께 흰색 메시지박스를 왼쪽에 출력한다.

MessageData 클래스 : 이 클래스는 수신받은 메시지를 화면(리스트뷰)에 출력하기 위해 쓰이는 클래스이다. type이 0이면 다른 사람의 메시지를 뜻하고 type이 1이면 자신이 보낸 메시지임을 뜻한다. 리스트뷰는 type에 따라 색과 형태가 다른 메시지박스를 출력한다. userName은 유저ID를 사용해 '유저 + ID' 형태로 저장하는 문자열 변수이고 msg는 받은 메시지를 저장한다.

[표 20] ChattingActivity.java

(2) XML 코드

main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/chat_list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:paddingTop="5dp"
        android:paddingBottom="5dp"
        android:background="#a2b7d2"
        android:transcriptMode="alwaysScroll"
        android:dividerHeight="5dp"
        android:divider="@android:color/transparent" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

            <EditText
                android:id="@+id/msg"
                android:layout_width="0dp"
                android:layout_weight="1"
                android:layout_height="wrap_content"
                android:maxLines="3"
                android:singleLine="true" />
```

```
            <Button
                android:id="@+id/send"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:enabled="false"
                android:text="전송" />

    </LinearLayout>

</LinearLayout>
```

채팅 화면을 구성하는 레이아웃이다. 리니어 레이아웃으로 되어 있으며 리스트뷰와 하위 리니어 레이아웃을 포함하고 있다. 리스트뷰는 사용자들과 주고받은 대화 목록이 출력되는 뷰이다. 그 밑의 리니어 레이아웃은 에디트텍스트와 버튼을 포함하고 있는데 각각 메시지 입력창과 전송버튼을 나타낸다.

[표 21] main.xml

message_box.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/user"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:textSize="17sp"
        android:text="유저1" />

        <TextView
            android:id="@+id/msg"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@id/user"
            android:layout_alignParentTop="true"
            android:background="#FFFFFF"
            android:maxWidth="260dp"
            android:padding="5dp"
            android:text="테스트 메세지"
            android:textSize="17sp" />

</RelativeLayout>
```

다른 사람이 보낸 메시지를 출력할 때 사용하는 메시지박스 레이아웃이다. 렐러티브 레

이아웃으로 구성되어 있으며 유저ID를 나타내는 텍스트뷰와 메시지를 나타내는 텍스트뷰 두가지로 이루어져있다. 흰색 배경의 텍스트뷰를 사용한다.

[표 22] message_box.xml

my_message_box.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">


        <TextView
            android:id="@+id/msg"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:background="#f6e626"
            android:maxWidth="260dp"
            android:padding="5dp"
            android:text="테스트 메세지"
            android:textSize="17sp" />

</RelativeLayout>
```

자신이 보낸 메시지를 출력할 때 사용하는 메시지박스이다. 렐러티브 레이아웃으로 구성하였으며 메시지만을 출력하면 되기 때문에 메시지를 출력하기 위한 텍스트뷰 하나로 이루어져있다. *android:layout_alignParentRight="true"* 속성을 사용해 리스트뷰에 추가되었을 때 오른쪽으로 정렬되어 출력되도록 하였다. 노란색 배경의 텍스트뷰를 사용한다.

[표 23] my_message_box.xml

2.4 프로토콜 및 메시지 포맷

데이터 및 메시지 송수신을 위해 두 가지 메시지 포맷을 사용한다.

1) 일반적인 메시지 포맷

- 사용자들끼리 주고받는 메시지를 위한 포맷.
- ***채팅방ID + 디바이스ID + 사용자ID + 메시지내용*** 으로 구성된다.
- ex) 채팅방ID가 007323028, 디바이스ID가 111122223333444, 사용자ID가 321, 보낼 메시지가 '안녕하세요' 라면,
  '00732302811112222333344440321안녕하세요' 형태로 메시지가 전송된다.
- 서버는 위와 같은 포맷의 메시지를 수신하면 해당 메시지를 연결된 모든 클라이언트에 브로드캐스트 한다.
- 클라이언트는 일반 메시지를 수신하면 subString 메서드를 사용해 각 ID와 메시지로 나누어서 처리한다.

2) 강의 목록 요청 메시지 포맷

- 클라이언트에서 서버에게 강의 목록을 요청할 때 사용.
- ***"REQUEST SUBJECT LIST"*** 문자열을 사용한다.
- 서버는 위와 같은 메시지를 수신하면 강의목록 객체(ArrayList<SubjectInfo>)를 클라이언트로 전송한다.
- 클라이언트는 강의목록 객체를 수신하여 리스트뷰에 적절히 출력한다.
- 겍체 입출력 스트림(Object Input/Output Stream)을 사용한다.

3. 컴파일 및 실행

서버 와 클라이언트 모두 이클립스를 사용하여 개발되었기 때문에 프로젝트 압축파일 역시 이클립스 프로젝트 형식으로 구성되어있다. 여기서는 각 프로젝트 압축파일을 이클립스로 불러와 컴파일하고 실행하는 방법을 설명한다.

3.1 서버 컴파일 및 실행

(1) 이클립스 실행 후 File - Import를 클릭한다.
(2) General - Existing Projects into Workspace를 선택한 후 Next를 클릭한다.
(3) Select archive file을 선택하고 Browse 버튼은 클릭한다.
(4) ChatServer(1.0).zip을 찾아 열기(O) 버튼을 클릭한다.
(5) Projects: 목록에 ChatServer가 체크되어 있는지 확인한다. 체크되어 있지 않다면 체크를 하고 Finish버튼을 클릭한다.
(6) 추가된 ChatServer에서 오른쪽 마우스를 클릭한다.
(7) Build Path 메뉴에서 Configure Build Path를 클릭한다.
(8) Libraries 탭을 클릭한다.
(9) 경로가 잘못된 jar파일을 모두 선택한 뒤 Remove버튼을 클릭해 제거한다.
(10) 오른쪽의 Add External JARs버튼을 클릭한다.
(11) workspace의 프로젝트 폴더의 lib폴더로 이동한다.
(12) lib폴더의 모든 jar파일을 선택한위 열기(O)버튼을 클릭한다.
(13) OK버튼을 클릭한다.
(14) ChatServer.java 파일의 88번째줄 main함수에서 서버의 포트번호를 설정해준다. (기본 9999).
(15) Ctrl + F11을 눌러 컴파일하고 실행한다.

3.2 안드로이드 클라이언트 컴파일 및 실행

(1) AndroidChattingClient(1.0).zip파일의 압축을 푼다.
(2) 이클립스 실행 후 File - Import를 클릭한다.
(3) Android의 Existing Android Code Into Workspace를 선택한 뒤 Next를 클릭한다.
(4) Browse 버튼을 클릭해 압축을 푼 폴더를 선택하고 확인 버튼을 클릭한다.
(5) Projcets에 AndroidChattingClient를 체크한다.
(6) 그 아래에 있는 Copy projects into workspace에 체크한다.
(7) Finish를 클릭한다.
(8) 추가된 프로젝트의 res - values - strings.xml 파일에서 server_ip 요소와 server_port 요소의 값을 실행하려는 서버 환경에 맞게 설정해준다.
(9) 안드로이드 클라이언트 프로젝트의 아무 java 파일을 연 후 Ctrl + F11을 눌러 컴파일하고 실행한다.(디바이스나 연결되어 있거나 에뮬레이터가 실행중이어야 함)

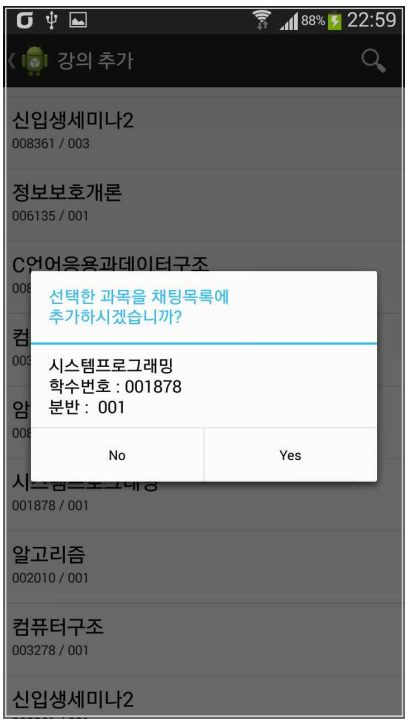
3.3 어플리케이션 사용법

[그림 2] 채팅방 목록이 없을 때

(1) 어플리케이션을 처음 실행하면 왼쪽과 같은 채팅방 리스트 화면이 출력된다. 아직 추가된 채팅방이 없기 때문에 '채팅 목록이 없습니다' 라는 문구가 보인다. 오른쪽 상단의 + 버튼을 누르면 채팅방을 생성할 수 있는 강의목록 화면이 띄워진다.

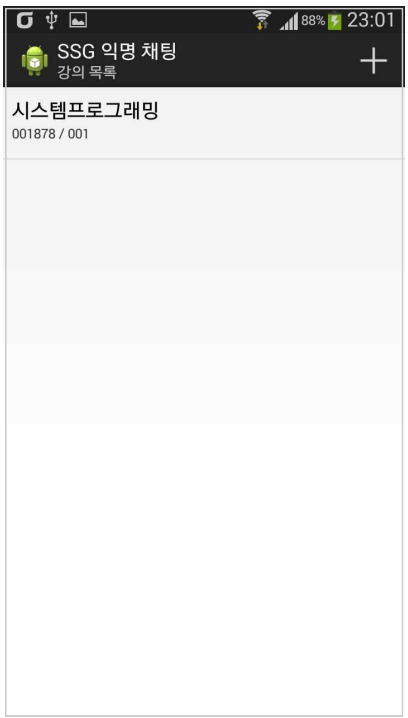
[그림 3] 강의 목록

(2) 서버로부터 강의목록을 받아 출력하는 화면이다. 스크롤하여 원하는 강의를 찾아 선택한다.



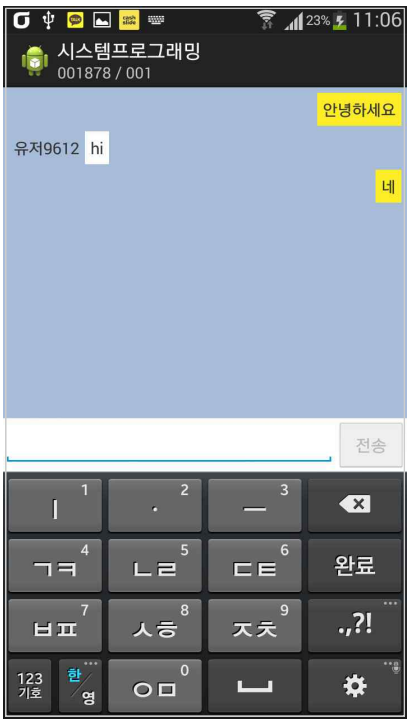
[그림 4] 강의 선택

(3) 원하는 강의를 선택하면 채팅목록에 추가할지 여부를 묻는 다이얼로그가 뜬다. Yes를 선택하면 선택한 강의가 채팅목록에 추가되고 채팅화면으로 이동한다. 예에서는 시스템프로그래밍 과목(001878/ 001)을 선택하였다.



[그림 5] 생성된 채팅방

(4) 채팅목록에 '시스템프로그래밍' 과목이 추가되었다. 해당 채팅방을 선택하면 채팅방으로 입장한다.



[그림 6] 채팅 화면

(5) 채팅 화면. 상단 액션바에 현재 입장한 채팅방의 과목명과 학수번호, 분반 정보를 보여준다. 가운데 부분은 다른 사용자와 주고 받은 메시지 목록을 출력한다. 자신이 보낸 메시지는 노란색 박스로, 다른 사용자들의 메시지는 유저ID와 함께 흰색 박스로 나타난다. 텍스트 입력창에 메시지를 입력하고 오른쪽 전송 버튼을 눌러 메시지를 전송한다.

(6) 위의 과정을 반복하여 원하는 강의의 채팅방을 추가로 생성한다. 생성된 각 채팅방에서 채팅을 진행한다.

4. 결과

결과적으로 처음에 목표했던 기능들 중 핵심기능들(강의목록 출력, 강의 선택하여 채팅방 생성, 각 채팅방 별 채팅)을 구현하였으나 전체적으로 부족한 점이 많고 아쉬움이 많이 남는 프로젝트였다. 그 원인으로는 먼저 학교 수업과 과제, 프로젝트 때문에 시간을 많이 할애하지 못한 점 때문이고, 두 번째는 팀원들 간의 스터디 및 역할 분담이 제대로 이뤄지지 않은 점 때문이라고 생각한다. 다음에도 이런 프로젝트를 진행하게 된다면 일단 계획부터 철저히 세우고 팀원들의 역할 분담을 명확히 한 뒤, 최대한 많은 시간을 투자하여 프로젝트를 진행해야할 것이다. 프로젝트의 결과 및 한계점들을 아래 표에 정리하였다.

항목	내용
낮은 완성도	소켓 및 파일입출력을 주로 사용하는 프로그램이기 때문에 예외가 많이 발생하고 이에 대한 예외처리가 굉장히 중요한데, 시간에 쫓겨 기능 구현에 급급한 나머지 이런 부분에 대한 예외처리를 제대로 구현하지 못했다.
비효율적	각 채팅방 별 메시지 전송 기능의 원래 목표는 클라이언트가 보낸 메시지를 서버가 각 채팅방별로 구분하여 보내는 방식이었는데, 실제 구현된 방법에서 서버는 클라이언트로부터 받은 메시지를 모든 클라이언트로 브로드캐스트 한다. 메시지 출력 여부는 클라이언트가 결정하기 때문에 클라이언트는 불필요한 메시지를 받아 처리해야하는 추가 작업을 해야 한다. 이는 단말기의 데이터 통신량을 늘리고 배터리가 더 빠르게 소모되는 원인이 될 수 있다.
편의성 부족	강의 검색 기능이 없어 원하는 강의를 찾으려면 약 2000개의 강의 목록을 손으로 스크롤해야 하는 불편함이 따른다. 이 외에도 이전에 대화 했던 내용을 저장한다던지, 어플리케이션이 실행중이지 않아도 메시지의 도착을 알려주는 알림기능이 없어 편의성이 부족하다.
서버 부재	채팅 서비스를 제공하려면 항상 켜져 있는 서버가 필요한데 프로그램을 실행할만한 서버를 구하지 못했다.
역기능 방지	1장에서 설명한 것처럼 어플리케이션이 의도한 바는 익명을 통한 자유로운 강의평가 및 교수평가 이지만, 반대로 익명이라는 점을 악용하여 각종 욕설, 비방 및 허위 사실을 유포할 수 있는 문제가 존재한다. 이런 역기능을 방지하기 위핸 해결책이 필요하다.

[표 24] 프로젝트 결과 및 한계점