PENGEMBANGAN APLIKASI VOIP MENGGUNAKAN *REAL-TIME TRANSPORT PROTOCOL*

SKRIPSI

Diajukan kepada Fakultas Sains dan Teknologi
Universitas Islam Negeri Sunan Kalijaga
Untuk Memenuhi Sebagian Syarat Memperoleh Gelar Sarjana
Strata Satu Teknik Informatika

Disusun oleh:
Mark Prima Estafeta Muchammad
NIM. 05650018

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
YOGYAKARTA
2011

PENGEMBANGAN APLIKASI VOIP MENGGUNAKAN *REAL-TIME TRANSPORT PROTOCOL*

SKRIPSI

Diajukan kepada Fakultas Sains dan Teknologi
Universitas Islam Negeri Sunan Kalijaga
Untuk Memenuhi Sebagian Syarat Memperoleh Gelar Sarjana
Strata Satu Teknik Informatika

Disusun oleh:
Mark Prima Estafeta Muchammad
NIM. 05650018

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
YOGYAKARTA
2011

Universitas Islam Negeri Sunan Kalijaga FM-UINSK-BM-05-07/R0

PENGESAHAN SKRIPSI/TUGAS AKHIR

Nomor : UIN.02/D.ST/PP.01.1/697/2011

Skripsi/Tugas Akhir dengan judul : Pengembangan Aplikasi VoIP Menggunakan Real-time Transport Protocol

Yang dipersiapkan dan disusun oleh :
Nama : Mark Prima Estafeta Muchammad
NIM : 05650018
Telah dimunaqasyahkan pada : 5 April 2011
Nilai Munaqasyah : A -

Dan dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga

TIM MUNAQASYAH :

Ketua Sidang

Imam Riyadi, M.Kom
NIP. 60020397

Penguji I

Sumarsono, M. Kom
NIP.19710209 200501 1 003

Penguji II

M. Mustaqim, MT
NIP. 19790331 200501 1 004

Yogyakarta, 12 April 2011
UIN Sunan Kalijaga
Fakultas Sains dan Teknologi
Dekan

Prof. Drs. H. Akh. Minhaji, M.A, Ph.D
NIP. 19580919 198603 1 002

Universitas Islam Negeri Sunan Kalijaga FM-UINSK-BM-05-03/R0

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi/Tugas Akhir
Lamp :

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama : Mark Prima Estafeta Muchammad
NIM : 05650018
Judul Skripsi : **Pengembangan Aplikasi VoIP Menggunakan *Real-time Transport Protocol***

sudah dapat diajukan kembali kepada Program Studi Teknik Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Teknik Informatika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqsyahkan. Atas perhatiannya kami ucapkan terima kasih.

Yogyakarta, 09/03/2011

Pembimbing I

Imam Riadi, M.Kom

Pembimbing II

Agus Mulyanto, M.Kom
NIP. 19710823 199903 1 003

PERNYATAAN KEASLIAN SKRIPSI

Yang bertanda tangan di bawah ini:

Nama : Mark Prima Estafeta Muchammad

NIM : 05650018

Program Studi : Teknik Informatika

Fakultas : Sains dan Teknologi

Menyatakan bahwa skripsi dengan judul "**PENGEMBANGAN APLIKASI VOIP MENGGUNAKAN *REAL-TIME TRANSPORT PROTOCOL***" tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 16 Maret 2011

Yang menyatakan,

Mark Prima Estafeta Muchammad

NIM. 05650018

KATA PENGANTAR

Segala puji bagi Allah *rabbul'alamin* yang telah memberikan pertolongan dan ilmu-Nya kepada penulis sehingga dapat menyelesaikan penelitian yang berjudul *Pengembangan Aplikasi VoIP Menggunakan* Real-time Transport Protocol. Dalam penelitian ini, penulis merancang dan mengimplementasikan sistem komunikasi suara berbasis jaringan internet menggunakan RTP yang kemudian dipasang pada sebuah aplikasi *instant messenger*.

Selanjutnya penulis mengucapkan terima kasih kepada:

1. Bapak Prof. Drs. H. Akh. Minhaji, M.A, Ph.D, selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga;
2. Bapak Imam Riadi, M.Kom, selaku Dosen Pembimbing I dan yang dengan kesabaran dan ketelitiannya memberikan koreksi dan masukan terhadap penulisan skripsi ini;
3. Bapak Agus Mulyanto, M.Kom, selaku Dosen Pembimbing II dan Ketua Program Studi yang dengan kesabarannya telah membimbing selama penyusunan skripsi ini;
4. Para Dosen Program Studi Teknik Informatika yang telah memberi bekal ilmu pengetahuan kepada penulis, semoga ilmunya menjadi amal *jariyah*;
5. Mantan laboran Teknik Informatika UIN Sunan Kalijaga, Mas Yusuf Murdani dan para laboran Teknik Informatika UIN Sunan Kalijaga yang telah meminjamkan tempat dan fasilitas selama penelitian penulis;

6. Kedua orang tua dan kedua mertua penulis yang selalu memberikan dukungan dan dorongan semangat selama penyusunan skripsi ini;
7. Istriku Ulya Lutfiana yang senantiasa memberikan dukungan dan masukan-masukan selama penelitian dan penyusunan skripsi ini;
8. Teman-temanku Rafi, Syamsul, Rahmadhan, Iqbal, Putri, Ardian, Fathan, Sunu, Sigit, Rifqi, Mubarok dan teman-teman Program Studi Teknik Informatika lainnya yang telah memberikan bantuan dan dukungan selama penelitian dan penyusunan skripsi ini.

Penulis menyadari masih banyak kekurangan dan kelemahan dalam penelitian ini. Oleh karena itu demi perkembangan penelitian selanjutnya penulis sangat mengaharap kritik dan saran dari pembaca. Akhirnya semoga penelitian ini bermanfaat bagi pembaca dan dapat bermanfaat.

Yogyakarta, Maret 2011

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

INTISARI

Kemajuan teknologi informasi dan komunikasi telah banyak memudahkan manusia dalam berbagai hal terutama dalam berkomunikasi. Keberadaan jaringan internet telah mempermudah manusia untuk dapat berkomunikasi dengan hampir semua orang dengan mudah dan murah. Pemanfaatan internet terkadang kurang efisien karena adanya komunikasi yang sebenarnya bersifat intranet.

Penelitian ini mengembangkan sebuah sistem komunikasi suara antarkomputer melalui protokol internet dan memasangnya pada sebuah *instant messenger* untuk jaringan lokal. Sistem komunikasi yang dikembangkan memanfaatkan teknologi *audio streaming* dengan menggunakan *Real-time Transport Protocol* (RTP) sebagai protokol komunikasi *real-time*. Sistem ini akan dipasang pada sebuah aplikasi *instant messenger*, yaitu Mokal. Sistem ini dibangun dengan menggunakan Java dan JMF API (*Java Media Framework Application Programming Interface*). Metode penelitian yang dipakai adalah pengembangan sistem dengan model linear sekuensial yang meliputi analisis terhadap *instant messenger* yang akan dikembangkan, perancangan sistem komunikasi suara dan integrasinya pada *instant messenger* tersebut, implementasi sistem komunikasi suara, serta pengujian yaitu *alpha* dan *beta*.

Berdasarkan pengujian, aplikasi VoIP menggunakan *real-time transport protocol* ini dapat berjalan dengan baik, tetapi ada dalam kualitas suara, yaitu ketika kualitas suara baik gemanya tinggi sedangkan jika gema diperkecil kualitas suaranya menurun. Komunikasi suara yang didukung oleh aplikasi ini adalah komunikasi secara berpasangan maupun berkolompok.

Kata kunci: VoIP, *audio streaming*, RTP, Java, JMF API.

ABSTRACT

The advancement of information and communication technology has given us so much ease in many things especially in communication. Internet has enabled us to communicate to almost everyone, cheaply and easily. Unfortunately, internet is sometimes used inefficiently. That is, by using it to communicate to people within the intranet that is possible via direct access.

This research was designed to develop an inter-computer voice communication system over Internet protocol and install it to an existing local instant messenger. The communication system developed here makes use of audio streaming technology using Real-time Transport Protocol (RTP) as its real-time communication protocol. This system was to be installed in an instant messenger called Mokal as an additional module. The system was written in Java programming language using JMF API (Java Media Framework Application Programming Interface) as the media processor. The research method used in this research was system development using the linear sequential model consists of the analysis of the instant messenger to be extended, the design of the voice communication system and its integration to the instant messenger, implementation of the voice communication system and alpha and beta testing.

The testing phase shows that this real-time transport protocol-based VoIP system can function properly but still has a problem with the voice quality. That is when the voice quality is good, the echo effect is high and when the echo effect is low, the voice quality also drops. The kinds of voice communication supported by this system are one-to-one voice communication as well as conference voice communication.

Keywords: VoIP, audio streaming, RTP, Java, JMF API

BAB I

PENDAHULUAN

1.1 Latar Belakang

Manusia sebagai makhluk sosial membutuhkan komunikasi dengan sesamanya. Masalah muncul ketika manusia ingin berkomunikasi secara langsung dengan orang yang berada di tempat yang berbeda. Manusia kemudian berusaha mengembangkan teknologi untuk dapat berkomunikasi dengan orang lain yang berada di tempat yang berbeda. Salah satu teknologi komunikasi yang banyak dipakai orang saat ini adalah internet. Internet mendukung sistem komunikasi berbasis teks, audio maupun visual. Internet memungkinkan penggunanya untuk dapat berkomunikasi satu maupun dua arah, bahkan lebih, dan juga secara langsung maupun tidak langsung.

Koneksi internet yang ada sering kali tidak digunakan seefisien mungkin. Penyebabnya adalah pengguna aplikasi-aplikasi komunikasi berbasis internet menggunakannya untuk berkomunikasi dengan orang lain yang berdekatan dalam satu jaringan lokal dengannya. Penggunaan seperti itu hanya akan memboroskan *bandwidth* koneksi internet karena seharusnya tanpa perlu terhubung ke server di luar LAN komunikasi tetap dapat dilakukan.

Berdasarkan permasalahan di atas, penulis berusaha mengembangkan aplikasi VoIP untuk LAN. Aplikasi VoIP untuk LAN diharapkan akan dapat mengefisienkan penggunaan koneksi internet. Aplikasi sejenis telah banyak dikembangkan sebelumnya, diantaranya TeamSpeak (teamspeak.com), Ventrilo

(ventrilo.com), D-Voicer (d-voicer.com), SJPhone (sjlabs.com) dan LAN Voice Chat (softlakecity.com). TeamSpeak dan Ventrilo menggunakan arsitektur *client-server*, sedangkan sisanya menggunakan arsitektur *peer-to-peer*, maksudnya komunikasi antar *peer* dilakukan langsung tanpa ada *server*. Pada D-Voicer, alamat tujuan menggunakan *ip address* dan harus dimasukkan secara manual. Pada LAN Voice Chat, penentuan alamat tujuan lebih mudah karena aplikasi ini dapat menyimpan *ip address* dalam *phonebook*. LAN Voice Chat memiliki tombol *push-to-talk* yang harus ditekan ketika akan berbicara. SJPhone sebenarnya sangat mudah digunakan, aplikasi ini dapat mendeteksi siapa saja yang online di jaringan lokal. Komunikasi suara dilakukan secara langsung tanpa harus menekan tombol *push-to-talk*, namun aplikasi ini hanya mendukung komunikasi suara saja.

Penelitian ini dirancang untuk melengkapi aplikasi serupa yang sudah ada. Penelitian ini memanfaatkan aplikasi yang sudah ada dengan menambahkan modul untuk komunikasi suara. Aplikasi yang akan dikembangkan adalah sebuah *local instant messenger* bernama Mokal. Penelitian ini diharapkan akan menghasilkan sebuah aplikasi perangkat lunak yang dapat mengakomodasi komunikasi suara maupun teks tanpa memerlukan *server*. Aplikasi tersebut diharapkan dapat dijalankan pada semua jenis sistem operasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana mempertukarkan data suara antarkomputer secara *real-time*;
2. Bagaimana mengimplementasikan pertukaran data suara tersebut menjadi sebuah modul aplikasi untuk aplikasi Mokal.

1.3 Batasan Masalah

Masalah dalam penelitian ini dibatasi pada hal:

1. Sistem komunikasi yang dikembangkan dalam penelitian ini adalah komunikasi suara dua arah antarkomputer;
2. *Codec* yang digunakan adalah G.711 μ-law;
3. Pembuatan modul aplikasi komunikasi suara untuk dipasang pada aplikasi Mokal;
4. Teknologi yang digunakan dalam penelitian ini adalah JMF 2.1.1.
5. Penelitian ini menitikberatkan pada pengembangan sistem komunikasi suara, hal-hal seperti kualitas suara, pemakaian CPU dan efisiensi *bandwidth* tidak terlalu diperhatikan.

1.4 Tujuan dan Manfaat Penelitian

Penelitian ini bertujuan untuk mengembangkan sebuah modul komunikasi suara yang dapat dipasang pada *local instant messenger* bernama Mokal dan dapat digunakan pada jaringan lokal tanpa harus ada koneksi internet. Hasil penelitian

ini diharapkan dapat menyediakan sarana komunikasi murah dengan memanfaatkan infrastruktur jaringan lokal.

1.5 Keaslian Penelitian

Penelitian yang berhubungan dengan aplikasi VoIP untuk jaringan lokal sudah ada. Penelitian ini merupakan pengembangan penelitian-penelitian sebelumnya. Pada penelitian ini dikembangkan sebuah aplikasi VoIP antarkomputer untuk jaringan lokal yang mudah digunakan dengan RTP sebagai protokol komunikasinya. Fitur yang dimiliki adalah pengiriman pesan teks dan suara, selain itu aplikasi ini bekerja tanpa membutuhkan server.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan kegiatan yang telah dilakukan oleh penulis selama perancangan sampai implementasi aplikasi VoIP ini, maka dapat diambil beberapa kesimpulan berikut:

1. Aplikasi Mokal diberi tambahan fitur komunikasi suara yang memanfaatkan protokol RTP dalam pengiriman data suaranya yang merupakan sebuah protokol standar pengiriman data media dari salah satu kelompok standar untuk VoIP yaitu H.323;
2. Implementasi pertukaran data suara sebagai sebuah modul pada aplikasi Mokal yang dikembangkan dalam penelitian ini dilakukan dengan membuat *class* baru yang berfungsi untuk mempertukarkan data suara dan menambahkan tombol pada antarmuka yang sudah ada pada aplikasi tersebut untuk mengontrolnya;
3. Berdasarkan pengujian, dapat disimpulkan bahwa sistem yang dikembangkan telah berjalan dengan baik, namun masih ada kendala dalam kualitas suara, yaitu ketika kualitas suara baik gemanya tinggi sedangkan jika gema diperkecil kualitas suaranya menurun.

5.2 Saran

Aplikasi VoIP ini tidak lepas dari kekurangan dan kelemahan, terutama kualitas suara yang disebabkan keterbatasan yang dimiliki penulis. Oleh karena itu, untuk pengembangan aplikasi VoIP yang lebih baik, penulis menyarankan beberapa hal, yaitu:

1. Pemilihan *codec* untuk pengembangan berikutnya diharapkan agar lebih mempertimbangkan kualitas suara, penggunaan CPU dan juga penggunaan *bandwidth*;
2. Penanganan atau reduksi efek gema sebagai akibat dari *feedback* suara;
3. Penambahan fitur-fitur lain seperti komunikasi video, pertukaran *file*, *phonebook* dan sebagainya.

DAFTAR PUSTAKA

A. Hamid, Isredza Rahmi, et. al. 2009. *Development of Call Center of a SIP Based Voice over Internet Protocol (VOIP)*. Proceeding International Industrial Informatics Seminar 2009 Book 1 Vol 1, No. 1, Yogyakarta.

Abdel-Wahab, Hussein M. 2004. *JMF Lectures*. http://www.cs.odu.edu/~cs778/jmflects/index.html, diakses tanggal 19 Januari 2011.

Azikin, Askari dan Purwanto, Yudha. 2005. *Video/TV Streaming dengan Video LAN Project*. Penerbit Andi, Yogyakarta.

Cisco Systems, Inc. 2010. http://www.cisco.com/en/US/tech/tk1077/technologies_tech_note09186a00801149b3.shtml, diakses tanggal 3 Januari 2011.

Hall, Brian. 2009. *Beej's Guide to Network Programming*, http://www.beej.us/guide/bgnet/output/html/singlepage/bgnet.html, diakses tanggal 24 Agustus 2008.

Handoyo dan Wicaksono, D. 2007. *Desain dan Implementasi Layanan VOIP melalui Jaringan LAN di Kampus ITS*. Institut Teknologi Sepuluh November, Surabaya, http://digilib.its.ac.id/bookmark/4279/LAN, diakses tanggal 11 Februari 2010.

Kirom, Mukhammad Ramdlan. 2004. *Perancangan Aplikasi Distance Learning Real Time Menggunakan Socket Programming dengan Protokol VoIP*. Sekolah Pascasarjana Institut Teknologi Bandung, Bandung, http://www.ittelkom.ac.id/library/index.php?option=com_repository&Itemid=34&task=detail&nim=111020259, diakses tanggal 11 Februari 2010.

Nariswara, Adi. 2009. *Analisis Performansi VoIP IPv4 dan IPv6 pada Jaringan Broadband WiMAX*. Institut Teknologi Telkom, Bandung, http://www.ittelkom.ac.id/library/index.php?option=com_content&view=article&id=632:teknologi-jaringan-voip&catid=10:jaringan&Itemid=15, diakses tanggal 21 Januari 2011.

Odom, Wendell. 2004. *CCNA Self-Study CCNA INTRO Exam Certification Guide*. Cisco Press, Indianapolis.

Prasetyo, Budi. 2006. *Analisis Implementasi Voice Over Internet Protocol (VOIP) Pada Jaringan Wireless Lan Berbasis Session Initiation Protocol (SIP)*.

Institut Teknologi Telkom, Bandung, http://www.ittelkom.ac.id/library/index.php?option=com_repository&ItemiI=34&task=detail&nim=111010009, diakses tanggal 11 Februari 2010.

Proboyekti, Umi. 2008. *Flowchart*. http://lecturer.ukdw.ac.id/othie/flowchart.pdf, diakses tanggal 3 Januari 2011.

Pressman, Roger S. 2005. *Software Engineering Sixth Edition*. Mc Graw Hill, New York.

Putra, Priguna Sidharta. 2008. *Perancangan dan Implementasi Jaringan VoIP Menggunakan Protokol H.323 di Pemda Ogan Ilir Sumatera Selatan*. Institut Teknologi Telkom, Bandung, http://www.ittelkom.ac.id/library/index.php?view=article&catid=10%3Ajaringan&id=129%3Avoice-over-internet-protocol-voip&option=com_content&Itemid=15, diakses tanggal 21 Januari 2011.

Schulzrine, Henning. *RTP: About RTP and the Audio-Video Transport Working Group*. http://www.columbia.edu/~hgs/rtp/index.html, diakses tanggal 20 Januari 2010.

Sulistyorini, Tri. 2004. *Flowchart (Diagram Alur)*. Universitas Gunadarma, Jakarta, tri_s.staff.gunadarma.ac.id/Downloads/files/11158/FlowchartDiagramAlur.pdf, diakses tanggal 3 Januari 2011.

Sun Microsystems, Inc. 2004. *JMF API Documentations*. java.sun.com.

Tarrant, David dan Hunt, Toby. 2004. *VOIP – Voice over IP Overview*. University of Southampton, Southampton, http://users.ecs.soton.ac.uk/dct05r/guides/VOIP-Overview.pdf, diakses tanggal 21 Januari 2011.

Wicaksana, A. T. 2007. S*imulasi VOIP Over Wireless LAN (VOWLAN) pada Jaringan LAN Hotel Graha Santika Semarang*. Institut Teknologi Telkom, Bandung, http://www.ittelkom.ac.id/library/index.php?option=com_repository&Itemid=34&task=detail&nim=111020259, diakses tanggal 11 Februari 2010.

Widyawan. *ComputerNetworks*. www.te.ugm.ac.id/~widyawan/Jarkom/2.0%2520Computer%2520Networks.ppt, diakses tanggal 17 Maret 2009.

LAMPIRAN

LAMPIRAN A
Kode Sumber (*Source Code*)

File *VoiceListener.java*

```
package pctalk.proc;

import java.io.IOException;
import java.net.*;
import javax.swing.JOptionPane;
import pctalk.constant.Constants;

/*
 * VoiceListener.java
 *
 * Created on 09 Maret 2010, 16:40
 *
 * @author Mark Muhammad
 */
public class VoiceListener extends Thread {
    private DatagramSocket sockVoice;
    public VoiceListener() {
        try {
            sockVoice = new
DatagramSocket(Constants.PORT_LSTN_VOICE);
            Constants.printf("voice listener online\n");
        } catch (Exception e) {
            Constants.perrorGUI("Voice chat listener creation",
e.getMessage());
        }
    }

    @Override
    public void run() {
        try {
            String data;
            while (true) {
                byte buf[] = new byte[128];
                DatagramPacket packVoiceReq = new
DatagramPacket(buf, 128);
                sockVoice.receive(packVoiceReq);
                data = new String(packVoiceReq.getData());
                if (!data.startsWith(Constants.PCTALK_REQ)) {
                    continue;
                }
                if (data.charAt(Constants.PCTALK_REQ.length()) !=
Constants.REQ_VOICE) {
                    continue;
                }
                Constants.printf("v> received voice chat
request\n");
                if (getStatus()) {
                    data = new String(Constants.PCTALK_BSY);
                    buf = data.getBytes();
```

```
                    packVoiceReq.setData(buf);
                    Constants.printf("v> you are already in a
voice chat\n");
                    send(packVoiceReq);
                    continue;
                }
                data =
data.substring(Constants.PCTALK_REQ.length() + 1);
                String rUser = data.trim();
                int yes = JOptionPane.showConfirmDialog(null,
                        rUser + " calling..." +
                        "\nWould you like to accept?",
                        "Incoming Call",
                        JOptionPane.YES_NO_OPTION);
                if (yes == JOptionPane.YES_OPTION) {
                    if (!setStatus(packVoiceReq.getAddress())) {
                        data = Constants.PCTALK_BSY;
                    } else {
                        data = Constants.PCTALK_APR;
                    }
                } else {
                    data = Constants.PCTALK_DCL;
                }
                buf = data.getBytes();
                packVoiceReq.setData(buf);
                send(packVoiceReq);
            }
        } catch (Exception e) {
            Constants.perrorGUI("VoiceListener: run",
e.getMessage());
        }
    }

    private boolean getStatus() {
        String str = Constants.GSTVOICE;
        byte buf[] = str.getBytes();
        DatagramPacket pack;
        try {
            pack = new DatagramPacket(buf, buf.length,
InetAddress.getLocalHost(),
                    Constants.PORT_MAIN);
            send(pack);
            Constants.printf("v> requesting current voice chat
state\n\t" + str + "\n");
            sockVoice.receive(pack);
            Constants.printf("v> current state received\n");
            str = new String(pack.getData());
            if (str.compareToIgnoreCase(Constants.ON_VOICE) == 0)
{
                return true;
            } else {
                return false;
            }
        } catch (IOException ioe) {
            Constants.perror(ioe.getMessage());
        }
```

```
        return false;
    }

    private boolean setStatus(InetAddress addr) {
        String str =
Constants.SSTVOICE.concat(addr.getHostAddress());
        byte buf[] = str.getBytes();
        DatagramPacket pack;
        try {
            pack = new DatagramPacket(buf, buf.length,
InetAddress.getLocalHost(),
                    Constants.PORT_MAIN);
            send(pack);
            sockVoice.receive(pack);
            buf = pack.getData();
            if (buf[0] == 0) {
                //state setting ok
                return true;
            } else {
                //already in a voice chat
                return false;
            }
        } catch (IOException ioe) {
            Constants.perror(ioe.getMessage());
        }
        return false;
    }

    private void send(DatagramPacket pack) {
        try {
            sockVoice.send(pack);
        } catch (IOException ex) {
Constants.perrorGUI("VoiceListener: sending message: ",
e.getMessage());
        }
    }
}
```

File *VoiceChat.java*

```
package pctalk.proc;

import java.io.IOException;
import pctalk.constant.Constants;
import java.net.InetAddress;
import java.util.Vector;
import javax.media.*;
import javax.media.control.*;
import javax.media.format.AudioFormat;
import javax.media.protocol.*;
import javax.media.rtp.*;
import javax.media.rtp.event.*;
```

```
/**
 *
 * @author Mark Muhammad
 */
public class VoiceChat
        implements
        ControllerListener,
        SessionListener,
        ReceiveStreamListener,
        Runnable {
    private InetAddress addr;
    private int port;
    private String audioFormat = AudioFormat.ULAW_RTP;
    private DataSource dataOut;
    private MediaLocator audioIn;
    private Processor processor;
    private int mode;
    private Player player;
    private RTPManager[] rtpMgrs = new RTPManager[2];
    private RTPManager rtpMgr;
    private boolean dataReceived = false;
    private final Object dataSync = new Object();
    public final Object control = new Object();
    public boolean isDone = false;

    public VoiceChat(InetAddress addr, int port, int mode) {
        this.addr = addr;
        this.port = port;
        this.mode = mode;
    }

    @Override
    public void run() {
        String result = getLineIn();
        if (result != null) {
            Constants.perror(result + "\n");
            return;
        } else {
            Constants.println("audio line-in ok");
        }
        result = start();
        if (result != null) {
            Constants.perror(result + "\n");
            stop();
        }
    }

    public String start() {
        String result;
        result = createProcessor();
        if (result != null) {
            return result;
        }
        if (mode == 0) {
            result = initTransceiver();
            if (result != null) {
```

```
                return result;
            }
            Constants.println("Transceiver: \n\t" +
rtpMgr.toString());
        } else if (mode == 1) {
            result = initReceiver();
            if (result != null) {
                return result;
            }
            Constants.println("Receiver: \n\t" +
rtpMgrs[1].toString());
            result = startTransmitter();
            if (result != null) {
                return result;
            }
            Constants.println("Transmitter: \n\t" +
rtpMgrs[0].toString());
        }
        long then = System.currentTimeMillis();
        long waitingPeriod = 60000;
        try {
            synchronized (dataSync) {
                while (!dataReceived && System.currentTimeMillis()
- then < waitingPeriod) {
                    Constants.printf("Waiting for RTP data...\n");
                    dataSync.wait(5000);
                }
            }
        } catch (Exception e) {
        }
        if (!dataReceived) {
            return "No RTP data received";
        }
        return null;
    }

    public void stop() {
        synchronized (this) {
            if (getProcessor() != null) {
                getProcessor().stop();
                getProcessor().close();
                processor = null;
                Constants.println("processor stopped");
            }
            if (player != null) {
                player.close();
                player = null;
                Constants.println("player stopped");
            }
            if (mode == 0) {
                if (rtpMgr != null) {
                    rtpMgr.removeTargets("Session ended...");
                    rtpMgr.dispose();
                    Constants.println("tranceiver stopped");
                }
            }
```

```
            if (mode == 1) {
                if (rtpMgrs != null) {
                    for (int i = 0; i < rtpMgrs.length; i++) {
                        if (rtpMgrs[i] == null) {
                            continue;
                        }
                        rtpMgrs[i].removeTargets("Session
ended...");
                        rtpMgrs[i].dispose();
                        Constants.println("tranceiver stopped");
                    }
                }
            }
        }
    }

    public void addTarget(InetAddress raddr) {
        SessionAddress sessionAddrT = new SessionAddress(raddr,
port + 2);
        SessionAddress sessionAddrR = new SessionAddress(raddr,
port);
        try {
            rtpMgrs[0].addTarget(sessionAddrT);
            rtpMgrs[1].addTarget(sessionAddrR);
        } catch (InvalidSessionAddressException ex) {
            Constants.perrorGUI("Voice chat", "Error adding remote
user: invalid address");
        } catch (IOException ex) {
            Constants.perrorGUI("Voice chat", "Error adding remote
user: I/O error");
        }
    }

    private String getLineIn() {
        Vector devList = CaptureDeviceManager.getDeviceList(
                new AudioFormat("linear", 8000, 8, 1));
        if (devList.isEmpty()) {
            return "Couldn't get audio line-in device";
        }
        CaptureDeviceInfo dev = (CaptureDeviceInfo)
devList.firstElement();
        audioIn = dev.getLocator();
        return null;
    }

    private String initReceiver() {
        try {
            SessionAddress lhost, rhost;
            rtpMgrs[1] = RTPManager.newInstance();
            rtpMgrs[1].addReceiveStreamListener(this);
            rtpMgrs[1].addSessionListener(this);
            if (!addr.isMulticastAddress()) {
                lhost = new
SessionAddress(InetAddress.getLocalHost(), port + 2);
                rhost = new SessionAddress(addr, port);
```

```
            } else {
                lhost = new SessionAddress(addr, port + 2);
                rhost = new SessionAddress(addr, port);
            }
            BufferControl bc = (BufferControl)
rtpMgrs[1].getControl(
                    "javax.media.control.BufferControl");
            if (bc != null) {
                bc.setBufferLength(150);
            }
            rtpMgrs[1].initialize(lhost);
            rtpMgrs[1].addTarget(rhost);
        } catch (Exception e) {
            return "Receiver creation error: " + e.getMessage();
        }
        return null;
    }

    private synchronized String startTransmitter() {
        String result;
        result = initTransmitter();
        if (result != null) {
            return result;
        }
        getProcessor().start();
        return null;
    }

    private String createProcessor() {
        if (audioIn == null) {
            return "Input media locator is null";
        }
        try {
            DataSource ds = Manager.createDataSource(audioIn);
            processor = Manager.createProcessor(ds);
        } catch (Exception e) {
            return "Processor creation error: " + e.getMessage();
        }
        String result = prepareProcessor();
        if (result != null) {
            return result;
        }
        dataOut = getProcessor().getDataOutput();
        return null;
    }

    private String prepareProcessor() {
        boolean retval;
        retval = waitForState(getProcessor(),
getProcessor().Configured);
        if (!retval) {
            return "Couldn't configure processor";
        }
        retval = encode();
        if (!retval) {
            return "Encoding failed";
```

```
        }
        retval = waitForState(getProcessor(),
getProcessor().Realized);
        if (!retval) {
            return "Couldn't realize processor";
        }
        return null;
    }

    private boolean encode() {
        boolean success = false;
        TrackControl tracks[] = getProcessor().getTrackControls();
        if (tracks == null || tracks.length < 1) {
            return false;
        }
        getProcessor().setContentDescriptor(
                new ContentDescriptor(ContentDescriptor.RAW_RTP));
        for (int i = 0; i < tracks.length; i++) {
            if (!success && tracks[i] instanceof FormatControl) {
                if (((FormatControl) tracks[i]).setFormat(
                        new AudioFormat(audioFormat, 8000, 8, 1))
== null) {
                    tracks[i].setEnabled(false);
                } else {
                    success = true;
                }
            } else {
                tracks[i].setEnabled(false);
            }
        }
        return success;
    }

    private String initTransmitter() {
        SendStream sendStream;
        SessionAddress lhost, rhost;
        try {
            rtpMgrs[0] = RTPManager.newInstance();
            lhost = new SessionAddress(InetAddress.getLocalHost(),
port);
            rhost = new SessionAddress(addr, port + 2);
            rtpMgrs[0].initialize(lhost);
            rtpMgrs[0].addTarget(rhost);
            sendStream = rtpMgrs[0].createSendStream(dataOut, 0);
            sendStream.start();
        } catch (Exception e) {
            return "Transmitter creation error: " +
e.getMessage();
        }
        return null;
    }

    private String initTransceiver() {
        try {
            SendStream sendStream;
            SessionAddress lhost;
```

```
            SessionAddress rhost;
            rtpMgr = RTPManager.newInstance();
            rtpMgr.addReceiveStreamListener(this);
            rtpMgr.addSessionListener(this);
            lhost = new SessionAddress(InetAddress.getLocalHost(),
port);
            rhost = new SessionAddress(addr, port);
            BufferControl bc = (BufferControl) rtpMgr.getControl(
                "javax.media.control.BufferControl");
            if (bc != null) {
                bc.setBufferLength(150);
            }
            rtpMgr.initialize(lhost);
            rtpMgr.addTarget(rhost);
            sendStream = rtpMgr.createSendStream(dataOut, 0);
            sendStream.start();
        } catch (Exception ex) {
            return "Transceiver creation error";
        }
        return null;
    }

    /**
     * this method overrides the one on SessionListener
     * @param evt
     */
    @Override
    public synchronized void update(SessionEvent evt) {
        if (evt instanceof NewParticipantEvent) {
            Participant p = ((NewParticipantEvent)
evt).getParticipant();
            Constants.println("  - A new participant had just
joined: " + p.getCNAME());
        }
    }

    /**
     * this method overrides the one on ReceiveStreamListener
     * @param evt
     */
    @Override
    public synchronized void update(ReceiveStreamEvent evt) {
        Participant participant = evt.getParticipant();     //
could be null.
        ReceiveStream stream = evt.getReceiveStream();  // could
be null.
        if (evt instanceof RemotePayloadChangeEvent) {
            Constants.println("  - Received an RTP
PayloadChangeEvent.");
            Constants.println("Sorry, cannot handle payload
change.");
        } else if (evt instanceof NewReceiveStreamEvent) {
            try {
                stream = ((NewReceiveStreamEvent)
evt).getReceiveStream();
                DataSource ds = stream.getDataSource();
```

```
                // Find out the formats.
                RTPControl ctl = (RTPControl)
ds.getControl("javax.media.rtp.RTPControl");
                if (ctl != null) {
                    Constants.println("  - Recevied new RTP
stream: " + ctl.getFormat());
                } else {
                    Constants.println("  - Recevied new RTP
stream");
                }
                if (participant == null) {
                    Constants.println("      The sender of this
stream had yet to be identified.");
                } else {
                    Constants.println("      The stream comes
from: " + participant.getCNAME());
                }
                // create a player by passing datasource to the
Media Manager
                player = javax.media.Manager.createPlayer(ds);
                if (player == null) {
                    return;
                }
                player.addControllerListener(this);
                player.realize();
                synchronized (dataSync) {
                    dataReceived = true;
                    dataSync.notifyAll();
                }
            } catch (Exception e) {
                Constants.println("NewReceiveStreamEvent exception
" + e.getMessage());
                return;
            }
            System.out.println("player>> " + player.toString());
        } else if (evt instanceof StreamMappedEvent) {
            if (stream != null && stream.getDataSource() != null)
{
                DataSource ds = stream.getDataSource();
                // Find out the formats.
                RTPControl ctl = (RTPControl)
ds.getControl("javax.media.rtp.RTPControl");
                Constants.println("  - The previously unidentified
stream ");
                if (ctl != null) {
                    Constants.println("      " + ctl.getFormat());
                }
                Constants.println("      had now been identified
as sent by: " + participant.getCNAME());
            }
        } else if (evt instanceof ByeEvent) {
            Constants.println("  - Got \"bye\" from: " +
participant.getCNAME());
            synchronized (control) {
                stop();
```

```
            }
        }
    }

    /**
     * this method overrides the one on ControllerListener and is
used by player
     * @param evt
     */
    @Override
    public synchronized void controllerUpdate(ControllerEvent evt)
{
        Player p = (Player) evt.getSource();
        if (p == null) {
            return;
        }
        if (evt instanceof RealizeCompleteEvent) {
            p.start();
        }
        if (evt instanceof ControllerErrorEvent) {
            Constants.perrorGUI("Voice chat", "Player error.");
            p.removeControllerListener(this);
            p.close();
        }
    }

/***********************************************************
     * Convenience methods to handle processor's state changes.
***********************************************************/
    private Object stateLock = new Object();
    private boolean failed = false;

    Object getStateLock() {
        return stateLock;
    }

    void setFailed() {
        failed = true;
    }

    private synchronized boolean waitForState(Processor p, int
state) {
        p.addControllerListener(new StateListener());
        failed = false;

        // Call the required method on the processor
        if (state == Processor.Configured) {
            p.configure();
        } else if (state == Processor.Realized) {
            p.realize();
        }

        // Wait until we get an event that confirms the
        // success of the method, or a failure event.
        // See StateListener inner class
        while (p.getState() < state && !failed) {
```

```
            synchronized (getStateLock()) {
                try {
                    getStateLock().wait();
                } catch (InterruptedException ie) {
                    return false;
                }
            }
        }

        if (failed) {
            return false;
        } else {
            return true;
        }
    }

    /**
     * @return the processor
     */
    public Processor getProcessor() {
        return processor;
    }


/**************************************************************
     * Inner Classes
**************************************************************/
    class StateListener implements ControllerListener {
        @Override
        public void controllerUpdate(ControllerEvent ce) {
            // If there was an error during configure or
            // realize, the processor will be closed
            if (ce instanceof ControllerClosedEvent) {
                setFailed();
            }
            // All controller events, send a notification
            // to the waiting thread in waitForState method.
            if (ce instanceof ControllerEvent) {
                synchronized (getStateLock()) {
                    getStateLock().notifyAll();
                }
            }
        }
    }
}
```

File *ChatBox.java*

```
package pctalk.ui;

import java.io.*;
import java.net.*;
import pctalk.constant.Constants;
import pctalk.proc.VoiceChat;
```

```
/**
 * ChatBox.java
 *
 * Created on 29 Jul 10, 13:33:57
 *
 * @author Mark Muhammad
 */
public class ChatBox extends javax.swing.JFrame implements
Serializable, Runnable {
    private String titleName = "Mokal - ";
    private String lUser;
    private String rUser;
    private Socket sock;
    private BufferedReader dataIn;
    private PrintStream dataOut;
    private VoiceChat vc = null;
    public InetAddress addr;

    /** Creates new form ChatBox */
    public ChatBox(String loc, String rem, Socket sock) {
        this.sock = sock;
        addr = sock.getInetAddress();
        titleName = titleName.concat(rem);
        lUser = loc;
        rUser = rem;
        initComponents();
        Constants.printf("initialization completed\n");
        Constants.printf("socket connection> " +
                sock.getInetAddress() + ":" + sock.getPort() +
"\n");
        initIO();
        startOperation();
    }

    public ChatBox(String loc, String rem, InetAddress addr) {
        titleName = titleName.concat(rem);
        lUser = loc;
        rUser = rem;
        this.addr = addr;
        initComponents();
        Constants.printf("initialization completed\n");
        sendRequest();
        if (!initConnection(addr)) {
            Constants.perror("couldn't build connection\n");
            return;
        }
        if (!initIO()) {
            Constants.perror("couldn't build I/O channel");
            return;
        }
        sendSelf();
        startOperation();
    }

    public void startOperation() {
        tfInput.grabFocus();
```

```
        setVisible(true);
        new Thread(this).start();
    }

    @Override
    public void run() {
        String msg;
        Constants.printf("begin I/O reading\n");
        while (true) {
            try {
                msg = dataIn.readLine();
            } catch (IOException e) {
                if (e.toString().indexOf("closed") >= 0) {
                    continue;
                }
                Constants.perror("run: " + e.toString() + "\n");
                continue;
            }
            if (msg == null) {
                areaOutput.append("---'" + rUser + "' has closed
connection channel---\n");
                if (vc != null) {
                    hangup();
                }
                destroyConnection();
                break;
            }
            Constants.printf(msg + "\n");
            if (!msg.startsWith(rUser)) {
                if
(msg.startsWith(Constants.PCTALK_REQ.concat(String.valueOf(Constan
ts.REQ_VOICE)))) {
                    //btnVoice.setEnabled(false);
                } else {
                    continue;
                }
            }
            String data = msg.substring(rUser.length());
            if (!data.startsWith(lUser)) {
                continue;
            }
            data = msg.substring(rUser.length() + lUser.length());
            Constants.printf("received '" + data + "' from [" +
rUser + "] --> " +
                    sock.getInetAddress() + ":" + sock.getPort() +
"\n");
            areaOutput.append(rUser + "> " + data + "\n");
        }
    }

//text chat routines==============================================
private boolean initConnection(InetAddress addr) {
        try {
            sock = new Socket(addr, Constants.PORT_TEXT);
        } catch (Exception e) {
            Constants.perrorGUI("Chat connection", e.toString());
```

```
            dispose();
            return false;
        }
        Constants.printf("connection to " +
sock.getInetAddress().getHostAddress() +
                ":" + sock.getPort() + " created\n");
        return true;
    }

    private boolean initIO() {
        try {
            dataIn = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
            dataOut = new PrintStream(sock.getOutputStream());
        } catch (Exception e) {
            Constants.perrorGUI("Chat I/O initiation",
e.toString());
            return false;
        }
        Constants.printf("I/O connection with " +
sock.getInetAddress() + " initiated\n");
        return true;
    }

    private void destroyConnection() {
        destroyIO();
        if (!sock.isClosed()) {
            try {
                sock.close();
            } catch (IOException ioe) {
                Constants.perror("chat socket closing error: " +
ioe.toString()+"\n");
                return;
            }
        }
        Constants.printf("socket closed\n");
    }

    private void destroyIO() {
        if (dataOut != null)
            dataOut.close();
        if (dataIn != null)
            try {
                dataIn.close();
            } catch (IOException e) {
                Constants.perror("I/O closing error: " +
e.toString()+"\n");
                return;
            }
        Constants.printf("I/O closed\n");
    }

    private void sendRequest() {
        try {
            byte buf[] = new
byte[Constants.PCTALK_REQ.length()+1];
```

```
            String temp =
Constants.PCTALK_REQ+Constants.REQ_TCHAT;
            buf = temp.getBytes();
            DatagramSocket dsock = new DatagramSocket();
            DatagramPacket dpack = new DatagramPacket(buf,
buf.length, addr, Constants.PORT_LSTN_TEXT);
            dpack.setData(buf);
            dsock.send(dpack);
        } catch(Exception e) {
            Constants.perror("chat box: couldn't send chat
request\n\t"+e.getMessage()+"\n");
        }
        try {
            this.wait(1000);
        } catch(Exception e) {
            Constants.perror("chat box: wait interrupted\n");
        }
    }

    private void sendSelf() {
        dataOut.println(lUser);
    }

    private void sendMessage(String str) {
        String msg = lUser.concat(rUser.concat(str));
        dataOut.println(msg);
        Constants.printf("sending '" + str + "' to [" + rUser + "]
--> " +
                sock.getInetAddress() + ":" + sock.getPort() +
"\n");
        areaOutput.append(lUser + "> " + str + "\n");
    }

    private void getAndSend() {
        String msg = tfInput.getText();
        if (!msg.isEmpty()) {
            sendMessage(msg);
        }
        tfInput.setText("");
    }
//end text chat routines----------------------------------------

//voice chat routines==============================================
    public void call() {
        try {
            String reqStr = Constants.PCTALK_REQ +
Constants.REQ_VOICE + this.lUser;
            sendRequest(reqStr);
            byte buf[] = reqStr.getBytes();
            int len = buf.length;
            DatagramSocket ds = new DatagramSocket();
            DatagramPacket dp = new DatagramPacket(buf, len);
            dp.setAddress(addr);
            dp.setPort(Constants.PORT_LSTN_VOICE);
            Constants.printf("sending voice chat request to "+
```

```
dp.getAddress().getHostAddress()+":"+dp.getPort()+"\n");
            lblStatus.setText("Dialing...");
            ds.send(dp);
            Constants.printf("waiting for reply...\n");
            ds.receive(dp);
            String data = new String(dp.getData());
            if (data.startsWith(Constants.PCTALK_DCL)) {
                Constants.printGUI("Voice chat request to " +
this.rUser +
                        " was declined");
                lblStatus.setText("");
                return;
            } else if (data.startsWith(Constants.PCTALK_BSY)) {
                Constants.printGUI(this.rUser + " is busy.");
                lblStatus.setText("");
                return;
            }
            Constants.printf("voice chat request was accepted\n");
            lblStatus.setText("Request granted");
            ds.close();
        } catch (Exception e) {
            Constants.perrorGUI("Voice chat request sending",
e.getMessage());
            lblStatus.setText("");
            return;
        }
        if (!setStatus()) {
            Constants.perrorGUI("Voice chat", "Voice chat
initialization failed.");
            lblStatus.setText("");
            return;
        }
    }

    private void sendRequest(String str) {
        dataOut.println(str);
        Constants.printf("sending '" + str + "' to [" + rUser + "]
--> " +
                sock.getInetAddress() + ":" + sock.getPort() +
"\n");
    }

    private void hangup() {
        if (vc == null)
            return;
        vc.stop();
        vc = null;
        resetStatus();
        lblStatus.setText("");
        btnVoice.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/img/dial.png")));
        btnVoice.setText("Call");
    }
```

```
    public VoiceChat getVC() {
        return vc;
    }

    public void setVC(VoiceChat givenvc) {
        vc = givenvc;
        btnVoice.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/img/hangup.png")));
        btnVoice.setText("Hang Up");
        lblStatus.setText("Connected");
        Thread t = new Thread(vc);
        t.setDaemon(true);
        t.start();
    }

    private boolean setStatus() {
        String str =
Constants.SSTVOICE.concat(addr.getHostAddress());
        byte buf[] = str.getBytes();
        DatagramSocket ipcsock;
        DatagramPacket pack;
        try {
            ipcsock = new DatagramSocket();
            pack = new DatagramPacket(buf, buf.length,
InetAddress.getLocalHost(),
                    Constants.PORT_MAIN);
            ipcsock.send(pack);
            ipcsock.receive(pack);
            buf = pack.getData();
            if (buf[0] == 0) {
                //state setting ok
                return true;
            } else {
                //already in a voice chat
                return false;
            }
        } catch (IOException ioe) {
            Constants.perror(ioe.getMessage());
        }

        return false;
    }

    private void resetStatus() {
        String str = Constants.RSTVOICE;
        byte buf[] = str.getBytes();
        DatagramSocket ipcsock;
        DatagramPacket pack;
        try {
            ipcsock = new DatagramSocket();
            pack = new DatagramPacket(buf, buf.length,
InetAddress.getLocalHost(),
                    Constants.PORT_MAIN);
            ipcsock.send(pack);
        } catch (IOException ioe) {
            Constants.perror(ioe.getMessage());
```

```
        }
    }
//end voice chat routines----------------------------------------

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this
method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {
        btnConf = new javax.swing.JButton();
        panelUpper = new javax.swing.JScrollPane();
        areaOutput = new javax.swing.JTextArea();
        panelLower = new javax.swing.JPanel();
        panelInputChat = new javax.swing.JPanel();
        panelButtons = new javax.swing.JPanel();
        btnVoice = new javax.swing.JButton();
        panelInput = new javax.swing.JPanel();
        tfInput = new javax.swing.JTextField();
        btnSend = new javax.swing.JButton();
        panelStatus = new javax.swing.JPanel();
        lblStatus = new javax.swing.JLabel();
        btnConf.setText("Text Conference");
btnConf.setBorder(javax.swing.BorderFactory.createEmptyBorder(1,
1, 1, 1));
setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CL
OSE);
        setTitle(titleName);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosed(java.awt.event.WindowEvent
evt) {
                formWindowClosed(evt);
            }
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                formWindowClosing(evt);
            }
        });
        areaOutput.setColumns(20);
        areaOutput.setEditable(false);
        areaOutput.setLineWrap(true);
        areaOutput.setRows(5);
        areaOutput.setWrapStyleWord(true);
areaOutput.setBorder(javax.swing.BorderFactory.createEmptyBorder(1
, 1, 1, 1));
        panelUpper.setViewportView(areaOutput);
        getContentPane().add(panelUpper,
java.awt.BorderLayout.CENTER);
        panelLower.setLayout(new java.awt.BorderLayout());
        panelInputChat.setLayout(new java.awt.BorderLayout());
```

```
        panelButtons.setLayout(new
java.awt.FlowLayout(java.awt.FlowLayout.LEFT, 10, 5));
        btnVoice.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/img/dial.png")));
// NOI18N
        btnVoice.setText("Call");
btnVoice.setBorder(javax.swing.BorderFactory.createEmptyBorder(1,
1, 1, 1));
        btnVoice.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnVoiceActionPerformed(evt);
            }
        });
        panelButtons.add(btnVoice);
        panelInputChat.add(panelButtons,
java.awt.BorderLayout.NORTH);
        panelInput.setLayout(new java.awt.BorderLayout());
        tfInput.setFont(new java.awt.Font("Courier New", 0, 12));
// NOI18N
        tfInput.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                tfInputActionPerformed(evt);
            }
        });
        panelInput.add(tfInput, java.awt.BorderLayout.CENTER);
        btnSend.setText("Send");
        btnSend.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnSendActionPerformed(evt);
            }
        });
        panelInput.add(btnSend, java.awt.BorderLayout.EAST);
        panelInputChat.add(panelInput,
java.awt.BorderLayout.CENTER);
        panelLower.add(panelInputChat,
java.awt.BorderLayout.CENTER);
panelStatus.setBorder(javax.swing.BorderFactory.createEtchedBorder
());
        panelStatus.setMinimumSize(new java.awt.Dimension(29,
25));
        panelStatus.setPreferredSize(new java.awt.Dimension(29,
25));
        panelStatus.setLayout(new
java.awt.FlowLayout(java.awt.FlowLayout.LEFT));
        panelStatus.add(lblStatus);
        panelLower.add(panelStatus, java.awt.BorderLayout.SOUTH);
        getContentPane().add(panelLower,
java.awt.BorderLayout.SOUTH);
```

```
        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-408)/2, (screenSize.height-
327)/2, 408, 327);
    }// </editor-fold>//GEN-END:initComponents

    private void tfInputActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_tfInputActionPerformed
        getAndSend();
}//GEN-LAST:event_tfInputActionPerformed

    private void btnSendActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_btnSendActionPerformed
        getAndSend();
}//GEN-LAST:event_btnSendActionPerformed

    private void formWindowClosing(java.awt.event.WindowEvent evt)
{//GEN-FIRST:event_formWindowClosing
        if (vc != null)
            hangup();
        destroyConnection();
    }//GEN-LAST:event_formWindowClosing

    private void
btnVoiceActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnVoiceActionPerformed
        if (evt.getActionCommand().compareTo("Call") == 0) {
            call();
        } else {
            hangup();
        }
    }//GEN-LAST:event_btnVoiceActionPerformed

    private void formWindowClosed(java.awt.event.WindowEvent evt)
{//GEN-FIRST:event_formWindowClosed
        if (vc != null)
            hangup();

        destroyConnection();
    }//GEN-LAST:event_formWindowClosed

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JTextArea areaOutput;
    private javax.swing.JButton btnConf;
    private javax.swing.JButton btnSend;
    public javax.swing.JButton btnVoice;
    private javax.swing.JLabel lblStatus;
    private javax.swing.JPanel panelButtons;
    private javax.swing.JPanel panelInput;
    private javax.swing.JPanel panelInputChat;
    private javax.swing.JPanel panelLower;
    private javax.swing.JPanel panelStatus;
    private javax.swing.JScrollPane panelUpper;
    private javax.swing.JTextField tfInput;
    // End of variables declaration//GEN-END:variables
}
```

File *TextConference.java*

```
/**
 * TextConference.java
 *
 * Created on 13 Jan 11, 15:11:27
 */
package pctalk.ui;

import java.io.IOException;
import java.net.*;
import java.util.List;
import javax.swing.DefaultListModel;
import pctalk.constant.Constants;
import pctalk.proc.*;

/**
 *
 * @author Mark Muhammad
 */
public class TextConference extends javax.swing.JFrame implements
Runnable {
    private InetAddress maddr;
    private InetAddress laddr;
    private int channel;
    private int port;
    private MulticastSocket msock;
    private DefaultListModel memberLst;
    private String nickname;
    private ListenerDaemon listener;
    private VoiceChat vc = null;
    private String ENABLEVC = "Enable voice chat";
    private String DISABLEVC= "Disable voice chat";
    private String inviter = null;

    /** Creates new form TextConference */
    public TextConference(
            InetAddress laddr,
            InetAddress addr,
            int port,
            String nick,
            int chnnum,
            ListenerDaemon listener) {
        maddr = addr;
        this.laddr = laddr;
        this.port = port;
        nickname = nick;
        channel = chnnum;
        this.listener = listener;
        start();
    }

    public TextConference(
            InetAddress laddr,
            InetAddress addr,
            int port,
```

```
            String nick,
            int chnnum,
            ListenerDaemon listener,
            String inviter) {
        maddr = addr;
        this.laddr = laddr;
        this.port = port;
        nickname = nick;
        channel = chnnum;
        this.listener = listener;
        this.inviter = inviter;
        start();
    }

    private void start() {
        memberLst = new DefaultListModel();
        try {
            msock = new MulticastSocket(this.port);
            msock.joinGroup(maddr);
        } catch (Exception e) {
            Constants.perrorGUI("conference", "Construction error:
\n\t" +
                    e.toString());
        }

        initComponents();
        btnAdd.grabFocus();
        if (inviter != null) {
            txtareaChat.append("---You were invited by
"+inviter+"\n");
            fetchXstUsr();
        }
        new Thread(this).start();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this
method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {
        panelUser = new javax.swing.JPanel();
        panelMain = new javax.swing.JPanel();
        panelUserBtn = new javax.swing.JPanel();
        btnAdd = new javax.swing.JButton();
        btnRm = new javax.swing.JButton();
        btnVoice = new javax.swing.JButton();
        btnLeave = new javax.swing.JButton();
        scrpaneUser = new javax.swing.JScrollPane();
        lstUser = new javax.swing.JList();
        scrpaneChat = new javax.swing.JScrollPane();
        txtareaChat = new javax.swing.JTextArea();
```

```
        panelInput = new javax.swing.JPanel();
        btnSend = new javax.swing.JButton();
        scrpaneInputChat = new javax.swing.JScrollPane();
        txtInputChat = new javax.swing.JTextField();
        panelUser.setLayout(new java.awt.BorderLayout());
setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON
_CLOSE);
        setTitle("Mokal - Conference");
        getContentPane().setLayout(new java.awt.BorderLayout(0,
5));
        panelMain.setLayout(new java.awt.BorderLayout(5, 0));
        panelUserBtn.setLayout(new
java.awt.FlowLayout(java.awt.FlowLayout.LEFT));
        btnAdd.setText("Add user");
        btnAdd.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnAddActionPerformed(evt);
            }
        });
        panelUserBtn.add(btnAdd);
        btnRm.setText("Remove user");
        panelUserBtn.add(btnRm);
        btnVoice.setText(ENABLEVC);
        btnVoice.setToolTipText("Enable voice chat");
        btnVoice.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnVoiceActionPerformed(evt);
            }
        });
        panelUserBtn.add(btnVoice);
        btnLeave.setText("Leave group");
        btnLeave.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnLeaveActionPerformed(evt);
            }
        });
        panelUserBtn.add(btnLeave);
        panelMain.add(panelUserBtn, java.awt.BorderLayout.NORTH);
        scrpaneUser.setPreferredSize(new java.awt.Dimension(150,
132));
        lstUser.setModel(memberLst);
        scrpaneUser.setViewportView(lstUser);
        panelMain.add(scrpaneUser, java.awt.BorderLayout.EAST);
        txtareaChat.setColumns(20);
        txtareaChat.setEditable(false);
        txtareaChat.setRows(5);
        scrpaneChat.setViewportView(txtareaChat);
        panelMain.add(scrpaneChat, java.awt.BorderLayout.CENTER);
```

```
        getContentPane().add(panelMain,
java.awt.BorderLayout.CENTER);
        panelInput.setLayout(new java.awt.BorderLayout(5, 0));
        btnSend.setText("Send");
        btnSend.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnSendActionPerformed(evt);
            }
        });
        panelInput.add(btnSend, java.awt.BorderLayout.EAST);
        scrpaneInputChat.setBorder(null);
        scrpaneInputChat.setPreferredSize(new
java.awt.Dimension(59, 50));
txtInputChat.setHorizontalAlignment(javax.swing.JTextField.LEFT);
        txtInputChat.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                txtInputChatActionPerformed(evt);
            }
        });
        scrpaneInputChat.setViewportView(txtInputChat);
        panelInput.add(scrpaneInputChat,
java.awt.BorderLayout.CENTER);
        getContentPane().add(panelInput,
java.awt.BorderLayout.SOUTH);
        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-600)/2, (screenSize.height-
500)/2, 600, 500);
    }// </editor-fold>//GEN-END:initComponents

    @Override
    public void run() {
        byte buf[] = new byte[256];
        DatagramPacket dpack = null;
        String strdata;
        try {
            dpack = new DatagramPacket(buf, buf.length);
        } catch (Exception e) {
            Constants.perror("conference run: " + e.toString());
        }
        while (true) {
            try {
                msock.receive(dpack);
                strdata = new String(dpack.getData(), 0,
dpack.getLength());
                if (strdata.startsWith(Constants.RSTFORUM)) {
                    String off =
strdata.substring(Constants.RSTFORUM.length());
                    memberLst.removeElement(off);
                    txtareaChat.append("---"+off+" has left
conference\n" );
```

```
                } else if (strdata.startsWith(Constants.ADFRMUSR))
{
                    String newUsr =
strdata.substring(Constants.ADFRMUSR.length());
                    memberLst.addElement(newUsr);
                    txtareaChat.append("---"+newUsr+" has joined
conference\n");
                } else if (strdata.startsWith(Constants.FTCHUSER))
{
                    sendSelf();
                } else if (strdata.startsWith(Constants.XSFRMUSR))
{
                    String xstingUsr =
strdata.substring(Constants.XSFRMUSR.length());
                    Constants.println("existing member:
"+xstingUsr);
                    if (xstingUsr.compareTo(nickname) != 0 &&
!memberLst.contains(xstingUsr))
                        memberLst.addElement(xstingUsr);
                } else {
                    txtareaChat.append(strdata + "\n");
                }
            } catch (IOException ex) {
                Constants.perror("conference run: " +
ex.toString());
            }
        }
    }

    public InetAddress getBroadcast(InetAddress laddr) {
        InetAddress lhost = laddr;
        InetAddress retVal = null;
        try {
            NetworkInterface iface =
NetworkInterface.getByInetAddress(lhost);
            List<InterfaceAddress> ifaddrs =
iface.getInterfaceAddresses();
            if (!ifaddrs.isEmpty()) {
                for (int i = 0; i < ifaddrs.size(); i++) {
                    InterfaceAddress ifaddr = ifaddrs.get(i);
                    InetAddress addr = ifaddr.getAddress();
                    if (addr.equals(lhost)) {
                        retVal = ifaddr.getBroadcast();
                        break;
                    }
                }
            }
        } catch (Exception e) {
            Constants.perrorGUI("Main listener broadcast
retriever", e.toString());
        }
        return retVal;
    }//end method

    private void fetchXstUsr() {
        byte[] buf = Constants.FTCHUSER.getBytes();
```

```
        try {
            DatagramPacket p = new DatagramPacket(buf, buf.length,
maddr, port);
            Constants.println("fetch member: "+new String(buf));
            msock.send(p);
        } catch(Exception e) {
            Constants.perror("conference fetch member:
"+e.toString());
        }
    }

    private void sendSelf() {
        byte[] buf =
Constants.XSFRMUSR.concat(nickname).getBytes();
        try {
            DatagramPacket p = new DatagramPacket(buf, buf.length,
maddr, port);
            Constants.println("self sending: "+new String(buf));
            msock.send(p);
        } catch(Exception e) {
            Constants.perror("conference send self:
"+e.toString());
        }
    }

    private void
btnLeaveActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnLeaveActionPerformed
        String chnstr = Integer.toString(channel);
        byte[] buf;
        try {
            buf = Constants.RSTFORUM.concat(nickname).getBytes();
            DatagramPacket dpack = new DatagramPacket(
                    buf,
                    buf.length,
                    maddr,
                    port);
            msock.send(dpack);
            msock.leaveGroup(maddr);
            if (memberLst.size() == 0) {
                DatagramSocket dsock = new DatagramSocket();
                buf= Constants.RSTFORUM.concat(chnstr).getBytes();
                dpack.setData(buf);
                InetAddress bcast = getBroadcast(laddr);
                dpack.setAddress(bcast);
                dpack.setPort(Constants.PORT_MAIN);
                dsock.send(dpack);
            }
        } catch (IOException ex) {
            Constants.perror("conference leave grp: " +
ex.toString());
        }
        hangup();
        dispose();
}//GEN-LAST:event_btnLeaveActionPerformed
```

```
    private void btnSendActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_btnSendActionPerformed
        getAndSend();
    }//GEN-LAST:event_btnSendActionPerformed

    private void getAndSend() {
        if (txtInputChat.getText().trim().isEmpty()) {
            return;
        }
        byte buf[] = new byte[txtInputChat.getText().length() +
nickname.length() + 2];
        String chatdata = txtInputChat.getText().trim();
        String data = nickname.concat("> ").concat(chatdata);
        DatagramPacket dpack = null;
        buf = data.getBytes();
        try {
            dpack = new DatagramPacket(buf, buf.length, maddr,
port);
            msock.send(dpack);
        } catch (Exception e) {
            Constants.perror("conference send text: " +
e.toString());
        }
        txtInputChat.setText("");
    }

    private void btnAddActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_btnAddActionPerformed
        NicknameListDialog nlstdialog = new NicknameListDialog(
                this,
                true,
                memberLst,
                listener,
                msock,
                maddr,
                port,
                channel,
                nickname);
        nlstdialog.setVisible(true);
    }//GEN-LAST:event_btnAddActionPerformed

    private void
btnVoiceActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnVoiceActionPerformed
        if (evt.getActionCommand().contains("Enable")) {
            vc = new VoiceChat(maddr, port+1, 0);
            Thread t = new Thread(vc);
            t.setDaemon(true);
            t.start();
            synchronized (this) {
                while (vc.getProcessor() == null) {
                    try {
                        wait(1000);
                        continue;
                    } catch (InterruptedException ex) {
                    }
```

```
                }
                btnVoice.setText(DISABLEVC);
            }
        } else {
            hangup();
        }
    }//GEN-LAST:event_btnVoiceActionPerformed

    private void
txtInputChatActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_txtInputChatActionPerformed
        getAndSend();
    }//GEN-LAST:event_txtInputChatActionPerformed

    private void hangup() {
        if (vc != null) {
                vc.stop();
                vc = null;
            }
            btnVoice.setText(ENABLEVC);
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnAdd;
    private javax.swing.JButton btnLeave;
    private javax.swing.JButton btnRm;
    private javax.swing.JButton btnSend;
    private javax.swing.JButton btnVoice;
    private javax.swing.JList lstUser;
    private javax.swing.JPanel panelInput;
    private javax.swing.JPanel panelMain;
    private javax.swing.JPanel panelUser;
    private javax.swing.JPanel panelUserBtn;
    private javax.swing.JScrollPane scrpaneChat;
    private javax.swing.JScrollPane scrpaneInputChat;
    private javax.swing.JScrollPane scrpaneUser;
    private javax.swing.JTextField txtInputChat;
    private javax.swing.JTextArea txtareaChat;
    // End of variables declaration//GEN-END:variables
}
```

File *MainGUI.java*

```
package pctalk.ui;

/**
 *  @file    : Mokal.java
 *  @version : 3.0
 *  @author  : Mark Muhammad <ken_igarashi2002@yahoo.com>
 *
 *  Created on 17 February 2010
 *
 *  Copyright (c) 2009, 2010 by Mark Muhammad
 *
 */
```

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.*;
import java.net.*;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import pctalk.constant.Constants;
import pctalk.proc.*;

public class MainGUI extends javax.swing.JFrame {
    private final String HAS_NAME = "You are online as ";
    private final String NO_NAME = "You have not entered a
nickname";
    private InetAddress ipaddr;
    private ListenerDaemon listener;
    private ArrayList chatList;
    private IPCChannel ipcchannel;
    private ChatHandler chathandler;
    public String nickname;

    /** Creates new form Mokal */
    public MainGUI() {
        initComponents();
        listNickname.grabFocus();
        setVisible(true);
        new DialogNickEntry(this, true).setVisible(true);
        if (nickname == null) {
            try {
                nickname =
InetAddress.getLocalHost().getHostAddress();
            } catch (Exception e) {
            }
        }
        chatList = new ArrayList();
        listener = new ListenerDaemon(nickname);
        setIpaddr(listener.getLaddr());
        listNickname.setModel(listener.list);
        ipcchannel = new IPCChannel();
        chathandler = new ChatHandler();
        lblStatus.setText("IP: " + getIpaddr().getHostAddress());
    }

    private void addCB(ChatBox cb) {
        chatList.add(cb);
    }

    /**
     * searching for a chat with the specified ip address
     * @param addr ip address of the remote user
     * @return ChatBox object of the corresponding chat or null if
no connection
     *      has been made with the specified address
     */
    private ChatBox find(InetAddress addr) {
```

```
        if (!chatList.isEmpty()) {
            System.out.println("searching for chat box with remote
address = " + addr);
            for (int i = 0; i < chatList.size(); i++) {
                ChatBox cb = (ChatBox) chatList.get(i);
                System.out.println("chat box title: " +
cb.getTitle() + "\n\taddress: " + cb.addr);
                if
(cb.addr.getHostAddress().compareTo(addr.getHostAddress()) == 0) {
                    return cb;
                }
            }
            System.out.println("search completed, chat box not
found");
        } else {
            System.out.println("chat list is empty, no active
chat");
        }
        return null;
    }

    private final void setIP(InetAddress src) {
        InetAddress temp = src;
        if (src != null) {
            if
(temp.getHostAddress().compareTo(ipaddr.getHostAddress()) != 0) {
                setIpaddr(temp);
            }
        }
    }

    public InetAddress getIpaddr() {
        return ipaddr;
    }

    public void setIpaddr(InetAddress ipaddr) {
        if (this.ipaddr != null) {
listener.sendMessage(String.valueOf(Constants.MSG_LEAVE));
            this.ipaddr = ipaddr;
listener.sendMessage(String.valueOf(Constants.MSG_HELLO));
        } else {
            this.ipaddr = ipaddr;
        }
    }

    private void chat(int flag) {
        String temp;
        if (txtIP.getText().isEmpty()) {
            temp = (String) listNickname.getSelectedValue();
        } else {
            temp = new String(txtIP.getText());
            txtIP.setText("");
            personDisable();
        }
        if (temp.isEmpty()) {
            return;
```

```
        }

        InetAddress addr = listener.getNickAddress(temp);
        String rUser = null;
        if (addr != null)
            rUser = listener.getAddressNick(addr);
        if (addr == null && rUser == null) {
            try {
                addr = InetAddress.getByName(temp);
                rUser = temp;
            } catch (Exception e) {
                Constants.perrorGUI("Chat initiation",
e.toString());
                return;
            }
        }
        ChatBox existing = find(addr);
        if (existing != null) {
            return;
        }
        final ChatBox cb = new ChatBox(nickname, rUser, addr);
        cb.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosed(WindowEvent evt) {
                chatList.remove(cb);
            }
        });
        addCB(cb);
        if (flag == 1) {
            cb.call();
        }
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this
method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {
        btnJoinRoom = new javax.swing.JButton();
        btnMakeRoom = new javax.swing.JButton();
        panelMain = new javax.swing.JPanel();
        panelListHolder = new javax.swing.JPanel();
        panelList = new javax.swing.JPanel();
        lblNickList = new javax.swing.JLabel();
        scrpaneMain = new javax.swing.JScrollPane();
        listNickname = new javax.swing.JList();
        panelIPInput = new javax.swing.JPanel();
        lblIP = new javax.swing.JLabel();
        txtIP = new javax.swing.JTextField();
        panelBefore = new javax.swing.JPanel();
        panelAfter = new javax.swing.JPanel();
        panelUpper = new javax.swing.JPanel();
```

```
        lblNickname = new javax.swing.JLabel();
        btnInfo = new javax.swing.JButton();
        panelUpperMenu = new javax.swing.JPanel();
        btnChNick = new javax.swing.JButton();
        btnChIP = new javax.swing.JButton();
        panelButtons = new javax.swing.JPanel();
        btnTextChat = new javax.swing.JButton();
        btnConf = new javax.swing.JButton();
        btnVoiceChat = new javax.swing.JButton();
        panelStatus = new javax.swing.JPanel();
        lblStatus = new javax.swing.JLabel();
        btnJoinRoom.setText("Join Room");
        btnJoinRoom.setToolTipText("Join conference");
btnJoinRoom.setBorder(javax.swing.BorderFactory.createEmptyBorder(
1, 1, 1, 1));
        btnJoinRoom.setEnabled(false);
        btnMakeRoom.setText("Create a Room");
        btnMakeRoom.setToolTipText("Initiate conference");
btnMakeRoom.setBorder(javax.swing.BorderFactory.createEmptyBorder(
1, 1, 1, 1));
        btnMakeRoom.setEnabled(false);
btnMakeRoom.setHorizontalTextPosition(javax.swing.SwingConstants.C
ENTER);
btnMakeRoom.setVerticalTextPosition(javax.swing.SwingConstants.BOT
TOM);
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE
);      setTitle("Mokal");
        setMinimumSize(getPreferredFormSize());
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
evt) {
                formWindowClosing(evt);
            }
        });
        panelMain.setLayout(new java.awt.BorderLayout(0, 5));
        panelListHolder.setLayout(new java.awt.BorderLayout());
        panelList.setLayout(new java.awt.BorderLayout());
        lblNickList.setText("Pick a nickname from the list");
        lblNickList.setMaximumSize(new java.awt.Dimension(168,
15));
        lblNickList.setMinimumSize(new java.awt.Dimension(168,
15));
        lblNickList.setPreferredSize(new java.awt.Dimension(168,
15));
        panelList.add(lblNickList, java.awt.BorderLayout.NORTH);
scrpaneMain.setBorder(javax.swing.BorderFactory.createEmptyBorder(
1, 1, 1, 1));
        listNickname.setFont(new java.awt.Font("Courier New", 0,
14));
listNickname.setSelectionMode(javax.swing.ListSelectionModel.SINGL
E_SELECTION);
        listNickname.setFixedCellHeight(20);
        listNickname.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent
evt) {
```

```
                listNicknameMouseClicked(evt);
            }
        });
        listNickname.addListSelectionListener(new
javax.swing.event.ListSelectionListener() {
            public void
valueChanged(javax.swing.event.ListSelectionEvent evt) {
                listNicknameValueChanged(evt);
            }
        });
        scrpaneMain.setViewportView(listNickname);
        panelList.add(scrpaneMain, java.awt.BorderLayout.CENTER);
        panelIPInput.setLayout(new java.awt.GridLayout(2, 0));
        lblIP.setText("or insert an IP address");
        lblIP.setMaximumSize(new java.awt.Dimension(140, 15));
        lblIP.setMinimumSize(new java.awt.Dimension(140, 15));
        lblIP.setPreferredSize(new java.awt.Dimension(140, 15));
        panelIPInput.add(lblIP);
txtIP.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(153, 153, 153)));
        txtIP.setMaximumSize(new java.awt.Dimension(150, 20));
        txtIP.setMinimumSize(new java.awt.Dimension(150, 20));
        txtIP.setPreferredSize(new java.awt.Dimension(150, 20));
        txtIP.addFocusListener(new java.awt.event.FocusAdapter() {
            public void focusGained(java.awt.event.FocusEvent evt)
{
                txtIPFocusGained(evt);
            }
        });
        txtIP.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                txtIPKeyReleased(evt);
            }
        });
        panelIPInput.add(txtIP);
        panelList.add(panelIPInput, java.awt.BorderLayout.SOUTH);
        panelListHolder.add(panelList,
java.awt.BorderLayout.CENTER);
        panelBefore.setPreferredSize(new java.awt.Dimension(3,
10));
        panelListHolder.add(panelBefore,
java.awt.BorderLayout.LINE_START);
        panelAfter.setPreferredSize(new java.awt.Dimension(3,
10));
        panelListHolder.add(panelAfter,
java.awt.BorderLayout.LINE_END);
        panelMain.add(panelListHolder,
java.awt.BorderLayout.CENTER);
        panelUpper.setPreferredSize(new java.awt.Dimension(255,
40));
        panelUpper.setLayout(new java.awt.BorderLayout());
        lblNickname.setText(NO_NAME);
        panelUpper.add(lblNickname, java.awt.BorderLayout.CENTER);
        btnInfo.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/img/info16.png")));
        btnInfo.setBorder(null);
```

```
        btnInfo.setFocusable(false);
        btnInfo.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnInfoActionPerformed(evt);
            }
        });
        panelUpper.add(btnInfo, java.awt.BorderLayout.EAST);
        panelUpperMenu.setLayout(new java.awt.GridLayout(1, 0));
        btnChNick.setText("Change Nickname");
        btnChNick.setToolTipText("Change your current nickname");
btnChNick.setBorder(javax.swing.BorderFactory.createEmptyBorder(1,
1, 1, 1));
        btnChNick.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnChNickActionPerformed(evt);
            }
        });
        panelUpperMenu.add(btnChNick);
        btnChIP.setText("Change IP Address");
        btnChIP.setToolTipText("Change your current IP address");
btnChIP.setBorder(javax.swing.BorderFactory.createEmptyBorder(1,
1, 1, 1));
        btnChIP.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnChIPActionPerformed(evt);
            }
        });
        panelUpperMenu.add(btnChIP);
        panelUpper.add(panelUpperMenu,
java.awt.BorderLayout.NORTH);
        panelMain.add(panelUpper, java.awt.BorderLayout.NORTH);
        panelButtons.setMinimumSize(new java.awt.Dimension(243,
35));
        panelButtons.setPreferredSize(new java.awt.Dimension(255,
40));
        panelButtons.setLayout(new java.awt.GridLayout(1, 3));
        btnTextChat.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/img/chat-
icon.png"))); // NOI18N
        btnTextChat.setToolTipText("Text chat");
        btnTextChat.setBorder(null);
        btnTextChat.setEnabled(false);
btnTextChat.setHorizontalTextPosition(javax.swing.SwingConstants.C
ENTER);
        btnTextChat.setMaximumSize(new java.awt.Dimension(48,
48));
        btnTextChat.setMinimumSize(new java.awt.Dimension(48,
48));
        btnTextChat.setPreferredSize(new java.awt.Dimension(48,
48));
```

```
btnTextChat.setVerticalTextPosition(javax.swing.SwingConstants.BOT
TOM);
        btnTextChat.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnTextChatActionPerformed(evt);
            }
        });
        panelButtons.add(btnTextChat);
        btnConf.setText("Create Conference");
        btnConf.setToolTipText("Create Conference");
        btnConf.setBorder(null);
        btnConf.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnConfActionPerformed(evt);
            }
        });
        panelButtons.add(btnConf);
        btnVoiceChat.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/img/voice-chat-
icon.png"))); // NOI18N
        btnVoiceChat.setToolTipText("Voice Chat");
        btnVoiceChat.setBorder(null);
        btnVoiceChat.setEnabled(false);
btnVoiceChat.setHorizontalTextPosition(javax.swing.SwingConstants.
CENTER);
        btnVoiceChat.setMaximumSize(new java.awt.Dimension(48,
48));
        btnVoiceChat.setMinimumSize(new java.awt.Dimension(48,
48));
        btnVoiceChat.setPreferredSize(new java.awt.Dimension(48,
48));
btnVoiceChat.setVerticalTextPosition(javax.swing.SwingConstants.BO
TTOM);
        btnVoiceChat.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                btnVoiceChatActionPerformed(evt);
            }
        });
        panelButtons.add(btnVoiceChat);
        panelMain.add(panelButtons, java.awt.BorderLayout.SOUTH);
        getContentPane().add(panelMain,
java.awt.BorderLayout.CENTER);
panelStatus.setBorder(javax.swing.BorderFactory.createEtchedBorder
());
        panelStatus.setMinimumSize(new java.awt.Dimension(255,
20));
        panelStatus.setPreferredSize(new java.awt.Dimension(255,
20));
        panelStatus.setLayout(new java.awt.BorderLayout());
```

```
        panelStatus.add(lblStatus, java.awt.BorderLayout.WEST);
        getContentPane().add(panelStatus,
java.awt.BorderLayout.SOUTH);
        pack();
        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        java.awt.Dimension dialogSize = getSize();
        setLocation((screenSize.width-
dialogSize.width)/2,(screenSize.height-dialogSize.height)/2);
    }// </editor-fold>//GEN-END:initComponents

    private void btnChIPActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_btnChIPActionPerformed
        final ViewIP chIPDialog = new ViewIP((java.awt.Frame)
this, true, ipaddr);
        chIPDialog.addWindowListener(new
java.awt.event.WindowAdapter() {
            @Override
            public void windowClosed(java.awt.event.WindowEvent
evt) {
                setIP(chIPDialog.getIpAddr());
            }
        });
        chIPDialog.setVisible(true);
    }//GEN-LAST:event_btnChIPActionPerformed

    private Dimension getScreenSize() {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        return toolkit.getScreenSize();
    }

    private int getScreenHeight() {
        return getScreenSize().height;
    }

    private int getScreenWidth() {
        return getScreenSize().width;
    }

    private Dimension getPreferredFormSize() {
        int h = getScreenHeight();
        h -= 50;
        if (h >= 700) {
            h = 700;
        }
        return new Dimension(this.getWidth(), h);
    }

    private void
listNicknameValueChanged(javax.swing.event.ListSelectionEvent evt)
{//GEN-FIRST:event_listNicknameValueChanged
        if (listNickname.getSelectedIndex() > -1) {
            personEnable();
        }
    }//GEN-LAST:event_listNicknameValueChanged
```

```
    private void
btnTextChatActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnTextChatActionPerformed
        chat(0);
    }//GEN-LAST:event_btnTextChatActionPerformed

    private void txtIPKeyReleased(java.awt.event.KeyEvent evt)
{//GEN-FIRST:event_txtIPKeyReleased
        if (txtIP.getText().compareTo("") != 0) {
            personEnable();
        } else {
            personDisable();
        }
    }//GEN-LAST:event_txtIPKeyReleased

    private void
btnChNickActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnChNickActionPerformed
        String temp = new String(nickname);
        new DialogNickEntry(this, true).setVisible(true);
        if (nickname.compareTo(temp) != 0) {
            String msg = String.valueOf(Constants.MSG_NNAME);
            listener.sendMessage(msg + nickname);
        }
    }//GEN-LAST:event_btnChNickActionPerformed

    private void formWindowClosing(java.awt.event.WindowEvent evt)
{//GEN-FIRST:event_formWindowClosing
        listener.sendMessage(String.valueOf(Constants.MSG_LEAVE));
    }//GEN-LAST:event_formWindowClosing

    private void txtIPFocusGained(java.awt.event.FocusEvent evt)
{//GEN-FIRST:event_txtIPFocusGained
        listNickname.clearSelection();
        personDisable();
    }//GEN-LAST:event_txtIPFocusGained

    private void btnInfoActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_btnInfoActionPerformed
        new About(this, true).setVisible(true);
    }//GEN-LAST:event_btnInfoActionPerformed

    private void
btnVoiceChatActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnVoiceChatActionPerformed
        chat(1);
    }//GEN-LAST:event_btnVoiceChatActionPerformed

    private void
listNicknameMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_listNicknameMouseClicked
        if ((evt.getClickCount() == 2) &&
(listNickname.getSelectedIndex() >= 0)) {
            chat(0);
        }
    }//GEN-LAST:event_listNicknameMouseClicked
```

```
    private void btnConfActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_btnConfActionPerformed
        new ForumChannelDialog(
                this,
                true,
                ipcchannel.addrpairs,
                getIpaddr(),
                nickname,
                listener).setVisible(true);
    }//GEN-LAST:event_btnConfActionPerformed

    private void personEnable() {
        btnTextChat.setEnabled(true);
        btnVoiceChat.setEnabled(true);
    }

    private void personDisable() {
        btnTextChat.setEnabled(false);
        btnVoiceChat.setEnabled(false);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new MainGUI();
            }
        });
    }

    /**
     * inner class for communication between components within
application
     */
    private class IPCChannel implements Runnable {
        private byte[] buf;
        private DatagramSocket ipcsock;
        private DatagramPacket ipcpack;
        public Object[][] addrpairs = {
            {"225.1.1.1:5111", 0},
            {"225.2.1.1:5211", 0},
            {"225.3.1.1:5311", 0},
            {"225.4.1.1:5411", 0},
            {"225.5.1.1:5511", 0},
            {"225.6.1.1:5611", 0},
            {"225.7.1.1:5711", 0},
            {"225.8.1.1:5811", 0},
            {"225.9.1.1:5911", 0},
            {"225.10.1.1:5011", 0}
        };

        IPCChannel() {
```

```
            try {
                buf = new byte[128];
                ipcsock = new DatagramSocket(Constants.PORT_MAIN);
                ipcpack = new DatagramPacket(buf, buf.length);
            } catch (Exception e) {
                Constants.perrorGUI("IPC channel creation",
e.toString());
                System.exit(1);
            }
            new Thread(this).start();
        }

        @Override
        public void run() {
            boolean state = false;
            String replyStr;
            String addrStr;
            VoiceChat vc;
            Constants.printf(">> ipc channel is online\n\t" +
                    ipcsock.getLocalAddress() +
                    ":" + ipcsock.getLocalPort() + "\n");
            while (true) {
                buf = new byte[128];
                try {
                    ipcpack = new DatagramPacket(buf, buf.length);
                    ipcsock.receive(ipcpack);
                    String dataStr = new String(ipcpack.getData(),
0,
                            ipcpack.getData().length);
                    System.out.println("data is: " + dataStr);
                    if (dataStr.startsWith(Constants.RSTVOICE)) {
                        System.out.println("v> state reset");
                        state = false;
                    } else if
(dataStr.startsWith(Constants.RSTFORUM)) {
                        String temp =
dataStr.substring(Constants.RSTFORUM.length()).trim();
                        int idx = new Integer(temp);
                        addrpairs[idx - 1][1] = 0;
                    } else if
(dataStr.startsWith(Constants.SSTFORUM)) {
                        String temp =
dataStr.substring(Constants.RSTFORUM.length()).trim();
                        int idx = new Integer(temp);
                        addrpairs[idx - 1][1] = 1;
                    } else if
(dataStr.startsWith(Constants.SSTTCHAT)) {
                        System.out.println("t> new text chat
initiation requested");
                        chathandler.newConn();
                    } else if
(dataStr.startsWith(Constants.SSTTCONF)) {
                        dataStr = dataStr.trim();
                        Constants.println("new conference request
"+dataStr);
                        String[] data = dataStr.substring(
```

```
Constants.SSTTCONF.length()).split(":");
                        String inviter = data[0];
                        InetAddress maddr = null;
                        try {
                            maddr =
InetAddress.getByName(data[1]);
                        } catch (UnknownHostException e) {
                            Constants.perrorGUI(
                                    "IPC Channel",
                                    "New conference request: " +
e.toString());
                            return;
                        }
                        int mport = new Integer(data[2]);
                        int channel = new Integer(data[3]);
                        new TextConference(
                                getIpaddr(),
                                maddr,
                                mport,
                                nickname,
                                channel,
                                listener,
                                inviter).setVisible(true);
                    } else if
(dataStr.startsWith(Constants.GSTVOICE)) {
                        Constants.printf("v> received voice chat
state request\n");
                        if (state) {
                            replyStr = Constants.ON_VOICE;
                        } else {
                            replyStr = Constants.OFFVOICE;
                        }
                        buf = replyStr.getBytes();
                        ipcpack.setData(buf);
                        ipcsock.send(ipcpack);
                    } else if
(dataStr.startsWith(Constants.SSTVOICE)) {
                        System.out.println("v> new voice chat
initiation requested");
                        if (!state) {
                            state = true;
                            buf[0] = 0;
                            addrStr =
dataStr.substring(Constants.SSTVOICE.length());
                            InetAddress addr =
InetAddress.getByName(addrStr);
                            ChatBox cb = find(addr);
                            System.out.println(cb == null ? "cb IS
null\n" : "cb NOT null\n");
                            if (cb == null) {
                                state = false;
                                buf[0] = 1;
                                ipcpack.setData(buf);
                                ipcsock.send(ipcpack);
```

```
                                continue;
                            }
                            ipcpack.setData(buf);
                            ipcsock.send(ipcpack);
                            vc = new VoiceChat(addr,
Constants.PORT_VOICE, 1);
                            cb.setVC(vc);
                        } else {
                            System.out.println("v> you are already
in a voice chat");
                            buf[0] = 1;
                            ipcpack.setData(buf);
                            ipcsock.send(ipcpack);
                        }
                    }
                } catch (IOException ex) {
                    Constants.perrorGUI("IPC channel",
ex.toString());
                } finally {
                    ipcpack = null;
                }
            }
        }
    }//end inner class IPCChannel

    private class ChatHandler {
        ServerSocket tchatsock;
        Socket newsock;

        ChatHandler() {
            try {
                tchatsock = new ServerSocket(Constants.PORT_TEXT);
                tchatsock.setReuseAddress(true);
                Constants.printf("text chat: listening on " +
tchatsock.getLocalPort() + "\n");
            } catch (IOException ex) {
                Constants.perror("text chat: unable to create
socket\n\t" + ex.toString());
            }
        }

        private void newConn() {
            try {
                newsock = tchatsock.accept();
                BufferedReader in = new BufferedReader(new
InputStreamReader(
                        newsock.getInputStream()));
                String temp = in.readLine();
                Constants.printf("remote user: " + temp + "\n");
                final ChatBox cb = new ChatBox(nickname, temp,
newsock);
                cb.addWindowListener(new WindowAdapter() {
                    @Override
                    public void windowClosed(WindowEvent evt) {
                        chatList.remove(cb);
                    }
```

```
                });
                addCB(cb);
            } catch (IOException ex) {
                Constants.perror("text chat: error creating new
connection\n\t" +
                        ex.toString());
            }
        }
    }//end inner class ChatHandler

    private class DialogNickEntry extends javax.swing.JDialog {
        private javax.swing.JButton btnCancel;
        private javax.swing.JButton btnOk;
        private javax.swing.JLabel labelEntry;
        private javax.swing.JPanel panelButtons;
        private javax.swing.JPanel panelEntry;
        private javax.swing.JTextField tfNick;

        public DialogNickEntry(java.awt.Frame parent, boolean
modal) {
            super(parent, modal);
            initComponents();
        }

        // <editor-fold defaultstate="collapsed" desc=" Generated
Code ">
        private void initComponents() {
            java.awt.GridBagConstraints gridBagConstraints;
            panelEntry = new javax.swing.JPanel();
            labelEntry = new javax.swing.JLabel();
            tfNick = new javax.swing.JTextField();
            panelButtons = new javax.swing.JPanel();
            btnOk = new javax.swing.JButton();
            btnCancel = new javax.swing.JButton();
            getContentPane().setLayout(new
java.awt.GridBagLayout());
setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON
_CLOSE);
            setTitle("Nickname");
            setResizable(false);
            panelEntry.setLayout(new java.awt.GridLayout(2, 1, 0,
10));
            labelEntry.setText("Enter your nickname");
            panelEntry.add(labelEntry);
            tfNick.setToolTipText("Enter a nickname to identify
yourself");
            tfNick.addActionListener(new
java.awt.event.ActionListener() {
                @Override
                public void
actionPerformed(java.awt.event.ActionEvent evt) {
                    tfNickActionPerformed(evt);
                }
            });
            panelEntry.add(tfNick);
```

```
            gridBagConstraints = new
java.awt.GridBagConstraints();
            gridBagConstraints.fill =
java.awt.GridBagConstraints.BOTH;
            gridBagConstraints.ipadx = 100;
            gridBagConstraints.ipady = 15;
            getContentPane().add(panelEntry, gridBagConstraints);
            panelButtons.setLayout(new java.awt.GridLayout());
            btnOk.setText("OK");
btnOk.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1,
1, 1));
            btnOk.setMinimumSize(new java.awt.Dimension(80, 25));
            btnOk.setPreferredSize(new java.awt.Dimension(80,
25));
            btnOk.addActionListener(new
java.awt.event.ActionListener() {
                @Override
                public void
actionPerformed(java.awt.event.ActionEvent evt) {
                    btnOkActionPerformed(evt);
                }
            });
            panelButtons.add(btnOk);
            btnCancel.setText("Cancel");
            btnCancel.setToolTipText("Cancelling will terminate
the process");
btnCancel.setBorder(javax.swing.BorderFactory.createEmptyBorder(1,
1, 1, 1));
            btnCancel.setMinimumSize(new java.awt.Dimension(80,
25));
            btnCancel.setPreferredSize(new java.awt.Dimension(80,
25));
            btnCancel.addActionListener(new
java.awt.event.ActionListener() {
                @Override
                public void
actionPerformed(java.awt.event.ActionEvent evt) {
                    btnCancelActionPerformed(evt);
                }
            });
            panelButtons.add(btnCancel);
            gridBagConstraints = new
java.awt.GridBagConstraints();
            gridBagConstraints.gridx = 0;
            gridBagConstraints.gridy = 1;
            gridBagConstraints.fill =
java.awt.GridBagConstraints.HORIZONTAL;
            gridBagConstraints.ipadx = 100;
            gridBagConstraints.insets = new java.awt.Insets(10, 0,
10, 0);
            getContentPane().add(panelButtons,
gridBagConstraints);
            java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
            setBounds((screenSize.width - 258) / 2,
(screenSize.height - 177) / 2, 258, 177);
```

```
        }// </editor-fold>

        private void
tfNickActionPerformed(java.awt.event.ActionEvent evt) {
            checkInput();
        }//end method

        private void
btnOkActionPerformed(java.awt.event.ActionEvent evt) {
            checkInput();
        }//end method

        private void
btnCancelActionPerformed(java.awt.event.ActionEvent evt) {
            dispose();
        }//end method

        /**
         * check nickname input
         */
        private void checkInput() {
            String temp;
            if (!(temp = tfNick.getText().trim()).isEmpty()) {
                if (temp.length() < Constants.MIN_NICKNAME ||
temp.length() > Constants.MAX_NICKNAME) {
                    JOptionPane.showMessageDialog(this,
                            "Nickname must not less than " +
Constants.MIN_NICKNAME + " characters\nnor more than " +
Constants.MAX_NICKNAME + " characters",
                            "Nickname Error",
                            JOptionPane.WARNING_MESSAGE);
                } else {
                    nickname = temp;
                    lblNickname.setText(HAS_NAME + nickname);
                    dispose();
                }
            } else {
                JOptionPane.showMessageDialog(this, "Please enter
your nickname", "Nickname Error", JOptionPane.WARNING_MESSAGE);
            }
        }//end method
    }//end class

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnChIP, btnChNick, btnConf,
btnInfo, btnJoinRoom, btnMakeRoom, btnTextChat, btnVoiceChat;
    private javax.swing.JLabel lblIP, lblNickList, lblNickname,
lblStatus;
    private javax.swing.JList listNickname;
    private javax.swing.JPanel panelAfter, panelBefore,
panelButtons, panelIPInput, panelList, panelListHolder, panelMain,
panelStatus, panelUpper, panelUpperMenu;
    private javax.swing.JScrollPane scrpaneMain;
    private javax.swing.JTextField txtIP;
    // End of variables declaration//GEN-END:variables
}
```

CURRICULUM VITAE

Nama	:	Mark Prima Estafeta Muchammad
Tempat Tanggal Lahir	:	Temanggung, 29 September 1987
Nama Bapak / Pekerjaan	:	Makmun Pitoyo/ PNS
Nama Ibu	:	Mardiastuti
Alamat Rumah	:	RT 02 RW 01 Giyanti Temanggung Jawa Tengah
Alamat di Yogya	:	Gang Nanas IV/4 Kopenrejo RT 03 RW 15 Nanggulan Maguwoharjo Sleman Yogyakarta
No HP	:	08995138256
Email	:	mark.p.e.muhammad@gmail.com

Riwayat Pendidikan

1993-1999	:	SD Negeri Temanggung II No. 2
1999-2002	:	SMP Negeri 2 Temanggung
2002-2005	:	SMA Negeri 1 Temanggung
2005-2011	:	Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta