군산대학교
KUNSAN NATIONAL UNIVERSITY

네트워크

모바일 시스템 SW

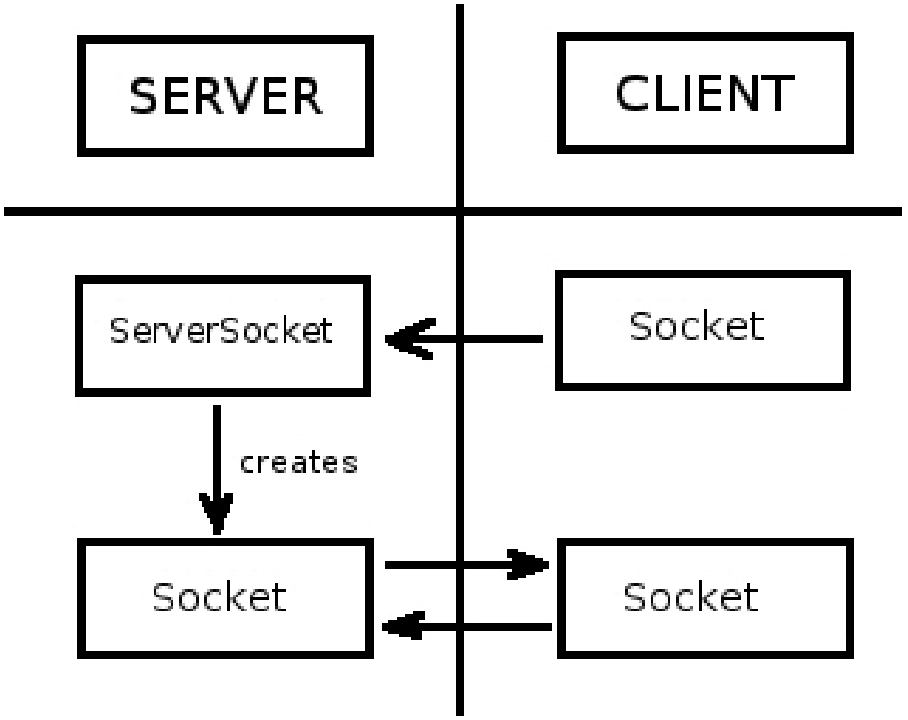
남 광 우

네트워크 소켓 연결

- **소켓**
  - ✓ 네트워크에서 정보가 들어가고 나가는 지점에 대한 유일한 식별자
  - ✓ 32비트 숫자로 구성되며 짝수번 소켓은 정보를 받아들이는 용도를, 홀수번 소켓은 정보를 보내는 용도를 나타냄
  - ✓ 자질구레한 부분은 운영 체제와 자바 네트워킹 API에서 처리해주므로 고수준 부분에만 신경을 쓰면 됨

```
Socket chatSocket = new Socket("196.164.1.103", 5000);
```

ServerSocket과 Socket



TCP 포트 번호

❖ TCP 포트 번호

- ✓ 서버에서 돌아가는 특정 프로그램을 나타내는 16비트 숫자
  - 웹 서버(HTTP) – 80번
  - 텔넷 서버 – 23번
  - FTP 서버 – 20번
  - POP3 – 110번
  - SMTP – 25번
  - 타임 서버 – 37번
  - HTTPS – 443번
- ✓ 물리적인 장치를 꽂는 장소는 아니고 용도에 따라 적절하게 쓰이는 숫자에 불과함

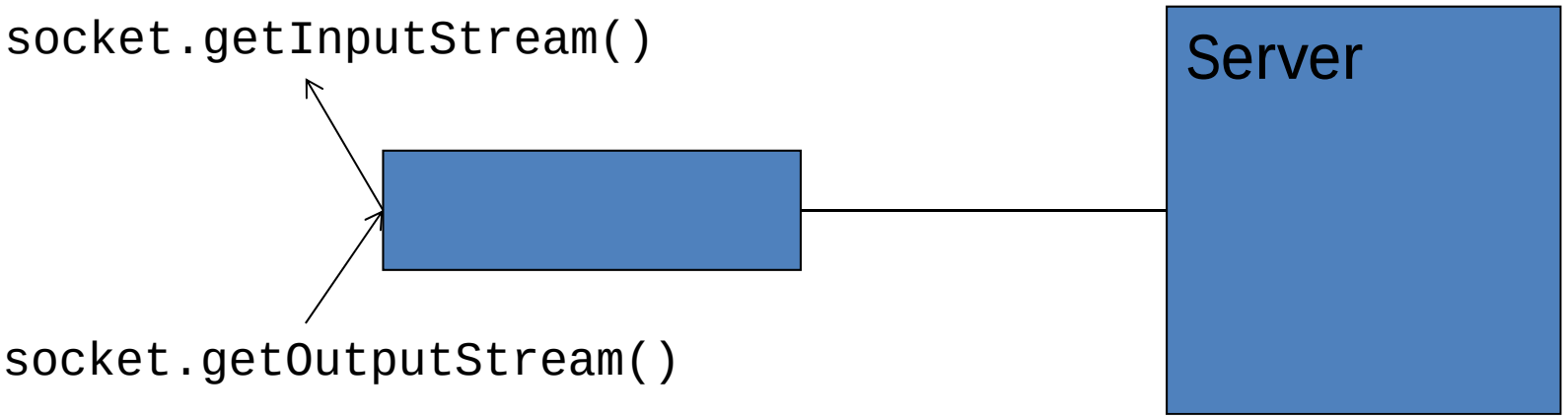
바보 같은 질문은 없습니다

- **연결하고 싶은 서버 프로그램의 포트 번호는 어떻게 알 수 있나요?**
  - ✓ 잘 알려진 서비스라면 인터넷에서 찾을 수 있습니다.
  - ✓ 그렇지 않은 경우에는 서비스를 제공하는 쪽에 직접 물어봐야 합니다.
- **같은 포트에서 여러 프로그램이 돌아갈 수도 있나요?**
  - ✓ 안 됩니다. 이미 사용중인 포트에 프로그램을 연결하려고 하면 BindException이라는 예외가 발생합니다.

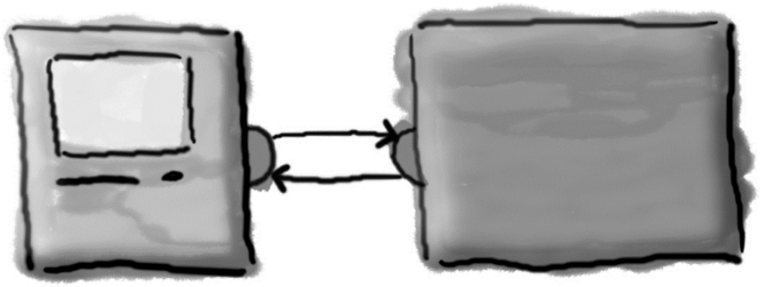
Socket 읽기과 쓰기

- 연결하고 데이터 읽기와 쓰기

```
Socket socket= new Socket("196.164.1.103", 5000);
```



Socket으로부터 데이터를 읽어오는 방법



- 자바에서는 대부분의 입출력 작업에서 고수준 연쇄 스트림이 실제로 어디에 연결되어 있는지에 대해 별로 신경을 쓰지 않아도 됩니다.

- 연결 스트림이 File이 아니라 Socket이지만 전과 마찬가지로 BufferedReader를 연쇄시켜서 사용하면 됩니다.

Socket으로부터 데이터를 읽어오는 방법

1. **서버에 Socket 연결**

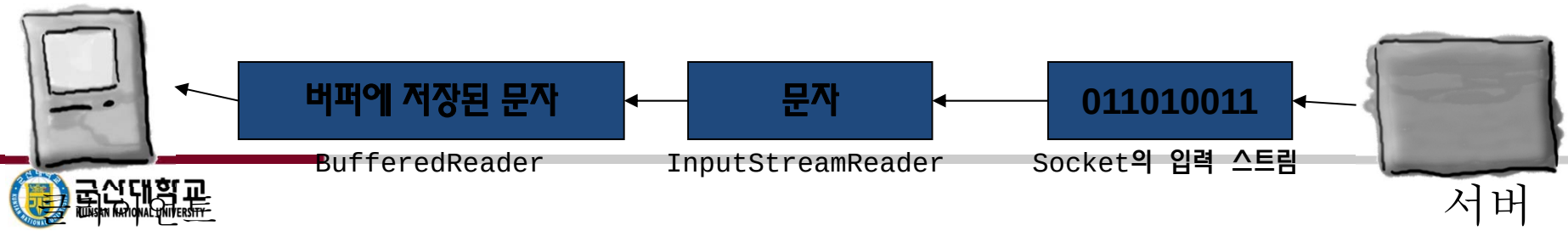
```
Socket chatSocket = new Socket("127.0.0.1", 500);
```

2. **Socket의 저수준 입력 스트림에 연쇄된 InputStreamReader 만들기**

```
InputStreamReader stream = new
    InputStreamReader(chatSocket.getInputStream());
```

3. **BufferedReader를 만들고 읽기**

```
BufferedReader reader = new BufferedReader(stream);
String message = reader.readLine();
```



Socket으로 데이터를 쓰는 방법

- **BufferedWriter vs. PrintWriter**
  - ✓ 소켓으로 데이터를 보낼 때는 파일에 데이터를 쓸 때와 마찬가지로 BufferedWriter를 쓸 수도 있지만 한 번에 하나씩의 String을 보낼 때는 PrintWriter를 쓰는 것이 가장 표준적인 방법

- **System.out에서와 마찬가지로 print()와 println() 메소드를 많이 사용합니다.**

Socket으로 데이터를 쓰는 방법

1. **서버에 Socket 연결**

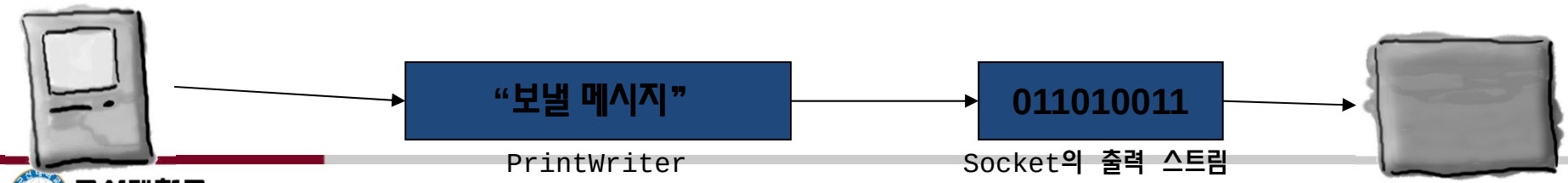
```
Socket chatSocket = new Socket("127.0.0.1", 500);
```

2. **Socket의 저수준 입력 스트림에 연쇄된 PrintWriter 만들기**

```
PrintWriter writer = new
    PrintWriter(chatSocket.getOutputStream());
```

3. **출력**

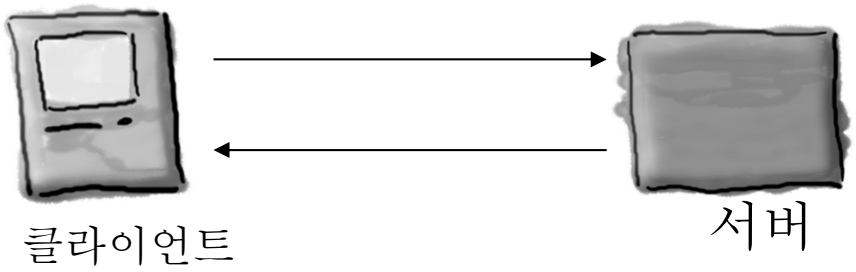
```
writer.print("보낼 메시지" );
writer.println("또 다른 메시지" );
```



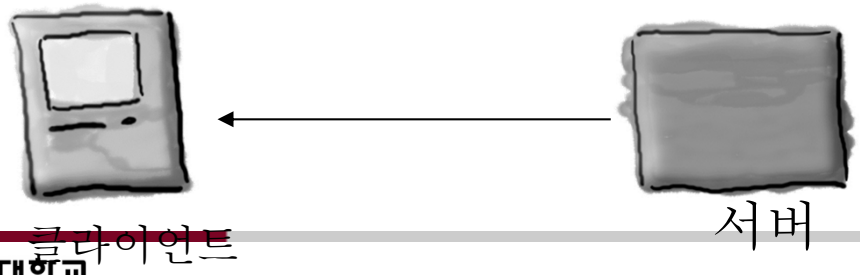
DailyAdviceClient

- **간단 예제**
  - 조언전문가 서버로부터 오늘의 조언 받아오기

1. 연결하기



2. 읽기

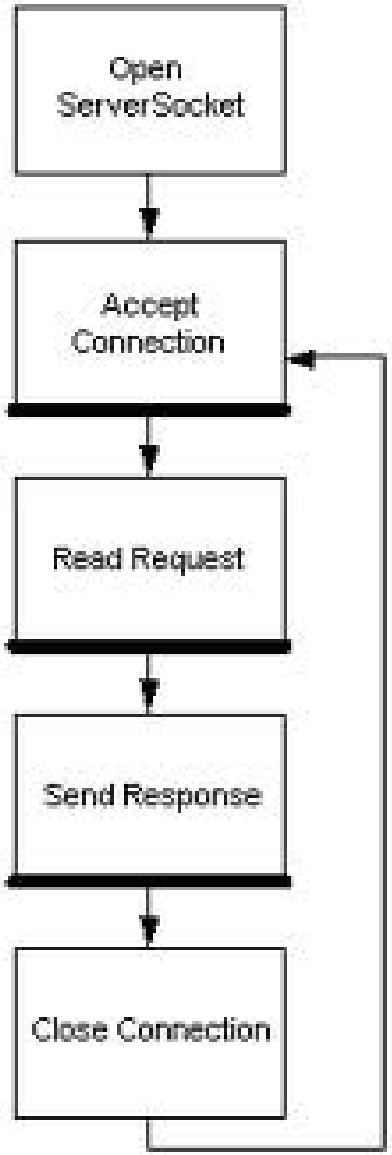


조언 전문가

DailyAdviceClient 코드

```
import java.io.*;
import java.net.*;


public class DailyAdviceClient {

    public void go() {
        try {
                Socket s = new Socket("127.0.0.1", 4200);
                InputStreamReader streamReader = new
                        InputStreamReader(s.getInputStream());
                BufferedReader reader = new BufferedReader(streamReader);
                String advice = reader.readLine();
                System.out.println("오늘의 조언: " + advice);
        } catch(IOException ex) {
                ex.printStackTrace();
        }
    } // go 메소드 끝

    public static void main(String[] args) {
        DailyAdviceClient client = new DailyAdviceClient();
        client.go();
    }
```

ServerSocket



DailyAdviceServer

1. **서버 애플리케이션에서 특정 포트에 대한 ServerSocket을 만듭니다.**

```
ServerSocket serverSock = new ServerSocket(4242);
```

2. **클라이언트에서 서버 애플리케이션으로 소켓 연결을 합니다.**

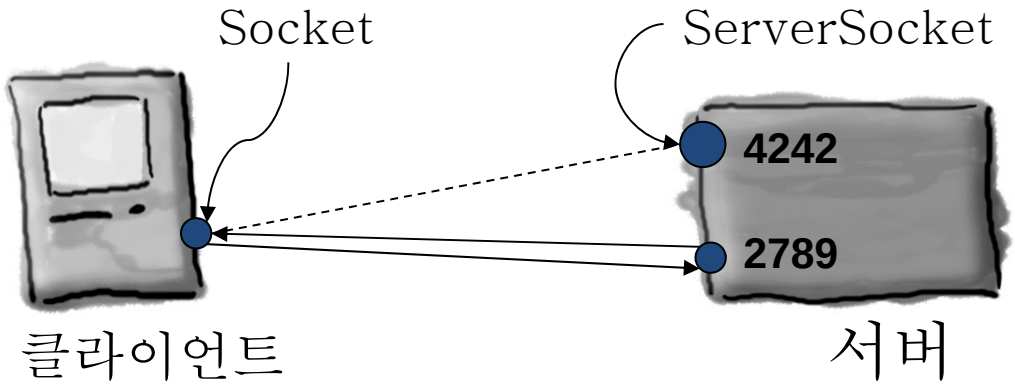
```
Socket sock = new Socket("190.165.1.103", 4242);
```

3. **서버에서 클라이언트와 통신하기 위한 새로운 소켓을 만듭니다.**

```
Socket sock = serverSocket.accept();
```

DailyAdviceServer : 간단한 서버 만들기

```
ServerSocket serverSock = new ServerSocket(4242);
Socket sock = new Socket("190.165.1.103", 4242);
```



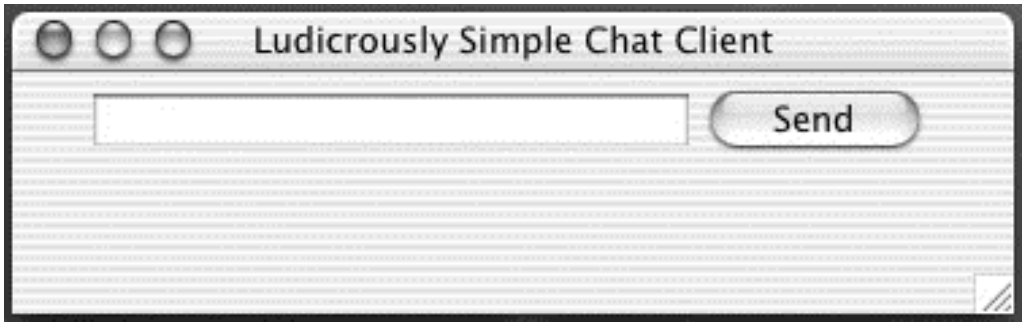

```
Socket sock = serverSocket.accept();
```

DailyAdviceServer 코드

❖ 소스 코드

```
public class DailyAdviceServer{

   String [] adviceList = { " 조금씩 드세요", " 운동하세요“, " 금연하세요” };

   public void go(){

      try{

         ServerSocket servSock = new ServerSocket(4242);

         while(true){

            Socket sock = servSock.accept();

            PrintWriter writer = new PrintWriter(sock.getOutputStream());
            String advice = getAdvice();
            writer.println(advice);
            writer.close();
            System.out.println(advice);


         }//fim do while


      }
      catch(IOException ex){

         ex.printStackTrace();

      }
                                          }//fim do método go

                                          private String getAdvice(){

                                            int random = (int) (Math.random() * adviceList.length);
                                            return adviceList[random];


                                          }//fim do método getAdvice

                                          public static void main(String[] args)
                                          {
                                             DailyAdviceServer server = new DailyAdviceServer();
                                             server.go();
                                          }//fim do main
                                       }//fim da classe DailyAdviceServer
```

채팅 클라이언트(1)



❖ 보내기 전용 채팅 클라이언트를 만들어봅시다.

코드를 직접 입력해서 테스트해봅시다.

채팅 클라이언트(2)

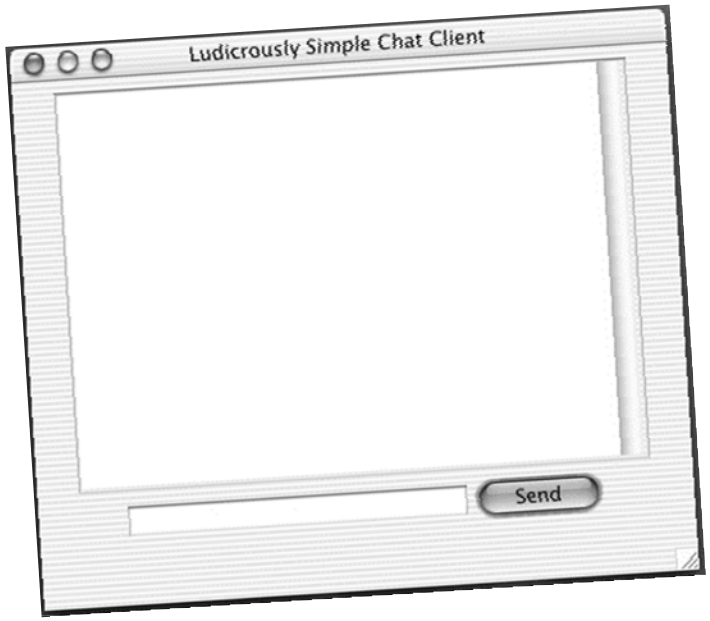
❖ 언제 서버로부터 메시지를 받을까요?

1. 20초마다 서버 확인

2. 사용자가 메시지를 보낼 때마다 메시지 확인

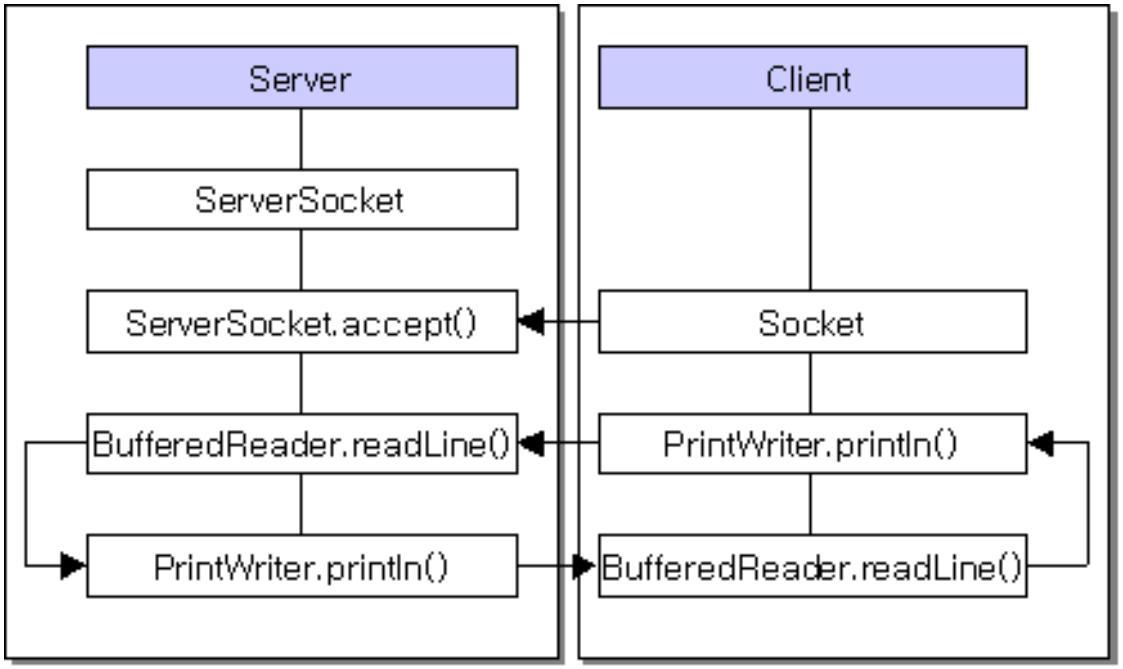
3. 서버에서 메시지를 보내면 바로 읽기

채팅 클라이언트(2)



- 보내기와 받기가 가능한 버전
- 서버로부터 메시지를 받는 방법은?
  - ✓ 네트워크 설정시에 입력 스트림(BufferedReader)을 같이 만들면 됨.
  - ✓ 메시지를 읽을 때는 readLine() 메소드 사용.

Simple Chat


Server
Client
ServerSocket
ServerSocket.accept()
Socket
BufferedReader.readLine()
PrintWriter.println()
PrintWriter.println()
BufferedReader.readLine()

SimpleChatClient.java

```
public class SimpleChatClient {
    JTextArea incoming;
    JTextField outgoing;
    BufferedReader reader;
    PrintWriter writer;
    Socket sock;
    public static void main(String[] args) {
       SimpleChatClient client = new SimpleChatClient();
       client.go();
    }
    public void go() {
        JFrame frame = new JFrame("Ludicrously Simple Chat Client");
        JPanel mainPanel = new JPanel();
        incoming = new JTextArea(15,50);
        incoming.setLineWrap(true);
        incoming.setWrapStyleWord(true);
        incoming.setEditable(false);
        JScrollPane qScroller = new JScrollPane(incoming);
        qScroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
        qScroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        outgoing = new JTextField(20);
        JButton sendButton = new JButton("Send");
        sendButton.addActionListener(new SendButtonListener());
        mainPanel.add(qScroller);
        mainPanel.add(outgoing);
        mainPanel.add(sendButton);
        setUpNetworking();
        Thread readerThread = new Thread(new IncomingReader());
        readerThread.start();

        frame.getContentPane().add(BorderLayout.CENTER, mainPanel);
        frame.setSize(800,500);
        frame.setVisible(true);
     }
```

SimpleChatClient.java

```
private void setUpNetworking(){
  try{
   sock = new Socket("127.0.0.1",5000);
   InputStreamReader streamReader = new InputStreamReader(sock.getInputStream());
   reader = new BufferedReader(streamReader);
   writer = new PrintWriter(sock.getOutputStream());
   System.out.println("networking established");
  }catch(IOException ex){
   ex.printStackTrace();
  }
 }

 public class SendButtonListener implements ActionListener{
  public void actionPerformed(ActionEvent ev){
   try{
   writer.println(outgoing.getText());
   writer.flush();
   }catch(Exception ex){
    ex.printStackTrace();
   }
   outgoing.setText("");
   outgoing.requestFocus();
  }
 }
```

```
public class IncomingReader implements Runnable{
  public void run(){
   String message;
   try{
    while((message = reader.readLine())!= null){
     System.out.println("read "+ message);
     incoming.append(message+ "\n");
     }
    }catch(Exception ex){
     ex.printStackTrace();
    }
   }
  }
 }
```

VerySimpleChatServer.java

```
public class VerySimpleChatServer {
  ArrayList clientOutputStreams;

  public class ClientHandler implements Runnable{
   BufferedReader reader;
   Socket sock;

   public ClientHandler(Socket clientSocket){
    try{
     sock = clientSocket;
     InputStreamReader isReader = new InputStreamReader(sock.getInputStream());
     reader = new BufferedReader(isReader);

    }catch(Exception ex){
     ex.printStackTrace();
    }

   }

   public void run(){
    String message;
    try{
     while((message = reader.readLine()) != null){
      System.out.println("read "+ message);
      tellEveryone(message);
     }
    }catch(Exception ex){
     ex.printStackTrace();
    }

   }

  }
```

VerySimpleChatServer.java

```
public static void main(String[] args){
  new VerySimpleChatServer().go();
 }

 public void go(){
  clientOutputStreams = new ArrayList();
  try{
   ServerSocket serverSock = new ServerSocket(5000);
   while(true){
    Socket clientSocket = serverSock.accept();
    PrintWriter writer = new PrintWriter(clientSocket.getOutputStream());
    clientOutputStreams.add(writer);

    Thread t = new Thread(new ClientHandler(clientSocket));
    t.start();
    System.out.println("got a connection");
   }

  }catch(Exception ex){
   ex.printStackTrace();
  }

 }
```

```
 public void tellEveryone(String message){
  Iterator it = clientOutputStreams.iterator();
  while(it.hasNext()){
   try{
    PrintWriter writer = (PrintWriter)it.next();
    writer.println(message);
    writer.flush();
   }catch(Exception ex){
    ex.printStackTrace();
   }
  }
 }

}
```

andorid socket client

```
public class NewClient extends Activity {
    private String html = "";
    private Handler mHandler;

    private Socket socket;
    private String name;
    private BufferedReader networkReader;
    private BufferedWriter networkWriter;
    private String ip = "xxx.xxx.xxx.xxx"; // IP
    private int port = 9999; // PORT번호

    @Override
    protected void onStop() {
        // TODO Auto-generated method stub
        super.onStop();
        try {
            socket.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

http://pulsebeat.tistory.com/24

andorid socket client

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mHandler = new Handler();

    try {
       setSocket(ip, port);
    } catch (IOException e1) {
       // TODO Auto-generated catch block
       e1.printStackTrace();
    }

    checkUpdate.start();

    final EditText et = (EditText) findViewById(R.id.EditText01);
    Button btn = (Button) findViewById(R.id.Button01);
    final TextView tv = (TextView) findViewById(R.id.TextView01);

    btn.setOnClickListener(new OnClickListener() {
       public void onClick(View v) {
          if (et.getText().toString() != null
                || !et.getText().toString().equals("")) {

             PrintWriter out = new PrintWriter(networkWriter,true);
             String return_msg = et.getText().toString();
             out.println(return_msg);
          }
       }
    });
 }
```

andorid socket client

```
    public void setSocket(String ip, int port) throws IOException {
        try {
            socket = new Socket(ip, port);
            networkWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            networkReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

        } catch (IOException e) {
            System.out.println(e);
            e.printStackTrace();
        }
    }
}
```

andorid socket client

```
private Thread checkUpdate = new Thread() {
    public void run() {
        try {
            String line;
            Log.w("ChattingStart", "Start Thread");
            while (true) {

                Log.w("Chatting is running", "chatting is running");
                line = networkReader.readLine();
                html = line;
                mHandler.post(showUpdate);
            }

        } catch (Exception e) {
        }
    }
};

private Runnable showUpdate = new Runnable() {
    public void run() {
        Toast.makeText(NewClient.this, "Coming word: " + html,
                Toast.LENGTH_SHORT).show();
    }
};
```

andorid socket client

androidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="socket.client"

  android:versionCode="1"

  android:versionName="1.0">

   <application android:icon="@drawable/icon"android:label="@string/app_name">

      <activity android:name=".NewClient"

        android:label="@string/app_name">

    <intent-filter>

      <action android:name="android.intent.action.MAIN" />

      <category android:name="android.intent.category.LAUNCHER" />

    </intent-filter>

  </activity>

  </application>

   <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```