ANALISIS, PERANCANGAN DAN IMPLEMENTASI APLIKASI CHATTING BERBASIS OBJEK

SKRIPSI

Untuk Memenuhi Sebagian Persyaratan
Mencapai derajat Sarjana S-1
Program Studi Teknik Informatika

Disusun oleh :

LU'LU'UN NISA' KURNIA PUTRI
NIM.05650017

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UIN 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.01.1/770/2011

Skripsi/Tugas Akhir dengan judul	:	Analisis, Perancangan dan Implementasi Aplikasi Chatting Berbasis Objek
Yang dipersiapkan dan disusun oleh	:	
Nama	:	Lu'lu'un Nisa' Kurnia Putri
NIM	:	05650017
Telah dimunaqasyahkan pada	:	15 April 2011
Nilai Munaqasyah	:	A / B

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

TIM MUNAQASYAH :

Ketua Sidang

Sumarsono, M. Kom
NIP. 19710209 200501 1 003

Penguji I

Shofwatul 'Uyun, M.Kom
NIP.19820511 200604 2 002

Penguji II

Agung Fatwanto, Ph.D
NIP. 19770103 200501 1 003

Yogyakarta, 27 April 2011
UIN Sunan Kalijaga
Fakultas Sains dan Teknologi
Dekan

KEMENTERIAN AGAMA UIN SUNAN KALIJAGA FAKULTAS SAINS DAN TEKNOLOGI YOGYAKARTA

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

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 Saudari:

Nama : Lu'lu'un Nisa' Kurnia Putri
NIM : 05650017
Judul Skripsi : **Analisis, Perancangan dan Implementasi Aplikasi *Chatting* Berbasis Objek**

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, 31 Maret 2011

Pembimbing I	Pembimbing II
Sumarsono.S.T.,M.Kom	Maria Ulfah S.S.Kom.,M.I.T
NIP. 19710209-200501-1-003	NIP. 19780106-200212-2-001

PERNYATAAN KEASLIAN SKRIPSI

Yang bertanda tangan di bawah ini:

Nama : Lu'lu'un Nisa' Kurnia Putri

NIM : 05650017

Program Studi : Teknik Informatika

Fakultas : Sains dan Teknologi

Menyatakan bahwa skripsi dengan judul "**Analisis, Perancangan dan Implementasi Aplikasi *Chatting* Berbasis Objek"** tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu 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, 4 April 2011

Yang Menyatakan

Lu'lu'un Nisa'Kurnia Putri

NIM. 05650017

MOTTO

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

إِنَّ مَعَ ٱلۡعُسۡرِ يُسۡرًا ۝٦

Sesungguhnya Sesudah Kesulitan Itu Ada Kemudahan.

(Q.S. Al insyiroh:6)

HALAMAN PERSEMBAHAN

Ananda persembahkan karya ini kepada:

Kedua Orangtua dan Keluarga

Almamater Teknik Informatika UIN Sunan Kalijaga

Semua Teman-Teman yang memberi semangat dan dukungan

Terima kasih untuk semuanya…

KATA PENGANTAR

Puji syukur selalu dipanjatkan kehadirat Allah SWT., karena kebesaran dan keagungan serta kasih saying-Nya, begitu banyak anugrah ilmu, rezeki dan cinta yang berlimpah kepada seluruh alam semesta, sehingga semua tak luput dari pengawasan-Nya.

Dalam penyusunan skripsi ini tidak mungkin dapat terselesaikan tanpa bantuan, dorongan serta saran dan kritik dari berbagai pihak. Oleh karena itu, penyusun mengucapkan terimakasih kepada:

1. Prof. Drs. H. Akh. Minhaji, M. A, Ph.D, selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
2. Ketua Program Studi Teknik Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Bapak Agus Mulyanto, S.Si, M.Kom selaku Pembimbing Akademik.
4. Bapak Sumarsono, S.T, M.Kom selaku pembimbing I dan Ibu Maria Ulfah Siregar, S.Kom, M.IT selaku dosen pembimbing II skripsi yang dengan kesabarannya telah memberikan bimbingan dan arahan serta motivasi kepada penulis sehingga dapat menyelesaikan skripsi ini.
5. Segenap Dosen dan Karyawan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
6. Dosen Program Studi Teknik Informatika yang banyak memberikan masukan ilmu kepada penulis.

7. Ayahanda M.Subandi dan Ibunda Suratmiatun tercinta, yang dengan kasih sayang dan cinta kasih yang tulus serta do'a yang selalu dipanjatkan untuk penulis sehingga dapat memberikan motivasi dan semangat untuk terus berkarya dan menggapai cita-cita.
8. Kakak-kakakku tersayang terima kasih atas do'a dan semangat yang diberikan.
9. Teman-teman seperjuangan T.Informatika angkatan 2005/2006 dan teman-teman khodyjah crew terima kasih atas bantuan dan semangat yang diberikan.
10. Semua pihak yang tidak dapat penulis sebutkan satu persatu di sini, terima kasih atas bantuan dan semangatnya sehingga penulis dapat menyelesaikan skripsi ini.

Semoga Allah SWT., memberikan balasan atas kebaikan mereka yang tak ternilai. Dengan segala kekurangan dan kehilafan dalam penulisan, penulis berharap masukan dan koreksi dari pembaca dan semoga skripsi ini bermanfaat bagi semua pihak. Atas segala khilaf penulis mohon maaf yang sedalam-dalamnya.

Yogyakarta,31 Maret 2011

Penulis

Lu'lu'un Nisa' Kurnia Putri

NIM. 05650017

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ANALISIS, PERANCANGAN DAN IMPLEMENTASI APLIKASI CHATTING BERBASIS OBJEK

Disusun oleh:

Lu'lu'un Nisa' Kurnia Putri

INTISARI

Komunikasi adalah bagian yang tidak dapat dipisahkan dengan kehidupan manusia. Bentuk dan media untuk berkomunikasi juga mengalami perubahan seiring dengan berkembangnya teknologi. Bentuk komunikasi dimulai dengan komunikasi bertatap muka dan melalui surat, kemudian setelah ditemukannya jaringan komputer dan koneksi internet, komunikasi semakin berkembang sehingga dapat dilakukan di mana saja dan kapan saja tanpa harus bertemu langsung pada satu tempat.

Penelitian ini mengembangkan sebuah sistem komunikasi teks antarkomputer melalui protokol internet. Sistem komunikasi ini dikhususkan untuk digunakan pada jaringan lokal atau LAN dan ditulis menggunakan bahasa pemrograman Java (J2SE) dengan memanfaatkan teknologi *multicast* (*MulticastSocket*) yang telah didukung dalam pemrograman Java. Metode penelitian yang dipakai adalah metode RAD (*Rapid Application Development*). Metode RAD adalah metode pengembangan sistem linier sekuensial yang menekankan siklus perkembangan yang sangat pendek dengan menggunakan pendekatan konstruksi berbasis komponen yang meliputi pemodelan bisnis, pemodelan data, pemodelan proses, pembuatan aplikasi dan pengujian.

Berdasarkan pengujian, aplikasi *chatting* ini dapat berjalan dengan baik. Komunikasi teks yang didukung adalah komunikasi *public*, komunikasi *private* dan komunikasi grup. Aplikasi *chatting* ini menggunakan koneksi *peer to peer* yang artinya masing-masing komputer berada pada tingkat yang sama. Aplikasi *chatting* ini dapat dikembangkan lebih lanjut dengan menambah fitur-fitur lain seperti proses *login*, komunikasi suara, video atau transfer file.

Kata kunci: *Instant Messaging* (IM), Java, *multicast*

ANALYSIS, DESIGN AND IMPLEMENTATION OF OBJECT-BASED CHATTING APPLICATION

Written by:

Lu'lu'un Nisa' Kurnia Putri

ABSTRACT

Communication is an inseparable part of human life. Along with the development of technology, forms and media of communication are also changing. Communication begins with face to face and by mail communication. After the invention of computer network and internet connections, communication develops even further in such it can be done anywhere and anytime without having to meet in person, and at the same place.

This research developed an inter-computer text communication system via the internet protocol. This communication system was mainly developed to be used in local network or LAN and was written in Java programming language (J2SE) using multicast technology (*Multicast Socket)* that has been supported in Java. The research method used in this research was RAD (Rapid Application Development), a linear sequential system development method which emphasizes on short development cycle by using component-based construction approach including business modeling, data modeling, process modeling, application creation and testing.

According to the result of the testing phase, this chat application can run well. The supported kinds of text communication are public communication, private communication and group communication. This chat application uses peer to peer connection which means that each computer is on the same level. This chat application can further be developed by adding other features such as login process, voice communication, video communication or file transfer.

Key words: *Instant Messaging* (IM), Java, *Multicast*

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Komunikasi adalah bagian yang tidak dapat dipisahkan dengan kehidupan manusia. Seiring dengan berkembangnya teknologi, bentuk dan media untuk berkomunikasi juga mengalami perubahan. Bentuk komunikasi dimulai dengan komunikasi bertatap muka dan melalui surat. Saat ini setelah ditemukannya jaringan komputer dan koneksi internet, dengan dukungan aplikasi *instant messaging* (IM) seperti *yahoo messenger* (YM), MSN *messenger* dan *google talk* komunikasi bisa dilakukan dimana saja dan kapan saja tanpa harus bertemu langsung pada satu tempat.

Instant messaging atau biasa disebut aktivitas *chatting* adalah komunikasi dua arah antara satu orang atau beberapa orang baik dengan teks, suara maupun video. Dan dengan memanfaatkan aplikasi *instant messaging* ini akan menghemat waktu, tenaga dan biaya karena tidak perlu lagi melakukan perjalanan yang melelahkan untuk menjalin komunikasi. Misalnya, pada perusahaan yang mempunyai gedung yang besar dan tinggi atau lokasi antar divisi yang jauh sehingga banyak dijumpai orang yang kesana kemari hanya untuk menyampaikan sesuatu yang akan membuang waktu dengan sia-sia.

Dengan alasan tersebut penulis memutuskan untuk membuat Aplikasi chatting yang akan dibangun pada jaringan LAN dengan menggunakan *Java Standart Edition* (J2SE). Dimana aplikasi ini dapat melakukan pengiriman pesan

teks dan dengan memanfaatkan IP multicast untuk pengirimannya aplikasi ini dapat melakukan *chatting group* atau forum diskusi yang akan diatur oleh moderator pada masing-masing grup.

1.2 RUMUSAN MASALAH

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

1. Bagaimana merancang aplikasi *chatting* yang mampu meningkatkan kemudahan dalam berkomunikasi.
2. Bagaimana mengimplementasikan rancangan aplikasi *chatting* tersebut dengan bahasa pemrograman berorientasi objek Java.

1.3 BATASAN MASALAH

Penelitian ini akan membatasi cakupan permasalahan yang terkait dengan analisis perancangan dan implementasi aplikasi *chatting* berbasis objek. Masalah dalam penelitian ini dibatasi dalam hal :

1. Perancangan pada aplikasi *chatting* yang akan dibangun menggunakan UML.
2. Aplikasi *chatting* yang dikembangkan memanfaatkan transmisi jaringan *multicast* pada jaringan LAN.
3. Aplikasi *chatting* yang dikembangkan memanfaatkan sistem koneksi *peer to peer*.
4. Aplikasi *Chatting* yang dikembangkan berbasis komunikasi teks.

1.4 TUJUAN PENELITIAN

Penelitian ini bertujuan untuk merancang dan mengimplementasikan aplikasi *chatting* yang mampu meningkatkan kemudahan dalam berkomunikasi menggunakan bahasa pemrograman berorientasi objek Java.

1.5 MANFAAT PENELITIAN

Hasil penelitian ini diharapkan dapat dimanfaatkan sebagai bahan pembelajaran bagi pemula dan dapat membantu dalam mempermudah komunikasi.

1.6 KEASLIAN PENELITIAN

Penelitian yang berhubungan dengan aplikasi *chatting* sudah banyak dilakukan. Perbedaan penelitian yang dilakukan dengan penelitian sebelumnya terletak pada perancangan dan teknologi yang digunakan. Perancangan dalam aplikasi chatting ini menggunakan UML dan teknologi yang digunakan adalah teknologi *multicast* untuk transmisi jaringan dan menggunakan sistem koneksi *peer to peer*.

BAB V
KESIMPULAN

5.1. Kesimpulan

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

1. Aplikasi *chatting* telah berhasil dibuat dengan kemampuan *chat public*, *chat group* dan *chat private*.
2. Aplikasi *chatting* yang dibuat hanya dapat digunakan pada jaringan LAN untuk membantu dalam mempermudah komunikasi antar pengguna dalam satu jaringan LAN menggunakan komunikasi teks.
3. Menurut hasil pengujian beta aplikasi *chatting* ini mudah digunakan karena menggunakan menu yang mudah dipelajari.

5.2. Saran

Aplikasi *chatting* ini tidak terlepas dari kekurangan dan kelemahan. Oleh karena itu, penulis memberikan beberapa saran yang dapat digunakan sebagai acuan dalam penelitian atau pengembangan selanjutnya, yaitu :

1. Aplikasi *chatting* dalam penelitian hanya dapat mengirimkan pesan teks, ke depan bisa dikembangkan menjadi suatu aplikasi yang lebih baik dengan menambah proses *login* dan kemampuan pengiriman suara, gambar, video atau transfer file.

2. Aplikasi *chatting* ini hanya dapat berjalan pada jaringan LAN, ke depan dapat dikembangkan dengan memanfaatkan teknologi LAN maupun internet.

Akhirnya dengan segala keterbatasan aplikasi yang dibuat penulis ini, penulis tetap berharap bahwa aplikasi ini akan memberikan gagasan baru dan semangat untuk mengembangkan lebih lanjut. Selain itu semoga aplikasi ini dapat digunakan sebagai pemanfaatan teknologi dalam berkomunikasi.

DAFTAR PUSTAKA

Arifin. 2004. "Implementasi Dan Analisis Jaringan Berbasis Multicast Pada Aplikasi Video Conferenci". Skripsi. Universitas Gadjah Mada, Yogyakarta.

Dharwiyanti, Sri dan Wahono, Romi Satria. 2003. "Pengantar Unified Modeling Language (UML)". IlmuKomputer.com. Akses tanggal 04 Juni 2009.

Fowler, Martin. 2005. "UML Distilled Edisi 3 Panduan Singkat Bahasa Pemodelan Objek Standar". Penerbit Andi. Yogyakarta.

Hariyanto, Bambang. 2007. "Esensi-Esensi Bahasa Pemrograman Java edisi 2". Penerbit Informatika. Bandung.

Hartati, G.Sri. 2007. "Pemrograman GUI Swing Java dengan NetBeans 5". Penerbit Andi.Yogyakarta.

Haryadi, Mohamad Fauzi. 2010. "Analisa dan Perancangan Aplikasi Chatting Berbasis Web Menggunakan Flash CS3". Naskah Publikasi. Amikom.Yogyakarta.

Irawan, Ivan. 2003. "Pemrograman Socket dengan C". http://ilmukomputer.com. Akses tanggal 28 Juli 2010.

Isjaya. 2010. "Internet Protocol Versi 6 (IPV6)". Universitas Hasanuddin. Makassar.

Naughton, Patrick. 1997. "Java Handbook/ Konsep Dasar Pemrograman Java". Penerbit Andi. Yogyakarta.

Odom, Wendell. 2004. "CCNA INTRO Exam Certification". Penerbit Indianapolis. Cisco Press.

Pressman, Roger.S. 2002. "Rekayasa Perangkat Lunak Pendekatan Praktisi (Buku Satu)". Penerbit Andi. Yogyakarta.

Primawan, A.Bayu, Wardhana, S. Raditya Wisnu dan Widjaja, Damar. 2008. "Aplikasi Chatting antar Komputer Menggunakan Bluetooth". Jurnal Konferensi Nasional Sistem dan Informatika. Universitas Sanata Dharma. Yogyakarta.

Riansyah. 2005. "Analisis Aplikasi Telekonferensi Memakai Ip Multicast Pada Jaringan Lan". Skripsi. Universitas Gadjah Mada. Yogyakarta.

Riawan, Helmi Fajar. 2010. "Perancangan Program Instant Messengger untuk Komunikasi Administrator dan Pelanggan di Solo Movie Surakarta". Skripsi. Universitas Sebelas Maret. Surakarta.

Setiawan, Romi dan Sutanta, Edhy. 2009. "Membangun Aplikasi Chatting Berbasis Multiuser". Jurnal DASI Vol. 10 no. 1. IST AKPRIND. Yogyakarta.

Siagian, A. Frans. 2007. "Perancangan Komunikasi Client Server dan Sistem Database". Skripsi. Universitas Sumatera Utara. Medan.

Suarga. 2009. "Dasar Pemrograman Komputer dalam Bahasa Java". Penerbit Andi. Yogyakarta.

Syafrizal, Melwin. 2005. "Pengantar Jaringan Komputer". Penerbit Andi. Yogyakarta.

Wahono, R. M. 2003. "Pengantar Unified Modeling Language". http://ilmukomputer.com. Akses tanggal 02 Mei 2007.

Yunita, Meryani. 2009. "Perancangan dan Pembuatan Aplikasi Chatting". Skripsi. Universitas Kristen Petra. Surabaya.

Lampiran-lampiran

KODE PROGRAM

Kode Program yang digunakan dalam aplikasi ini adalah sebagai berikut:

A. MainWindow.java

```
public class MainWindow extends JFrame
        implements WindowListener, ActionListener {

public MainWindow() {
        super("Chatting Application");
        thisUser = null;
        userPanel = null;
        chatPanel = null;
        createGroupDlg = null;
        joinGroupDlg = null;
        leaveGroupDlg = null;
        setDefaultCloseOperation(0);
        LinkManager.initLinkManager(this);
        if (!LinkManager.isFunctioning()) {
            JOptionPane.showMessageDialog(this, "Could not listen
on port: 49252. \nApplication already running or \nsome problem
occured on port.", "Application Cannot Run", 0);
            return;
        }
        thisUser = LinkManager.getThisUser();
        String s = conf.getLastUserName();
        if (s != null) {
            thisUser.rename(s);
        }
        userPanel = new UserPanel(AppUser.userVector, this);
        chatPanel = new ChatPanel();
        Container container = getContentPane();
        JSplitPane jsplitpane = new JSplitPane(1, userPanel,
chatPanel);
        jsplitpane.setOneTouchExpandable(true);
        jsplitpane.setDividerLocation(180);
        jsplitpane.setResizeWeight(0.0D);
        jsplitpane.setContinuousLayout(true);
        container.add(jsplitpane);
        buildMenus();
        enabledAllButtons(false);
        pack();
        addWindowListener(this);
        setSize(new Dimension(683, 395));
        setIconImage(iconframe.getImage());
        setVisible(true);
        addNewUser(thisUser);
        createGroupDlg = new CreateGroupDialog(this);
        joinGroupDlg = new JoinGroupDialog(this,
ConfGroup.groupVector);
        leaveGroupDlg = new LeaveGroupDialog(this,
thisUser.getGroupVector());
        LinkManager.startLinkManager();
```

```
    }
    public void addNewUser(AppUser appuser) {
        chatPanel.receiveInfoMessage((new
StringBuilder()).append(appuser.getUserName()).append("
(").append(appuser.getIPAddress()).append(") is
online").toString());
        userPanel.refreshList();
    }

    public void removeUser(AppUser appuser) {
        chatPanel.receiveInfoMessage((new
StringBuilder()).append(appuser.getUserName()).append("
(").append(appuser.getIPAddress()).append(") is
offline").toString());
        Vector vector = appuser.getGroupVector();
        for (int i = 0; i < vector.size(); i++) {
            ConfGroup confgroup = (ConfGroup) vector.elementAt(i);
            if (appuser.isModeratorOf(confgroup)) {
                endConfGroup(confgroup);
            } else {
                appuser.leaveGroup(confgroup);
                leaveConfGroup(appuser, confgroup);
            }
        }
        appuser.setInactive();
        GroupChatWindow.refreshListAll();
        userPanel.refreshList();
        refreshDisplay();
    }

    public void addNewGroup(ConfGroup confgroup) {
        chatPanel.receiveInfoMessage((new
StringBuilder()).append("New group is created :
").append(confgroup.getGroupName()).append(" by
").append(confgroup.getModeratorName()).toString());
        joinGroupDlg.refreshList();
        refreshDisplay();
    }

    public void endConfGroup(ConfGroup confgroup) {
        chatPanel.receiveInfoMessage((new
StringBuilder()).append("Group
").append(confgroup.getGroupName()).append(" has
ended").toString());
        GroupChatWindow.closeGroupChatWindow(confgroup);
        confgroup.endGroup();
        leaveGroupDlg.refreshList();
        joinGroupDlg.refreshList();
        GroupChatWindow.refreshDisplayAll();
        refreshDisplay();
    }

    public void joinConfGroup(AppUser appuser, ConfGroup
confgroup) {
        GroupChatWindow groupchatwindow =
GroupChatWindow.getWindowForThisGroup(confgroup);
```

```
        if (groupchatwindow != null) {
            groupchatwindow.refreshList();
        }
        refreshDisplay();
    }
    public void leaveConfGroup(AppUser appuser, ConfGroup
confgroup) {
        GroupChatWindow groupchatwindow =
GroupChatWindow.getWindowForThisGroup(confgroup);
        if (groupchatwindow != null) {
            groupchatwindow.refreshList();
        }
        refreshDisplay();
    }

    public void userJoinConfGroup(ConfGroup confgroup) {
        if (thisUser.isJoining(confgroup)) {
            return;
        } else {
            confgroup.addMember(thisUser);
            LinkManager.sendJoinInfo(confgroup);
            new GroupChatWindow(confgroup, this);
            leaveGroupDlg.refreshList();
            joinGroupDlg.refreshList();
            GroupChatWindow.refreshDisplayAll();
            refreshDisplay();
            return;
        }
    }

    public void userLeaveConfGroup(ConfGroup confgroup) {
        if (confgroup.getModerator() == thisUser) {
            LinkManager.sendEndInfo(confgroup);
            endConfGroup(confgroup);
        } else {
            LinkManager.sendLeaveInfo(confgroup);
            confgroup.removeMember(thisUser);
        }
        GroupChatWindow.closeGroupChatWindow(confgroup);
        leaveGroupDlg.refreshList();
        joinGroupDlg.refreshList();
        GroupChatWindow.refreshDisplayAll();
        refreshDisplay();
    }

    public void userCreateNewGroup(ConfGroup confgroup) {
        chatPanel.receiveInfoMessage((new
StringBuilder()).append("New group is created :
").append(confgroup.getGroupName()).append(" by
").append(confgroup.getModeratorName()).toString());
        LinkManager.sendNewGroupMessage(confgroup);
        new GroupChatWindow(confgroup, this);
        leaveGroupDlg.refreshList();
        GroupChatWindow.refreshDisplayAll();
        refreshDisplay();
    }
```

```
    public void requestJoinGroup(ConfGroup confgroup, String s) {
        LinkManager.sendJoinRequest(confgroup, s);
    }

    public void requestJoinGroup(ConfGroup confgroup) {
        LinkManager.sendJoinRequest(confgroup);
    }

    public void receivePublicChatMessage(String s) {
        chatPanel.receiveChatMessage(s);
    }

    public void receiveGroupChatMessage(ConfGroup confgroup,
String s) {
        GroupChatWindow groupchatwindow =
GroupChatWindow.getWindowForThisGroup(confgroup);
        if (groupchatwindow != null) {
            groupchatwindow.receiveChatMessage(s);
        }
    }

    public void receivePrivateChatMessage(AppUser appuser, String
s) {
        PrivateChatWindow privatechatwindow =
PrivateChatWindow.getWindowForThisUser(appuser);
        if (privatechatwindow != null) {
            privatechatwindow.receiveChatMessage(s);
        } else {
            PrivateChatWindow privatechatwindow1 = new
PrivateChatWindow(appuser, this);
            privatechatwindow1.receiveChatMessage(s);
        }
    }

    public void receiveRequestReply(ConfGroup confgroup, int i) {
        joinGroupDlg.receiveRequestReply(confgroup, i);
    }

    public void receiveKickMessage(ConfGroup confgroup) {
        userLeaveConfGroup(confgroup);
        JOptionPane.showMessageDialog(this, "Sorry, you're kicked
by the moderator", (new StringBuilder()).append("Kicked from :
").append(confgroup.getGroupName()).toString(), 1);
    }

    public void receiveInviteMessage(ConfGroup confgroup) {
        int i = JOptionPane.showConfirmDialog(this, (new
StringBuilder()).append("You're invited to join
").append(confgroup.getGroupName()).append(" by
").append(confgroup.getModerator().getUserName()).append("\n Would
you like to join?").toString(), "Invitation to join group", 0);
        if (i == 0) {
            userJoinConfGroup(confgroup);
        }
    }
```

```
 public void receiveRenameMessage(AppUser appuser, String s) {
        chatPanel.receiveInfoMessage((new
StringBuilder()).append(appuser.getUserName()).append(" changed
his/her name to ").append(s).toString());
        appuser.rename(s);
        userPanel.refreshList();
        GroupChatWindow.refreshListAll();
    }

    public void leaveAllGroups() {
        ConfGroup confgroup;
        for (Vector vector = thisUser.getGroupVector();
vector.size() > 0; userLeaveConfGroup(confgroup)) {
            confgroup = (ConfGroup) vector.elementAt(0);
        }
    }

    private void exitApplication() {
        int i = JOptionPane.showConfirmDialog(this, "Are you sure
you want to exit?", "Confirm Exit", 0);
        if (i == 0) {
            leaveAllGroups();
            System.exit(0);
        }
    }

    public void actionPerformed(ActionEvent actionevent) {
        String s = actionevent.getActionCommand();
        if (s.equals("Rename")) {
            String s1 = JOptionPane.showInputDialog(this, "Enter
your new name to be displayed", "Rename", 3);
            if (s1 == null) {
                return;
            }
            s1 = s1.trim();
            if (s1.equals("")) {
                JOptionPane.showMessageDialog(this, "Your name
should contain at least one non-spacing character", "Cannot
Rename", 1);
            } else {
                LinkManager.sendRenameMessage(s1);
                receiveRenameMessage(thisUser, s1);
                conf.setLastUserName(s1);
                conf.save();
            }
        }  else if (s.equals("Exit")) {
            exitApplication();
        } else if (s.equals("Create Group")) {
            showCreateGroupDialog();
        } else if (s.equals("Join Group")) {
            showJoinGroupDialog();
        } else if (s.equals("Leave Group")) {
            showLeaveGroupDialog();
        } else if (s.equals("Leave All Groups")) {
            leaveAllGroups();
```

```
        } else if (s.equals("About Program")) {
            JOptionPane.showMessageDialog(this, "Video Chat
Application v 1.0\nCopyright (c) 2011 Putri, all rights
reserved.", "Information",JOptionPane.INFORMATION_MESSAGE);
        }
    }

    public void windowActivated(WindowEvent windowevent) {
        chatPanel.requestFocus();
    }

    public void windowClosed(WindowEvent windowevent) {
    }

    public void windowClosing(WindowEvent windowevent) {
        exitApplication();
    }

    public void windowOpened(WindowEvent windowevent) {
        chatPanel.requestFocus();
    }

    public static Configuration getConfiguration() {
        return conf;
    }

    public static void main(String args[]) {
        MainWindow mainwindow = new MainWindow();
    }
}
```

B. LinkManager.java

```
public class LinkManager {
    static class TimeoutThread extends Thread {

        private int timeout;
        private int state;

        @Override
        public void run() {
            boolean loop = false;
            do {
                try {
                    sleep(timeout);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                if (state == 1) {
                    loop = LinkManager.IntroduceTimeout();
                } else if (state == 2) {
                    loop = LinkManager.AskGroupTimeout();
                }
            } while (loop);
```

```
        }

        public TimeoutThread(int state, int timeout) {
            this.timeout = timeout;
            this.state = state;
        }
    }
    public LinkManager() {}
    public static void initLinkManager(MainWindow main) {
        linkServer = new LinkSocketServer(49252);
        if (!linkServer.isFunctioning()) {
            return;
        } else {
            publicMcReceiver = new
MulticastReceiver("239.167.0.1", 49250);
            publicMcSender = new MulticastSender("239.167.0.1",
49250);
            mainWindow = main;
            userVector = AppUser.userVector;
            groupVector = ConfGroup.groupVector;
            queueVector = new Vector();
            thisUser = createThisUser();
            return;
        }
    }

    public static boolean isFunctioning() {
        if (linkServer == null) {
            return false;
        } else {
            return linkServer.isFunctioning();
        }
    }

    public static void startLinkManager() {
        publicMcReceiver.start();
        linkServer.start();
        sendIntroduceMessage();
    }

    public static void seeQueue() {
        System.out.println("Queue List: ");
        for (int i = 0; i < queueVector.size(); i++) {
            AppUser user = (AppUser) queueVector.elementAt(i);
            System.out.print((new
StringBuilder(String.valueOf(user.getUserName()))).append("
").toString());
        }
    }

    public static void delayThread(int timeout) {
        try {
            Thread.currentThread();
            Thread.sleep(timeout);
        } catch (InterruptedException e) {
            e.printStackTrace();
```

```
        }
    }

    public static AppUser getThisUser() {
        return thisUser;
    }

    public static AppUser createThisUser() {
        try {
            InetAddress addr = InetAddress.getLocalHost();
            AppUser user = new AppUser(addr.getHostAddress(),
addr.getHostName(), null);
            return user;
        } catch (UnknownHostException e) {
            System.out.println("Error when get User Address");
        }
        return null;
    }

    public static void removeUser(AppUser user) {
        if (user == null || userVector.indexOf(user) == -1) {
            return;
        }
        if (queueVector.indexOf(user) != -1) {
            queueVector.remove(user);
        }
        mainWindow.removeUser(user);
    }

    private static String[] getMessageElement(String msg) {
        String arr[] = msg.split("\t");
        for (int i = 0; i < arr.length; i++) {
            arr[i] = arr[i].trim();
        }
        return arr;
    }

    public static void receivePublicMulticastMessage(String
message) {
        if (message == null) {
            return;
        }
        System.out.println((new StringBuilder("Receive Public
Multicast Message : \n")).append(message).toString());
        int lastidx = 0;
        String arr[] = (String[]) null;
        int idx = message.indexOf("\t", lastidx + 1);
        String header = message.substring(lastidx, idx);
        lastidx = idx;
        idx = message.indexOf("\t", lastidx + 1);
        String sender;
        if (idx != -1) {
            sender = message.substring(lastidx + 1, idx);
        } else {
            sender = message.substring(lastidx + 1,
message.length());
```

```
        }
        if (sender.equals(thisUser.getIPAddress())) {
            return;
        }
        if (header.equals("PUBLIC_CHAT_MSG")) {
            mainWindow.receivePublicChatMessage(message);
        } else if (header.equals("INTRODUCE")) {
            AppUser user = null;
            LinkSocket linksocket = null;
            Socket socket = null;
            arr = getMessageElement(message);
            user = AppUser.getUserWithThisIP(arr[1]);
            if (user != null) {
                if (user.getStatus() == 0) {
                    user.setStatus(1);
                } else {
                    return;
                }
            }
            linksocket = LinkSocket.getLinkSocket(arr[1]);
            if (linksocket != null) {
                return;
            }
            try {
                socket = new Socket(arr[1], 49252);
                linksocket = new LinkSocket(socket);
                if (user == null) {
                    user = new AppUser(arr[1], arr[2],
linksocket);
                } else {
                    user.rename(arr[2]);
                    user.changeSocket(linksocket);
                }
                mainWindow.addNewUser(user);
                linksocket.start();
                sendIntroduceReply(linksocket);
            } catch (IOException e) {
                e.printStackTrace();
            }
            mainWindow.refreshDisplay();
        } else if (header.equals("NEW_GROUP")) {
            arr = getMessageElement(message);
            AppUser user = AppUser.getUserWithThisIP(arr[1]);
            if (user == null) {
                return;
            }
            ConfGroup group = new ConfGroup(arr[2],
arr[3].equals("1"), Integer.parseInt(arr[4]));
            group.addModerator(user);
            mainWindow.addNewGroup(group);
        } else if (header.equals("JOIN_GROUP")) {
            arr = getMessageElement(message);
            AppUser user = AppUser.getUserWithThisIP(arr[1]);
            if (user == null) {
                return;
            }
```

```
            AppUser mod = AppUser.getUserWithThisIP(arr[2]);
            if (mod == null) {
                return;
            }
            ConfGroup group = mod.getGroup(arr[3]);
            if (group != null && !group.isJoining(user)) {
                group.addMember(user);
                mainWindow.joinConfGroup(user, group);
            }
        } else if (header.equals("LEAVE_GROUP")) {
            arr = getMessageElement(message);
            AppUser user = AppUser.getUserWithThisIP(arr[1]);
            if (user == null) {
                return;
            }
            AppUser mod = AppUser.getUserWithThisIP(arr[2]);
            if (mod == null) {
                return;
            }
            ConfGroup group = mod.getGroup(arr[3]);
            if (group != null) {
                group.removeMember(user);
                mainWindow.leaveConfGroup(user, group);
            }
        } else if (header.equals("END_GROUP")) {
            arr = getMessageElement(message);
            AppUser mod = AppUser.getUserWithThisIP(arr[1]);
            if (mod == null) {
                return;
            }
            ConfGroup group = mod.getGroup(arr[2]);
            if (group != null) {
                mainWindow.endConfGroup(group);
            }
        } else if (header.equals("RENAME")) {
            arr = getMessageElement(message);
            AppUser user = AppUser.getUserWithThisIP(arr[1]);
            if (user == null) {
                return;
            }
            mainWindow.receiveRenameMessage(user, arr[2]);
        } else if (header.equals("PORT_INFO")) {
            arr = getMessageElement(message);
            AppUser user = AppUser.getUserWithThisIP(arr[1]);
            if (user == null) {
                return;
            }
            user.setPort(Integer.parseInt(arr[2]),
Integer.parseInt(arr[3]));
            if (queueVector.indexOf(user) != -1) {
                queueVector.remove(user);
            }
            mainWindow.refreshDisplay();
        } else if (header.equals("STATUS_UPDATE")) {
            arr = getMessageElement(message);
            AppUser user = AppUser.getUserWithThisIP(arr[1]);
```

```
            if (user == null) {
                return;
            }
            user.setStatus(Integer.parseInt(arr[2]));
            mainWindow.refreshDisplay();
        }
    }

public static void receiveGroupMulticastMessage(String message,
MulticastReceiver mcReceiver) {
        if (message == null) {
            return;
        }
        System.out.println((new StringBuilder("Receive Group
Multicast Msg ")).append(mcReceiver.getAddress()).append(":
\n").append(message).toString());
        int lastidx = 0;
        String arr[] = (String[]) null;
        int idx = message.indexOf("\t", lastidx + 1);
        String header = message.substring(lastidx, idx);
        lastidx = idx;
        idx = message.indexOf("\t", lastidx + 1);
        String sender;
        if (idx != -1) {
            sender = message.substring(lastidx + 1, idx);
        } else {
            sender = message.substring(lastidx + 1,
message.length());
        }
        if (sender.equals(thisUser.getIPAddress())) {
            return;
        }
        if (header.equals("GROUP_CHAT_MSG")) {
            ConfGroup group =
ConfGroup.getGroupWithThisReceiver(mcReceiver);
            if (group != null) {
                mainWindow.receiveGroupChatMessage(group,
message);
            }
        } else if (header.equals("CLOSE_STREAM")) {
            arr = getMessageElement(message);
            AppUser user = AppUser.getUserWithThisIP(arr[1]);
            if (user == null) {
                return;
            }
            ConfGroup group =
ConfGroup.getGroupWithThisReceiver(mcReceiver);
        if (header.equals("ALLOW_SPEAK")) {
            arr = getMessageElement(message);
            AppUser user = AppUser.getUserWithThisIP(arr[2]);
            if (user == null) {
                return;
            }
            ConfGroup group =
ConfGroup.getGroupWithThisReceiver(mcReceiver);
```

```
    public static void receiveSocketMessage(String message,
LinkSocket linksock) {
        if (message == null) {
            return;
        }
        String arr[] = message.split("\t");
        InetAddress addr = linksock.getInetAddress();
        System.out.println((new StringBuilder("Receive socket msg
from ")).append(addr.getHostAddress()).append(" :
\n").append(message).toString());
        for (int i = 0; i < arr.length; i++) {
            arr[i] = arr[i].trim();
        }

        if (arr[0].equals("PRIVATE_CHAT_MSG")) {
            AppUser user =
AppUser.getUserWithThisSocket(linksock);
            if (user != null) {
                mainWindow.receivePrivateChatMessage(user,
message);
            }
        } else if (arr[0].equals("JOIN_REQUEST")) {
            ConfGroup group = thisUser.getGroup(arr[1]);
            if (group != null) {
                AppUser user =
AppUser.getUserWithThisSocket(linksock);
                if (group.isJoining(user)) {
                    return;
                }
                if (!group.getType()) {
                    if (group.getPassword().equals(arr[2])) {
                        sendRequestReply(linksock, group, 1);
                        group.addMember(user);
                        mainWindow.joinConfGroup(user, group);
                    } else {
                        sendRequestReply(linksock, group, 3);
                    }
                } else {
                    sendRequestReply(linksock, group, 1);
                    group.addMember(user);
                    mainWindow.joinConfGroup(user, group);
                }
            }
        } else if (arr[0].equals("KICK_MEMBER")) {
            AppUser mod = AppUser.getUserWithThisIP(arr[1]);
            if (mod == null) {
                return;
            }
            ConfGroup group = mod.getGroup(arr[2]);
            if (group == null) {
                return;
            }
            if (!thisUser.isJoining(group)) {
                return;
            }
            mainWindow.receiveKickMessage(group);
```

```
        } else if (arr[0].equals("INVITE_USER")) {
            AppUser mod = AppUser.getUserWithThisIP(arr[1]);
            if (mod == null) {
                return;
            }
            ConfGroup group = mod.getGroup(arr[2]);
            if (group == null) {
                return;
            }
            if (thisUser.isJoining(group)) {
                return;
            }
           mainWindow.receiveInviteMessage(group);

public static void sendIntroduceMessage() {
        asFirstUser = true;
        thisUser.setStatus(1);
        mainWindow.refreshDisplay();
        String msg = (new
StringBuilder("INTRODUCE\t")).append(thisUser.getIPAddress()).appe
nd("\t").append(thisUser.getUserName()).toString();
        publicMcSender.send(msg);
        (new TimeoutThread(1, 3000)).start();
    }

    public static void sendIntroduceReply(LinkSocket linkSock) {
        String msg = (new
StringBuilder("INTRO_REPLY\t")).append(thisUser.getIPAddress()).ap
pend("\t").append(thisUser.getUserName()).append("\t").append(this
User.getStatus()).toString();
        if (thisUser.getStatus() != 1) {
            msg = (new
StringBuilder(String.valueOf(msg))).append("\t").append(thisUser.g
etVideoPort()).append("\t").append(thisUser.getAudioPort()).toStri
ng();
        }
        if (linkSock != null) {
            linkSock.send(msg);
        }
    }

    public static void sendCheckGroupMessage() {
        thisUser.setStatus(2);
        String msg = "ASK_GROUP";
        int retryCount = 0;
        AppUser user = null;
        LinkSocket linksock = null;
        groupInfoReplyStats = false;
        boolean send;
        do {
            send = false;
            for (int i = 0; i < userVector.size(); i++) {
                user = (AppUser) (AppUser)
userVector.elementAt(i);
                if (user.getStatus() != 3) {
                    continue;
```

```
                }
                linksock = user.getLinkSocket();
                if (linksock == null) {
                    continue;
                }
                linksock.send(msg);
                (new TimeoutThread(2, 4000)).start();
                send = true;
                break;
            }
            if (!send) {
                delayThread(2000);
                retryCount++;
            }
        } while (!send && retryCount < 4);
        if (!send) {
            thisUser.setStatus(3);
            sendStatusUpdate();
        }
        mainWindow.refreshDisplay();
    }

public static void sendToAllUser(String message) {
        AppUser user = null;
        LinkSocket linksock = null;
        for (int i = 0; i < userVector.size(); i++) {
            user = (AppUser) (AppUser) userVector.elementAt(i);
            linksock = user.getLinkSocket();
            if (linksock != null) {
                linksock.send(message);
            }
        }
    }

    public static void sendToPublic(String message) {
        if (publicMcSender != null) {
            publicMcSender.send(message);
        }
    }

    public static void sendToUser(String message, AppUser user) {
        if (user == null) {
            return;
        }
        LinkSocket ls = user.getLinkSocket();
        if (ls != null) {
            ls.send(message);
        }
    }

    public static void sendToGroup(String message, ConfGroup
group) {
        if (group == null) {
            return;
        }
        MulticastSender mcSender = group.getMulticastSender();
```

```
        if (mcSender != null) {
            mcSender.send(message); }}
```

C. LinkSocketServer.java

```
public class LinkSocketServer extends Thread {
    private ServerSocket serverSocket;
    private boolean listening;
    class ProcessLinkThread extends Thread {
        @Override
        public void run() {
            LinkSocket lsock = null;
            if (socket != null) {
                lsock = new LinkSocket(socket);
                lsock.start();
            }
        }
        Socket socket;
        final LinkSocketServer this$0
        public ProcessLinkThread(Socket socket) {
            super();
            this$0 = LinkSocketServer.this;
            this.socket = null;
            this.socket = socket;
        }
    }

    public LinkSocketServer(int port) {
        serverSocket = null;
        listening = true;
        try {
            serverSocket = new ServerSocket(port);
        } catch (IOException e) {
            System.err.println((new StringBuilder("Could not
listen on port: ")).append(port).append(".").toString());
            e.printStackTrace();
        }
    }

    public boolean isFunctioning() {
        return serverSocket != null;
    }

    public void close() {
        listening = false;
        try {
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    @Override
    public void run() {
        LinkSocket lsock = null;
        listening = true;
        while (listening) {
            try {
```

```
                Socket socket = serverSocket.accept();
                (new ProcessLinkThread(socket)).start();
            } catch (IOException e) {
                System.err.println("IOException in
LinkSocketServer run()");
                e.printStackTrace();
            }
        }
    }}
```

D. LinkSocket.java

```
public class LinkSocket extends Thread {
    private static Vector linkVector = new Vector();
    private Socket socket;
    private ObjectOutputStream output;
    private ObjectInputStream input;
    private boolean functioning;

    class ProcessReceiveThread extends Thread {
        @Override
        public void run() {
            LinkManager.receiveSocketMessage(message, ls);
        }
        String message;
        LinkSocket ls;
        final LinkSocket this$0;
        public ProcessReceiveThread(String msg, LinkSocket ls) {
            super();
            this$0 = LinkSocket.this;
            message = msg;
            this.ls = ls;
        }
    }

    class ProcessSendThread extends Thread {
        @Override
        public void run() {
            try {
                output.writeObject(message);
                output.flush();
            } catch (IOException e) {
                e.printStackTrace();
                System.out.println("Error sending message");
            }
        }
        String message;
        final LinkSocket this$0;
        public ProcessSendThread(String msg) {
            super();
            this$0 = LinkSocket.this;
            message = msg;
        }
    }
```

```
    public LinkSocket(Socket socket) {
        this.socket = null;
        this.socket = socket;
        try {
            output = new
ObjectOutputStream(socket.getOutputStream());
            output.flush();
            input = new
ObjectInputStream(socket.getInputStream());
            functioning = true;
            linkVector.add(this);
        } catch (IOException e) {
            e.printStackTrace();
            functioning = false;
        }}
    public static LinkSocket getLinkSocket(String IP) {
        for (int i = 0; i < linkVector.size(); i++) {
            LinkSocket ls = (LinkSocket) (LinkSocket)
linkVector.elementAt(i);
            InetAddress addr = ls.getInetAddress();
            if (addr.getHostAddress().equals(IP)) {
                return ls;
            }
       }
        return null;
    }

    public InetAddress getInetAddress() {
        if (socket == null) {
            return null;
        } else {
            return socket.getInetAddress();
        }
    }

    public void close() {
        try {
            output.close();
            input.close();
            socket.close();
            functioning = false;
            linkVector.remove(this);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    public void send(String msg) {
        if (functioning && output != null) {
            (new ProcessSendThread(msg)).start();
        } else {
            System.out.println("No output stream!");
        }
```

```
    }

    @Override
    public void run() {
        if (!functioning) {
            return;
        }
        String message = null;
        do {
            try {
                message = (String) input.readObject();
                (new ProcessReceiveThread(message, this)).start();
            } catch (ClassNotFoundException
classNotFoundException) {
                System.out.println("ERROR : Unknown object type
received");
            } catch (IOException ioExc) {
                functioning = false;
                AppUser user =
AppUser.getUserWithThisSocket(this);
                if (user != null) {
                    System.out.println((new StringBuilder("ERROR
IOException: ")).append(user.getUserName()).append(" -
").append(user.getIPAddress()).toString());
                    LinkManager.removeUser(user);
                }
            }
        } while (functioning);
    }
}
```

E. MulticastSender.java

```
public class MulticastSender {
    private String mcAddr;
    private int port;
    private boolean functioning;
    private MulticastSocket mcSocket;
    private InetAddress address;
    private DatagramPacket packet;

    class SenderThread extends Thread {
        @Override
        public void run() {
                synchronized (mcSocket) {
                    byte buf[] = new byte[256];
                    buf = dString.getBytes();
                    try {
                        packet = new DatagramPacket(buf,
buf.length, address, port);
                        mcSocket.send(packet);

                    } catch (IOException e) {
                        System.out.println("IOException in
MulticastSender run()");
```

```
                        e.printStackTrace();
                    }
                }
                return;
        }
        private String dString;
        final MulticastSender this$0;

        public SenderThread(String msg) {
            super();
            this$0 = MulticastSender.this;
            dString = msg;
        }
    }

    public MulticastSender(String mcAddr, int port) {
        this.mcAddr = mcAddr;
        this.port = port;
        try {
            address = InetAddress.getByName(mcAddr);
            mcSocket = new MulticastSocket(port);
            mcSocket.joinGroup(address);
            functioning = true;
        } catch (IOException e) {
            System.out.println("IOException in MulticastSender
constructor");
            e.printStackTrace();
            functioning = false;
        }
    }

    public void close() {
        mcSocket.close();
        functioning = false;
    }

    public void send(String msg) {
        if (functioning) {
            (new SenderThread(msg)).start();
        } else {
            System.out.println((new StringBuilder("MulticastSender
(")).append(getAddress()).append(") is not
functioning").toString());
        }
    }

    public String getAddress() {
        return address.getHostAddress();
    }

    public int getPort() {
        return mcSocket.getPort();
    }}
```

F. MulticastReceiver.java

```
public class MulticastReceiver extends Thread {
    private MulticastSocket mcSocket;
    private InetAddress address;
    private DatagramPacket packet;
    private boolean listening;

    class ProcessReceiveThread extends Thread {
        @Override
        public void run() {
            String message = new String(packet.getData());
            if (receiver == LinkManager.publicMcReceiver) {

LinkManager.receivePublicMulticastMessage(message);
            } else {
                LinkManager.receiveGroupMulticastMessage(message,
receiver);
            }
        }
        DatagramPacket packet;
        MulticastReceiver receiver;
        final MulticastReceiver this$0;
        public ProcessReceiveThread(DatagramPacket packet,
MulticastReceiver rec) {
            super();
            this$0 = MulticastReceiver.this;
            this.packet = packet;
            receiver = rec;
        }
    }

    public MulticastReceiver(String mcAddr, int port) {
        try {
            address = InetAddress.getByName(mcAddr);
            mcSocket = new MulticastSocket(port);
            mcSocket.joinGroup(address);
        } catch (IOException e) {
            System.out.println("IOException in MulticastReceiver
constructor");
            e.printStackTrace();
        }
    }

    public String getAddress() {
        return address.getHostAddress();
    }
    public int getPort() {
        return mcSocket.getPort();
    }
    public void close() {
        try {
            mcSocket.leaveGroup(address);
            mcSocket.close();
            listening = false;
        } catch (IOException e) {
```

```
            System.out.println("IOException in MulticastReceiver
stopThread()");
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        listening = true;
        do {
            try {
                byte buf[] = new byte[256];
                packet = new DatagramPacket(buf, buf.length);
                mcSocket.receive(packet);
                (new ProcessReceiveThread(packet, this)).start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        } while (listening);
        System.out.println("MulticastReceiver is closed...");
    }
}
```

G. ConfGroup.java

```
public ConfGroup(String gname, boolean type, String pwd) {
        mcReceiver = null;
        mcSender = null;
        groupName = null;
        mcAddress = null;
        memberVector = null;
        moderator = null;
        groupID = getNextGroupID();
        if (groupID == -1) {
            System.out.println("Maximum number of group");
            return;
        }
        mcAddress = "239.168.0.x".replaceAll("x",
String.valueOf(groupID));
        groupName = gname;
        groupType = type;
        password = pwd;
        memberVector = new Vector();
        if (groupVector.indexOf(this) == -1) {
            groupVector.add(this);
        }
    }

    public ConfGroup(String gname, boolean type, int id) {
        mcReceiver = null;
        mcSender = null;
        groupName = null;
        mcAddress = null;
        memberVector = null;
        moderator = null;
```

```
        mcAddress = "239.168.0.x".replaceAll("x",
String.valueOf(id));
        groupName = gname;
        groupType = type;
        password = "";
        groupID = id;
        memberVector = new Vector();
        if (groupVector.indexOf(this) == -1) {
            groupVector.add(this);
        }
    }

    private void reassignDuplicatedID() {
        for (int i = 0; i < groupVector.size(); i++) {
            ConfGroup group1 = (ConfGroup)
groupVector.elementAt(i);
            boolean ok = true;
            for (int j = 0; j < groupVector.size(); j++) {
                ConfGroup group2 = (ConfGroup)
groupVector.elementAt(j);
                if (i == j || group1.groupID != group2.groupID ||
group1.getModeratorIP().compareTo(group2.getModeratorIP()) <= 0) {
                    continue;
                }
                int newID = getNextGroupID();
                System.out.println((new StringBuilder("REASSIGN
")).append(group1.getGroupName()).append(" :
").append(newID).toString());
                group1.setNewID(newID);
                ok = false;
                break;
            }

            if (!ok) {
                i = 0;
            }
        }

    }

    private int getNextGroupID() {
        int j = 1;
        boolean b;
        do {
            b = false;
            for (int i = 0; i < groupVector.size(); i++) {
                ConfGroup group = (ConfGroup)
groupVector.elementAt(i);
                if (group.getGroupID() != j) {
                    continue;
                }
                b = true;
                break;
            }

            if (b) {
```

```
                j++;
            }
        } while (b);
        if (j < 255) {
            return j;
        } else {
            return -1;
        }
    }

    public void addModerator(AppUser mod) {
        if (mod == null || moderator == mod) {
            return;
        }
        moderator = mod;
        mod.createGroup(this);
        if (memberVector.indexOf(mod) == -1) {
            memberVector.add(mod);
        }
        reassignDuplicatedID();
    }

    public void addMember(AppUser member) {
        if (member == null || memberVector.indexOf(member) != -1)
{
            return;
        } else {
            memberVector.add(member);
            member.joinGroup(this);
            return;
        }
    }

    public void removeMember(AppUser member) {
        if (member == null || memberVector.indexOf(member) == -1)
{
            return;
        } else {
            memberVector.remove(member);
            member.leaveGroup(this);
            return;
        }
    }

    public void endGroup() {
        if (groupVector.indexOf(this) == -1) {
            return;
        }
        groupVector.remove(this);
        moderator.leaveGroup(this);
        for (int i = 0; i < memberVector.size(); i++) {
            AppUser member = (AppUser) memberVector.elementAt(i);
            member.leaveGroup(this);
        }

    }
```

```
    public void setNewID(int newID) {
        groupID = newID;
        mcAddress = "239.168.0.x".replaceAll("x",
String.valueOf(groupID));
    }

    public void startGroup() {
        stopGroup();
        mcReceiver = new MulticastReceiver(mcAddress, 50000);
        mcSender = new MulticastSender(mcAddress, 50000);
        mcReceiver.start();
    }

    public void stopGroup() {
        if (mcReceiver != null) {
            mcReceiver.close();
            mcReceiver = null;
        }
        if (mcSender != null) {
            mcSender.close();
            mcSender = null;
        }
    }

    public String getModeratorIP() {
        return moderator.getIPAddress();
    }

    public String getModeratorName() {
        return moderator.getUserName();
    }

    public boolean isJoining(AppUser user) {
        if (getModerator() == user) {
            return true;
        }
        return memberVector.indexOf(user) != -1;
    }

    public static ConfGroup
getGroupWithThisReceiver(MulticastReceiver mcRec) {
        ConfGroup group = null;
        for (int i = 0; i < groupVector.size(); i++) {
            group = (ConfGroup) groupVector.elementAt(i);
            if (group.mcReceiver == mcRec) {
                return group;
            }
        }

        return null;
    }

    public static void printGroupInfo() {
        System.out.println("\nConfGroup Info : ");
```

```
        if (groupVector.size() == 0) {
            System.out.println("   No group available!");
        }
        for (int i = 0; i < groupVector.size(); i++) {
            ConfGroup group = (ConfGroup)
groupVector.elementAt(i);
            Vector memberVector = group.getMemberVector();
            System.out.println((new StringBuilder(String.valueOf(i
+ 1))).append(". ").append(group.getGroupName()).append("
(").append(group.getMcAddress()).append(")
").append(group.getType() ? "Public" : "Private").append("
").append(group.getPassword()).toString());
            System.out.println((new StringBuilder("   Moderator :
")).append(group.getModerator().getUserName()).toString());
            System.out.print("   Member : ");
            for (int j = 0; j < memberVector.size(); j++) {
                AppUser member = (AppUser) (AppUser)
memberVector.elementAt(j);
                System.out.print(member.getUserName());
                if (member != memberVector.lastElement()) {
                    System.out.print(", ");
                }
            }

            System.out.println("");
        }

    }
}
```

H. AppUser.java

```
public class AppUser {

    public AppUser(String IP, String name, LinkSocket ls) {
        IPAddr = null;
        username = null;
        groupVector = null;
        linksocket = null;
        IPAddr = IP;
        username = name;
        linksocket = ls;
        groupVector = new Vector();
        status = 1;
        if (userVector.indexOf(this) == -1) {
            userVector.add(this);
        }
    }

    public static void sortVector() {
        AppUser arrUser[] = new AppUser[userVector.size()];
        userVector.copyInto(arrUser);
        insertionSort(arrUser);
```

```
    }

    private static void insertionSort(AppUser arrUser[]) {
        for (int i = 1; i < arrUser.length; i++) {
            int j = i - 1;
            AppUser temp;
            for (temp = arrUser[i]; j >= 0 &&
arrUser[j].username.compareTo(temp.username) > 0; j--) {
                arrUser[j + 1] = arrUser[j];
            }

            arrUser[j + 1] = temp;
        }

        userVector.clear();
        for (int i = 0; i < arrUser.length; i++) {
            userVector.add(arrUser[i]);
        }

    }

    public void joinGroup(ConfGroup group) {
        if (group == null || isJoining(group)) {
            return;
        }
        if (groupVector.indexOf(group) == -1) {
            groupVector.add(group);
        }
        group.addMember(this);
    }

    public void createGroup(ConfGroup group) {
        if (group == null) {
            return;
        }
        if (groupVector.indexOf(group) == -1) {
            groupVector.add(group);
        }
        group.addModerator(this);
    }

    public void leaveGroup(ConfGroup group) {
        if (group == null || !isJoining(group)) {
            return;
        }
        groupVector.remove(group);
        if (group.getModerator() == this) {
            group.endGroup();
        } else {
            group.removeMember(this);
        }
    }



    public void rename(String newName) {
```

```
        username = newName;
    }

    public void changeSocket(LinkSocket newSocket) {
        LinkSocket temp = linksocket;
        linksocket = newSocket;
        if (temp != null) {
            temp.close();
        }
    }

    public void setInactive() {
        if (linksocket != null) {
            linksocket.close();
            linksocket = null;
        }
        for (int i = 0; i < groupVector.size(); i++) {
            ConfGroup group = (ConfGroup)
groupVector.elementAt(i);
            leaveGroup(group);
        }

        status = 0;
        groupVector = new Vector();

    }

    public static void removeUserWithThisIP(String remIP) {
        AppUser user = getUserWithThisIP(remIP);
        if (user != null) {
            user.setInactive();
        }
    }

    public static void removeUserWithThisSocket(LinkSocket socket)
{
        AppUser user = getUserWithThisSocket(socket);
        if (user != null) {
            user.setInactive();
        }
    }

    public ConfGroup getGroup(String groupName) {
        for (int i = 0; i < groupVector.size(); i++) {
            ConfGroup group = (ConfGroup)
groupVector.elementAt(i);
            if (group.getGroupName().equals(groupName)) {
                return group;
            }
        }

        return null;
    }

    public boolean isJoining(ConfGroup group) {
        if (group.getModerator() == this) {
```

```
            return true;
        }
        return groupVector.indexOf(group) != -1;
    }

    public boolean isModeratorOf(String groupName) {
        for (int i = 0; i < groupVector.size(); i++) {
            ConfGroup group = (ConfGroup)
groupVector.elementAt(i);
            if (group.getModerator() == this &&
group.getGroupName().equals(groupName)) {
                return true;
            }
        }

        return false;
    }

    public static AppUser getUserWithThisIP(String IP) {
        if (userVector.size() == 0) {
            return null;
        }
        for (int i = 0; i < userVector.size(); i++) {
            AppUser user = (AppUser) userVector.elementAt(i);
            if (user.IPAddr.equals(IP)) {
                return user;
            }
        }

        return null;
    }

    public static AppUser getUserWithThisSocket(LinkSocket socket)
{
        for (int i = 0; i < userVector.size(); i++) {
            AppUser user = (AppUser) userVector.elementAt(i);
            if (user.linksocket == socket) {
                return user;
            }
        }

        return null;

    }



    public static AppUser getUserWithThisIndex(int i) {
        if (i < 0 || i > userVector.size() - 1) {
            return null;
        } else {
            return (AppUser) userVector.elementAt(i);
        }
    }

    public static int getIndexOf(AppUser user) {
```

```
        return userVector.indexOf(user);
    }

    public static void printUserInfo() {
        System.out.println("\nAppUser Info : ");
        if (userVector.size() == 0) {
            System.out.println("   No user online!");
        }
        for (int i = 0; i < userVector.size(); i++) {
            AppUser user = (AppUser) userVector.elementAt(i);
            System.out.println((new StringBuilder(String.valueOf(i
+ 1))).append(". ").append(user.getUserName()).append(" -
").append(user.getIPAddress()).append("
(").append(user.getStatus()).toString());
            Vector groupVector = user.getGroupVector();
            System.out.println("     Join group : ");
            for (int j = 0; j < groupVector.size(); j++) {
                ConfGroup group = (ConfGroup) (ConfGroup)
groupVector.elementAt(j);
                System.out.println((new StringBuilder("
")).append(j).append(".
").append(group.getGroupName()).toString());
                if (group.getModerator() == user) {
                    System.out.print(" as moderator\n");
                } else {
                    System.out.print(" as member\n");
                }
            }

        }

        LinkManager.seeQueue();
    }}
```