LAMPIRAN:

KODE SUMBER

IrcApplication.java

```
/* Nama file:   IrcApplication.java
 * Author:      (c) 2005 Novareza Klifartha
 *
 * IrcApplication merupakan kelas aplikasi MIDlet J2ME.
 * Class ini hanya mengatur hubungan aplikasi ke MIDlet saja,
 * dan hanya berisi penciptaan obyek-obyek utama yang
 * digunakan MIDlet.
 *
 * Implementasi utama dikendalikan oleh dua kelas, yaitu:
 *      (1) IrcConnection, kelas yang mengatur koneksi jaringan
 *          dan data-data protokol.
 *      (2) IrcMain, kelas yang mengatur koordinasi antar kelas dan
 *          dalam penggunaan thread.
 *
 * VERSION HISTORY:
 * 1.0        - Basic feature testing passed well
 *
 * 0.9_rc2    - Remove application icon for optimizing file size
 *            - Add garbage collector for optimizing memory
 *
 * 0.9_rc1    - First complete implementation
 *            - First complete code comments
 *            - First complete user interface
 *            - Optimize memory usage
 *            - Optimize thread usage
 *            - Optimize some methods
 *
 * 0.9_04     - Add change nickname
 *            - Optimize code
 *            - Add code comments
 *
 * 0.9_03     - Support join channel with key
 *            - Add chat user feature without joining channel
 *
 * 0.9_02     - Improve user interface
 *            - Optimize some algorithm
 *            - Fix some protocol implementation
 *            - Add code comments
 *
 * 0.9_01     - Basic implementation
 */
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class IrcApplication extends MIDlet {
    // kelas untuk koneksi
    private IrcConnection connection;
    // kelas utama
    private IrcMain main;
```

```
    // aplikasi MIDlet mulai dijalankan
    public void startApp() {
        try {
            // buat koneksi;
            connection = new IrcConnection(this);
            // modul utama
            main = new IrcMain(this, connection);
        }
        catch (IllegalArgumentException iae) {}
    }

    // aplikasi MIDlet di-suspend
    public void pauseApp() {}

    // aplikasi MIDlet dihentikan
    public void destroyApp(boolean unconditional) {}
}
```

IrcChannelUser.java

```
/* Nama file:   IrcChannelUser.java
 * Author:      (c) 2005 Novareza Klifartha
 **/

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.*;

public class IrcChannelUser {
    // display
    private Display dChannelUser;
    // koneksi
    private IrcConnection cChannelUser;
    // kelas utama
    private IrcMain mChannelUser;
    // form
    private Form fJoin, fChatUser, fChangeNickname;
    // kotak input
    private TextField tfChannel, tfKey, tfChatUser,
tfChangeNickname;
    // command
    private Command cJoin, cCancel;
    // kotak teks
    private TextBox tbLastDisplay, tbActiveDisplay;
    private TextBox tbLastChannel, tbLastUser, tbChannel, tbUser;
    // form pilihan
    private List lSelectUser, lLastSelectUser;
    private List lLastDisplay;
    // vector (dynamic array)
    private Vector vChannel, vNames, vUser;

    /*
     * Constructor
     **/
    public IrcChannelUser(Display d, IrcConnection c, IrcMain m) {
        // simpan obyek
        this.dChannelUser = d;
        this.cChannelUser = c;
        this.mChannelUser = m;
        // pembuatan command
        cJoin = new Command("Join", Command.OK, 1);
```

```
        cCancel = new Command("Cancel", Command.CANCEL, 1);
        // pembuatan tampilan "join Channel"
        createJoin();
    }

    /*
     * Membuat antarmuka untuk join ke channel
     **/
    public void createJoin() {
        fJoin = new Form("Join Channel");
        tfChannel = new TextField("Channel", "#myroom", 24,
TextField.ANY);
        tfKey = new TextField("Key", null, 24, TextField.ANY);
        fJoin.append(tfChannel);
        fJoin.append(tfKey);
        fJoin.addCommand(cJoin);
        fJoin.addCommand(cCancel);
        fJoin.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    if (c == cJoin) {
                        onJoin();
                    }
                    else if (c == cCancel) {
                        onStatus();
                    }
                }
            }
        );
    }

    /*
     * Menampilkan status
     **/
    private void onStatus() {
        mChannelUser.showStatus();
    }

    /*
     * Menampilkan daftar display
     **/
    public void showDisplay() {
        boolean status = false;
        TextBox t;
        List l = new List("Available Display", Choice.EXCLUSIVE);
        // channel display
        if ((vChannel != null) && (vChannel.size() != 0)) {
            for (int i = 0; i<= vChannel.size() - 1; i++) {
                t = (TextBox) vChannel.elementAt(i);
                l.append(t.getTitle(), null);
            }
            status = true;
        }
        // user display
        if ((vUser != null) && (vUser.size() != 0)) {
            for (int i = 0; i<= vUser.size() - 1; i++) {
                t = (TextBox) vUser.elementAt(i);
                l.append(t.getTitle(), null);
            }
            status = true;
        }
        //
```

```
        l.addCommand(new Command("Back", Command.BACK, 1));
        l.addCommand(new Command("Select", Command.OK, 2));
        l.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    if (c.getLabel().equals("Back")) {
                        onStatus();
                    }
                    else if (c.getLabel().equals("Select")) {
                        lLastDisplay = (List)
dChannelUser.getCurrent();
                        onDisplay();
                    }
                }
            } // end of "new CommandListener() {}"
        ); // end of "setCommandListener() {}"
        if (status == true) {
            dChannelUser.setCurrent(l);
        }
        else {
            new IrcFunction().msgError(dChannelUser, "No display to
show!");
        }
    }

    /*
     * Menampilkan antarmuka join ke channel
     **/
    public void showJoin() {
        dChannelUser.setCurrent(fJoin);
    }

    /*
     * Proses join
     **/
    private void onJoin() {
        boolean foundInVector = false;
        // apakah vector "null"
        if (vChannel == null) {
            vChannel = new Vector();
        }
        // jika tidak "null", cari di vector
        else {
            // cari...
            if (findChannel(tfChannel.getString()) != -1) {
                foundInVector = true;
            }
        }
        // jika pencarian gagal
        if (foundInVector == false) {
            cChannelUser.ircJoin(tfChannel.getString(),
tfKey.getString());
        }
        // wait...
        dChannelUser.setCurrent(new TextBox("Please wait...", "", 1,
TextField.ANY));
        // reset input channel dan key
        tfChannel.setString("#myroom");
        tfKey.setString("");
    }

    /*
```

```
 * Mencari display untuk suatu channel
 **/
private int findChannel(String channel) {
    if (channel.charAt(0) != '#') {
        channel = '#' + channel;
    }
    int min = 0;
    int max = -1;
    // apakah null?
    if (vChannel != null) {
        max = vChannel.size() - 1;
    }
    // cari index
    int index = -1;
    if (max >= 0) {
        tbChannel = (TextBox) vChannel.elementAt(0);
        if (tbChannel.getTitle().equals(channel)) {
            index = 0;
            dChannelUser.setCurrent(tbChannel);
        }
    }
    else {
        for (int i = min; i <= max; i++) {
            tbChannel = (TextBox) vChannel.elementAt(i);
            if (tbChannel.getTitle().equals(channel)) {
                index = i;
                dChannelUser.setCurrent(tbChannel);
                break;
            }
        }
    }
    return index;
}

/*
 * Mengambil display untuk channel
 **/
private TextBox getChannel(int index) {
    return (TextBox) vChannel.elementAt(index);
}

/*
 * Mengambil display untuk user
 **/
private TextBox getUser(int index) {
    return (TextBox) vUser.elementAt(index);
}

/*
 * Menentukan display untuk channel
 **/
private void setChannel(int index) {
    tbLastDisplay = tbActiveDisplay;
    tbActiveDisplay = (TextBox) vChannel.elementAt(index);
    dChannelUser.setCurrent(tbActiveDisplay);
}

/*
 * Menentukan display untuk memilih user
 **/
private void setNames(int index) {
    tbLastDisplay = tbActiveDisplay;
    lSelectUser = (List) vNames.elementAt(index);
```

```
    dChannelUser.setCurrent(lSelectUser);
}

/*
 * Menentukan display untuk user
 **/
private void setUser(int index) {
    tbLastDisplay = tbActiveDisplay;
    tbActiveDisplay = (TextBox) vUser.elementAt(index);
    dChannelUser.setCurrent(tbActiveDisplay);
}

/*
 * Mengaktifkan display untuk target yang dituju
 **/
public boolean showDisplayToTarget(IrcMessage pm) {
    boolean status = false;
    int index = -1;
    // ke channel
    if (pm.getTarget().charAt(0) == '#') {
        // reset object
        if (vChannel != null) {
            if (vChannel.size() == 0) {
                vChannel = null;
            }
        }
        // jika display channel belum diciptakan
        if (vChannel == null) {
            vChannel = new Vector();
            tbActiveDisplay = createChannel(pm.getTarget());
            vChannel.addElement(tbActiveDisplay);
            dChannelUser.setCurrent(tbActiveDisplay);
        }
        // jika display channel sudah ada
        else {
            index = findChannel(pm.getTarget());
            setChannel(index);
        }
    }
    // ke user
    else {
        // reset object
        if (vUser != null) {
            if (vUser.size() == 0) {
                vUser = null;
            }
        }
        // jika display user belum diciptakan
        if (vUser == null) {
            vUser = new Vector();
            tbActiveDisplay = createUser(pm.getSender());
            vUser.addElement(tbActiveDisplay);
            new IrcFunction().appendTextBox(tbActiveDisplay,
                "<" + pm.getSender() + "> " + pm.getMessage());
            dChannelUser.setCurrent(tbActiveDisplay);
        }
        // jika display user sudah ada
        else {
            index = findUser(pm.getSender());
            setUser(index);
        }
    }
```

```
            // menampilkan display sesuai index-nya
            if (index != -1) {
                status = true;
                // mendeteksi jenis message
                if (pm.MessageType == pm.MSG_PRIVMSG) {
                    new IrcFunction().appendTextBox(tbActiveDisplay,
                        "<" + pm.getSender() + "> " + pm.getMessage());
                }
                else if (pm.MessageType == pm.MSG_JOIN) {
                    // ada yang JOIN
                    lSelectUser = (List) vNames.elementAt(index);
                    // cari.. kalo belum ada, tambahkan!
                    boolean userFound = false;
                    int j = lSelectUser.size() - 1;
                    for (int i = 0; i <= j; i++) {
                        if
(lSelectUser.getString(i).equals(pm.getSender())) {
                            userFound = true;
                        }
                    }
                    // tambahkan ke daftar user
                    if (!userFound) {
                        lSelectUser.append(pm.getSender(), null);
                    }
                }
                else if (pm.MessageType == pm.MSG_PART) {
                    // ada yang PART (keluar channel)
                    lSelectUser = (List) vNames.elementAt(index);
                    // cari.. hapus!
                    int j = lSelectUser.size() - 1;
                    for (int i = 0; i <= j; i++) {
                        if
(lSelectUser.getString(i).equals(pm.getSender())) {
                            lSelectUser.delete(i);
                        }
                    }
                }
                else if (pm.MessageType == pm.MSG_NAMES) {
                    // membuat daftar user untuk suatu channel
                    onNames(pm.getTarget(), pm.getMessage());
                }
            }
            return status;
        }

        /*
         * Proses menampilkan daftar user
         **/
        private void onNames(String channel, String names) {
            boolean foundInVector = false;
            // apakah vector "null"
            if (vNames == null) {
                vNames = new Vector();
            }
            // jika tidak "null", cari di vector
            else {
                // cari...
                if (findNames(channel) != -1) {
                    foundInVector = true;
                }
            }
            // jika pencarian gagal
            if (foundInVector == false) {
```

```
            // buat baru, tambahkan ke vector
            lSelectUser = createNames(channel, names);
            vNames.addElement(lSelectUser);
            //dChannel.setCurrent(lActiveDisplay);
        }
    }

    /*
     * Menampilkan pesan user yang diterima
     **/
    private void onUser() {
        String user =
lSelectUser.getString(lSelectUser.getSelectedIndex());
        // hilangkan karakter pertama, jika ada tanda
        // @ (operator) atau % (half-operator)
        if ((user.indexOf("@") > 0) || (user.indexOf("%") > 0)) {
            user = user.substring(1);
        }
        boolean foundInVector = false;
        // apakah vector "null"
        if (vUser == null) {
            vUser = new Vector();
        }
        // jika tidak "null", cari di vector
        else {
            // cari...
            if (findUser(user) != -1) {
                foundInVector = true;
            }
        }
        // jika pencarian gagal
        if (foundInVector == false) {
            // buat baru, tambahkan ke vector
            tbActiveDisplay = createUser(user);
            vUser.addElement(tbActiveDisplay);
            dChannelUser.setCurrent(tbActiveDisplay);
        }
        user = null;
    }

    /*
     * Menampilkan antarmuka untuk melakukan chat ke user lain
     * secara langsung
     **/
    public void showChatUser() {
        fChatUser = new Form("Chat To Another User");
        tfChatUser = new TextField("Nickname", "", 50,
TextField.ANY);
        fChatUser.append(tfChatUser);
        fChatUser.addCommand(new Command("Back", Command.BACK, 1));
        fChatUser.addCommand(new Command("Chat", Command.OK, 1));
        fChatUser.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    if (c.getLabel().equals("Back")) {
                        onStatus();
                    }
                    else if (c.getLabel().equals("Chat")) {
                        onChatUser();
                    }
                }
            }
```

```
        );
        dChannelUser.setCurrent(fChatUser);
    }

    /*
     * Menampilkan antarmuka untuk mengganti nickname
     **/
    public void showChangeNickname() {
        fChangeNickname = new Form("Change Nickname");
        tfChangeNickname = new TextField("New Nickname", "", 50,
TextField.ANY);
        fChangeNickname.append(tfChangeNickname);
        fChangeNickname.addCommand(new Command("Cancel",
Command.CANCEL, 1));
        fChangeNickname.addCommand(new Command("Change", Command.OK,
1));
        fChangeNickname.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    if (c.getLabel().equals("Cancel")) {
                        onStatus();
                    }
                    else if (c.getLabel().equals("Change")) {
                        onChangeNickname();
                    }
                }
            }
        );
        dChannelUser.setCurrent(fChangeNickname);
    }

    /*
     * Proses mengganti nickname
     **/
    private void onChangeNickname() {
        cChannelUser.ircNickname(tfChangeNickname.getString());
        onStatus();
    }

    /*
     * Proses chat ke user secara langsung
     **/
    private void onChatUser() {
        String user = tfChatUser.getString();
        // hilangkan karakter pertama, jika ada tanda
        // @ (operator) atau % (half-operator)
        if ((user.indexOf("@") > 0) || (user.indexOf("%") > 0)) {
            user = user.substring(1);
        }
        boolean foundInVector = false;
        // apakah vector "null"
        if (vUser == null) {
            vUser = new Vector();
        }
        // jika tidak "null", cari di vector
        else {
            // cari...
            if (findUser(user) != -1) {
                foundInVector = true;
            }
        }
        // jika pencarian gagal
```

```
        if (foundInVector == false) {
            // buat baru, tambahkan ke vector
            tbActiveDisplay = createUser(user);
            vUser.addElement(tbActiveDisplay);
            dChannelUser.setCurrent(tbActiveDisplay);
        }
    }

    /*
     * Proses menampilkan display target
     **/
    private void onDisplay() {
        String target =
lLastDisplay.getString(lLastDisplay.getSelectedIndex());
        boolean status = false;
        // cari di channel
        if (vChannel != null) {
            if (findChannel(target) != -1) {
                setChannel(findChannel(target));
                status = true;
            }
        }
        // cari di user
        if (vUser != null) {
            if ((findUser(target) != -1) && (status != true)) {
                setUser(findUser(target));
            }
        }
        target = null;
        System.gc();
    }

    /*
     * Mencari display yang berisi daftar user untuk suatu channel
     **/
    private int findNames(String channel) {
        if (channel.charAt(0) != '#') {
            channel = '#' + channel;
        }
        int min = 0;
        int max = vNames.size() - 1;
        int index = -1;
        for (int i = min; i <= max; i++) {
            lSelectUser = (List) vNames.elementAt(i);
            if (lSelectUser.getTitle().equals("Users on " +
channel)) {
                index = i;
                dChannelUser.setCurrent(lSelectUser);
                break;
            }
        }
        return index;
    }

    /*
     * Mencari display untuk user chat
     **/
    private int findUser(String user) {
        int min = 0;
        int max = -1;
        // apakah null?
        if (vUser != null) {
            max = vUser.size() - 1;
```

```
        }
        // cari index
        int index = -1;
        if (max >= 0) {
            tbUser = (TextBox) vUser.elementAt(0);
            if (tbUser.getTitle().equals(user)) {
                index = 0;
                dChannelUser.setCurrent(tbUser);
            }
        }
        else  {
            for (int i = min; i <= max; i++) {
                tbUser = (TextBox) vUser.elementAt(i);
                if (tbUser.getTitle().equals(user)) {
                    index = i;
                    dChannelUser.setCurrent(tbUser);
                    break;
                }
            }
        }
        return index;
    }

    /*
     * Membuat antarmuka untuk channel
     **/
    public TextBox createChannel(String channel) {
        channel = channel.toLowerCase();
        TextBox tb = new TextBox(channel, "", 1024, TextField.ANY);
        tb.addCommand(new Command("Send message", Command.ITEM, 1));
        tb.addCommand(new Command("Select user", Command.ITEM, 2));
        tb.addCommand(new Command("Exit channel", Command.ITEM, 3));
        tb.addCommand(new Command("Go to status", Command.BACK, 1));
        tb.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    if (c.getLabel().startsWith("Send message")) {
                        sendMessageChannel();
                    }
                    else if (c.getLabel().startsWith("Select user"))
{
                        selectUserOnChannel();
                    }
                    else if (c.getLabel().equals("Exit channel")) {
                        exitChannel();
                    }
                    else if (c.getLabel().equals("Go to status")) {
                        mChannelUser.showStatus();
                    }
                }
            } // end of "new CommandListener() {}"
        ); // end of "setCommandListener() {}"
        return tb;
    }

    /*
     * Memilih user di channel
     **/
    public void selectUserOnChannel() {
        int index = findNames(tbActiveDisplay.getTitle());
        if (index != -1) {
            tbLastChannel = tbActiveDisplay;
```

```
            setNames(index);
        }
    }

    /*
     * Membuat daftar user untuk suatu channel
     **/
    public List createNames(String channel, String names) {
        final String whitespace = " ";
        int usercount = new IrcFunction().getParsedCount(names,
whitespace);
        String usernickname = "";

        List l = new List("Users on " + channel, Choice.EXCLUSIVE);
        for (int i = 1; i <= usercount; i++) {
            usernickname = new IrcFunction().getParsedString(names,
whitespace,  i);
            l.append(usernickname, null);
        }

        l.addCommand(new Command("Back", Command.BACK, 1));
        l.addCommand(new Command("Select", Command.OK, 2));
        l.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    if (c.getLabel().equals("Back")) {
                        dChannelUser.setCurrent(tbLastDisplay);
                    }
                    else if (c.getLabel().equals("Select")) {
                        lLastSelectUser = (List)
dChannelUser.getCurrent();
                        onUser();
                    }
                }
            } // end of "new CommandListener() {}"
        ); // end of "setCommandListener() {}"

        usernickname = null;
        System.gc();
        return l;
    }

    /*
     * Membuat antarmuka untuk chat user
     **/
    public TextBox createUser(String user) {
        // hilangkan tanda operator (@ atau %)
        String targetuser = user;
        if (targetuser.charAt(0) == '@') {
            targetuser = targetuser.substring(1);
        }
        else if (targetuser.charAt(0) == '%') {
            targetuser = targetuser.substring(1);
        }
        //
        TextBox tb = new TextBox(targetuser, "", 1024,
 TextField.ANY);
        tb.addCommand(new Command("Go to status", Command.BACK, 1));
        tb.addCommand(new Command("Send message", Command.ITEM, 1));
        tb.addCommand(new Command("Close chat", Command.ITEM, 2));
        tb.setCommandListener(
            new CommandListener() {
```

```
                public void commandAction(Command c, Displayable d)
{
                    if (c.getLabel().startsWith("Send message")) {
                        sendMessageUser();
                    }
                    else if (c.getLabel().equals("Go to status")) {
                        mChannelUser.showStatus();
                    }
                    else if (c.getLabel().equals("Close chat")) {
                        exitUser();
                    }
                }
            } // end of "new CommandListener() {}"
        ); // end of "setCommandListener() {}"
        return tb;
    }

    /*
     * Membuat antarmuka untuk mengirim pesan ke channel
     **/
    private void sendMessageChannel() {
        TextBox tb = new TextBox("Message to " +
tbActiveDisplay.getTitle(),"", 1024, TextField.ANY);
        tb.addCommand(new Command("Back", Command.BACK, 1));
        tb.addCommand(new Command("Send", Command.OK, 2));
        tb.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    if (c.getLabel().startsWith("Send")) {
                        sendMessageChannelDone();
                    }
                    else if (c.getLabel().equals("Back")) {
                        dChannelUser.setCurrent(tbLastDisplay);
                        tbActiveDisplay = tbLastDisplay;
                    }
                }
            } // end of "new CommandListener() {}"
        ); // end of "setCommandListener() {}"
        tbLastChannel = tbActiveDisplay;
        tbActiveDisplay = tb;
        dChannelUser.setCurrent(tb);
    }

    /*
     * Membuat antarmuka untuk mengirim pesan ke user
     **/
    private void sendMessageUser() {
        TextBox tb = new TextBox("Message to " +
tbActiveDisplay.getTitle(),"", 1024, TextField.ANY);
        tb.addCommand(new Command("Back", Command.BACK, 1));
        tb.addCommand(new Command("Send", Command.OK, 2));
        tb.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    if (c.getLabel().startsWith("Send")) {
                        sendMessageUserDone();
                    }
                    else if (c.getLabel().equals("Back")) {
                        dChannelUser.setCurrent(tbLastDisplay);
                        tbActiveDisplay = tbLastDisplay;
                    }
```

```
                    else if (c.getLabel().equals("Go to status")) {
                        mChannelUser.showStatus();
                    }
                }
            } // end of "new CommandListener() {}"
        ); // end of "setCommandListener() {}"
        tbLastUser = tbActiveDisplay;
        tbActiveDisplay = tb;
        dChannelUser.setCurrent(tb);
    }

    /*
     * Proses mengirim pesan ke user
     **/
    private void sendMessageUserDone() {
        // hilangkan tanda operator (@ atau %)
        String targetuser = tbLastUser.getTitle();
        if (targetuser.charAt(0) == '@') {
            targetuser = targetuser.substring(1);
        }
        else if (targetuser.charAt(0) == '%') {
            targetuser = targetuser.substring(1);
        }
        // kirimkan pesan ke user
        cChannelUser.ircSendMessageToUser(targetuser,
tbActiveDisplay.getString());
        // tambahkan pesan di user window yang aktif
        new IrcFunction().appendTextBox(tbLastUser, "<" +
cChannelUser.getNickname() +
                "> " + tbActiveDisplay.getString());
        dChannelUser.setCurrent(tbLastUser);
        tbActiveDisplay = tbLastUser;
    }

    /*
     * Keluar dari channel
     **/
    private void exitChannel() {
        String channel = tbActiveDisplay.getTitle();
        TextBox display;
        int index;
        // keluar dari channel
        cChannelUser.ircPart(channel);
        // hapus dari vector
        index = findChannel(channel);
        display = getChannel(index);
        vChannel.removeElementAt(index);
        channel = null;
        display = null;
        System.gc();
        // ke status window
        onStatus();
    }

    /*
     * Menutup sesi chat dengan user lain
     **/
    private void exitUser() {
        String nickname = tbActiveDisplay.getTitle();
        TextBox display;
        int index;
        // hapus dari vector
        index = findUser(nickname);
```

```
            display = getUser(index);
            vUser.removeElementAt(index);
            nickname = null;
            display = null;
            System.gc();
            // ke status window
            onStatus();
        }

        /*
         * Proses mengirim pesan ke channel
         **/
        private void sendMessageChannelDone() {
            // kirimkan pesan ke channel

cChannelUser.ircSendMessageToChannel(tbLastChannel.getTitle(),
tbActiveDisplay.getString());
            // tambahkan pesan di channel window yang aktif
            new IrcFunction().appendTextBox(tbLastChannel, "<" +
cChannelUser.getNickname() +
                    "> " + tbActiveDisplay.getString());
            dChannelUser.setCurrent(tbLastChannel);
            tbActiveDisplay = tbLastChannel;
        }
}
```

IrcConnection.java

```
/* Nama file:   IrcConnection.java
 * Author:      (c) 2005 Novareza Klifartha
 **/

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.*;

public class IrcConnection {
    // midlet
    private MIDlet mConnection;
    // display
    private Display dConnection;
    // socket
    private IrcSocket sConnection;
    // form
    private Form fConnect;
    private TextField tfHostname, tfUsername, tfNickname,
tfRealname;
    // event handler tombol
    private Command cConnect, cExit;
    // thread
    private Thread tConnection;
    // status koneksi
    private boolean connected = false;
    private boolean namevalid = false;
    private String hostname, username, nickname, realname,
lastnickname;
    // untuk menampilkan status koneksi
    private TextBox logstatus;
    // list vektor untuk menyimpan antrian message
    private Vector vQueue;
    // IRC message
```

```
    public IrcMessage pmConnection;
    // versi aplikasi
    public final String VERSION = "J2ME IRC Client by Novareza
Klifartha";
    // karakter khusus untuk melapisi protokol
    public final char magicChar = '¤';
    // state
    private int currentState;
    public final int STATE_NONE        = 0;
    public final int STATE_REGISTER    = 1;
    public final int STATE_MOTD        = 2;
    public final int STATE_NICKNAME    = 3;
    public final int STATE_JOIN        = 4;
    public final int STATE_NAMES       = 5;
    public final int STATE_CLOSINGLINK = 6;
    // error state
    private int currentError;
    public final int ERR_NONE      = 0;
    public final int ERR_REGISTER  = -1;
    public final int ERR_NICKNAME  = -2;
    public final int ERR_JOIN      = -3;
    public final int ERR_PART      = -4;
    public final int ERR_NAMES     = -5;
    public final int ERR_PRIVMSG   = -6;

    /*
     * Constructor
     **/
    public IrcConnection(MIDlet m) {
        // siapkan display
        this.mConnection = m;
        this.dConnection = Display.getDisplay(m);
        // siapkan queue
        this.vQueue = new Vector();
        // rancangan form
        fConnect = new Form("Connect");
        tfHostname = new TextField("IRC Server", "localhost", 24,
TextField.ANY);
        tfUsername = new TextField("Username", "myname", 24,
TextField.ANY);
        tfNickname = new TextField("Nickname", "myname", 24,
TextField.ANY);
        tfRealname = new TextField("Realname", "myname", 24,
TextField.ANY);
        cConnect = new Command("Connect", Command.OK, 1);
        cExit = new Command("Exit", Command.EXIT, 2);
        fConnect.append(tfHostname);
        fConnect.append(tfUsername);
        fConnect.append(tfNickname);
        fConnect.append(tfRealname);
        fConnect.addCommand(cConnect);
        fConnect.addCommand(cExit);
        // siapkan event handler
        fConnect.setCommandListener(
                new CommandListener() {
                    public void commandAction(Command c, Displayable
d) {
                        if (c == cConnect) {
                            // ketika tombol Connect dipilih
                            onConnect();
                        }
                        else if (c == cExit) {
                            // ketika tombol Exit dipilih
```

```
                        onExit();
                    }
                }
            }
        );
        // aktifkan display
        dConnection.setCurrent(fConnect);
        // tahan hingga koneksi berhasil
        while (connected == false) {}
    }

    private boolean isValidName(String name) {
        boolean status = false;
        String s = name.toUpperCase();
        int ch = (int) s.charAt(0);
        if ((ch > 64) && (ch < 91)) {
            status = true;
        }
        return status;
    }

    private void onClosingLink() {
        currentState = STATE_CLOSINGLINK;
    }

    private void onConnect() {
        // cek user/nick
        namevalid = true;
        if (!isValidName(tfUsername.getString())) namevalid = false;
        if (!isValidName(tfNickname.getString())) namevalid = false;
        if (namevalid == false) {
            new IrcFunction().msgError(dConnection, "Invalid
username or nickname!");
        }
        // jalankan hanya jika belum terkoneksi
        if ((namevalid == true) && (connected == false)) {
            // ciptakan thread untuk koneksi
            tConnection = new Thread() {
                public void run() {
                    // connecting...
                    dConnection.setCurrent(
                            new TextBox("Connecting...", "", 1,
TextField.ANY)
                    );
                    // buat socket baru
                    sConnection = new
IrcSocket(tfHostname.getString());
                    // tunggu sampai koneksi selesai
                    while (sConnection.isConnected() == false) {
                        // hentikan menunggu jika ternyata error
                        if (sConnection.isError() == true) {
                            break;
                        }
                    }
                    // jika error, maka koneksi gagal
                    if (sConnection.isError() == false) {
                        new IrcFunction().msgInfo(dConnection,
"Connected!");
                        connected = true;
                        hostname = tfHostname.getString();
                        username = tfUsername.getString();
                        nickname = tfNickname.getString();
                        realname = tfRealname.getString();
```

```
                    }
                    else {
                        new IrcFunction().msgError(dConnection,
"Connection failed!");
                        sConnection.close();
                        new IrcFunction().delay(1000);
                        dConnection.setCurrent(fConnect);
                    }
                }
            };
            // jalankan thread di atas
            tConnection.start();
        }
    }

    /*
     * Keluar aplikasi
     **/
    private void onExit() {
        mConnection.notifyDestroyed();
    }

    /*
     * Memeriksa koneksi
     **/
    public boolean isConnected() {
        // baca status koneksi
        return connected;
    }

    /*
     * Menutup koneksi
     **/
    public void close() {
        if (connected == true) {
            sConnection.close();
            connected = false;
        }
    }

    /*
     * Meregister user
     **/
    public void ircRegister() {
        if (connected == true) {
            currentState = STATE_REGISTER;
            currentError = ERR_NONE;
            // register user
            sConnection.writeln("USER " + username + " " +
"localhost" +
                    " " + hostname + " :" + realname);
            sConnection.writeln("NICK " + nickname);
        }
    }

    /*
     * Membaca MOTD (Message of The Day)
     **/
    public void ircReadMOTD(TextBox log) {
        if (connected == true) {
            currentState = STATE_MOTD;
            logstatus = log;
            // baca MOTD hingga habis
```

```
            String motd = "";
            String pong = "";
            while (motd.indexOf("376") == -1) { // End of /MOTD
                // baca per baris
                motd = sConnection.readln();
                // beberapa server meminta balasan "PING"
                // sebelum menampilkan MOTD
                if (motd.startsWith("PING")) {
                    pong = "PONG " + motd.substring(motd.indexOf("
") + 1);
                    sConnection.writeln(pong);
                }
                // log to TextBox
                int len = log.getString().length() + motd.length();
                if (len > log.getMaxSize()) {
                    // hapus bagian awal
                    int start = len - log.getMaxSize() + 1;
                    log.setString(log.getString().substring(start));
                }
                // tambahkan di bagian akhir
                log.insert("\n" + motd, log.getString().length() +
1);
            }
            currentState = STATE_NONE;
        }
    }

    /*
     * Menentukan nickname
     **/
    public boolean ircNickname(String nickname) {
        boolean status = false;
        if (connected == true) {
            currentState = STATE_NICKNAME;
            sConnection.writeln("NICK " + nickname);
            this.lastnickname = this.nickname;
            this.nickname = nickname;
            status = true;
        }
        return status;
    }

    /*
     * Join ke channel
     **/
    public boolean ircJoin(String channel, String key) {
        boolean status = false;
        if (connected == true) {
            currentState = STATE_JOIN;
            // fix channel format
            if (channel.charAt(0) != '#') {
                channel = '#' + channel;
            }
            // join ke channel
            if (key.equals("")) {
                sConnection.writeln("JOIN " + channel);
            }
            else {
                sConnection.writeln("JOIN " + channel + " " + key);
            }
            status = true;
        }
        return status;
```

```
    }

    /*
     * Menampilkan user-user di suatu channel
     **/
    public void ircNames(String channel) {
        if (connected == true) {
            sConnection.writeln("NAMES " + channel);
        }
    }

    /*
     * Mengirim pesan ke channel
     **/
    public void ircSendMessageToChannel(String channel, String
message) {
        if (connected == true) {
            sConnection.writeln("PRIVMSG " + channel + " :" +
message);
        }
    }

    /*
     * Mengirim pesan ke user
     **/
    public void ircSendMessageToUser(String user, String message) {
        if (connected == true) {
            sConnection.writeln("PRIVMSG " + user + " :" + message);
        }
    }

    /*
     * Keluar dari channel
     **/
    public void ircPart(String channel) {
        if (connected == true) {
            sConnection.writeln("PART " + channel);
        }
    }

    /*
     * Keluar dari jaringan IRC
     **/
    public void ircQuit(String quitmsg) {
        if (connected == true) {
            sConnection.writeln("QUIT :" + quitmsg);
        }
    }

    /*
     * Mengaktifkan thread untuk pembacaan data dari socket
     **/
    public void startListener() {
        Thread listen = new Thread() {
            public void run() {
                while (true) {
                    int dt = 0;
                    // aktifkan magic char
                    if (sConnection.magicCharActive == false) {
                        // tulis magic char ke socket
                        sConnection.writeln("PRIVMSG "
                        + nickname + " :" + sConnection.magicChar);
                        sConnection.magicCharActive = true;
```

```
                        // tunggu agak lama, biar data di buffer
lumayan banyak
                        dt = 100;
                    }
                    else {
                        dt = 500;
                    }
                    // baca data dari socket, pahami protokolnya
                    decodeIrcProtocol(sConnection.readln());
                    // sleep
                    try {
                        Thread.sleep(dt);
                    }
                    catch (InterruptedException ie) {}
                }
            }
        };
        listen.start();
    }

    /*
     * Mengetahui kondisi user terakhir
     **/
    public int getLastState() {
        return currentState;
    }

    /*
     * Mengetahui error terakhir
     **/
    public int getLastError() {
        return currentError;
    }

    /*
     * Incoming PING request
     **/
    private void onPing(String rawdata) {
        String pingReply;
        pingReply = "PONG " + rawdata.substring(rawdata.indexOf(" ")
+ 1);
        sConnection.writeln(pingReply);
    }

    /*
     * Aktivitas join di channel
     **/
    private void onJoin(String rawdata) {
        String senderNick, channel;
            // get nickname who joins
            senderNick = rawdata.substring(1, rawdata.indexOf("!"));
            // get room
            channel = rawdata.substring(rawdata.indexOf("JOIN"));
            channel = channel.substring(channel.indexOf(":") + 1);
            channel = channel.substring(0, channel.length() - 1);
            pmConnection = new IrcMessage(senderNick, channel, "");
            pmConnection.MessageType = pmConnection.MSG_JOIN;
            vQueue.addElement(pmConnection);
    }

    /*
     * Aktivitas keluar channel
     **/
```

```
    private void onPart(String rawdata) {
        String senderNick, channel;
            // get nickname who parts
            senderNick = rawdata.substring(1, rawdata.indexOf("!"));
            // get room
            channel = rawdata.substring(rawdata.indexOf("PART"));
            channel = channel.substring(channel.indexOf("#"));
            channel = channel.substring(0, channel.length() - 1);
            pmConnection = new IrcMessage(senderNick, channel, "");
            pmConnection.MessageType = pmConnection.MSG_PART;
            vQueue.addElement(pmConnection);
    }

    /*
     * Informasi daftar nickname di suatu channel
     **/
    private void onNames(String rawdata) {
        currentError = ERR_NONE;
        currentState = STATE_NAMES;
        String channel, names;
        channel = rawdata.substring(rawdata.indexOf("= ") + 2,
rawdata.indexOf(" :"));
        // all nickname on channel
        names = rawdata.substring(rawdata.indexOf(" :") + 2);
        pmConnection = new IrcMessage("", channel, names);
        pmConnection.MessageType = pmConnection.MSG_NAMES;
        vQueue.addElement(pmConnection);
        currentState = STATE_NONE;
    }

    /*
     * Mendeteksi pesan masuk dari channel/user
     **/
    private void onMessages(String rawdata) {
        final char noticeChar = ' ';
        String senderNick, target, message, pingID,
               notice, noticeType, noticeReply;

               // get sender nickname
               senderNick = rawdata.substring(1,
rawdata.indexOf("!"));
               // indentify target
               target = rawdata.substring(rawdata.indexOf(" ") +
1); // spasi ke-1 = "PRIVMSG..."
               target = target.substring(target.indexOf(" ") + 1);
// spasi ke-2 = "<target>..."
               target = target.substring(0, target.indexOf(" "));
               // read message
               message =
rawdata.substring(rawdata.indexOf("PRIVMSG"));
               message = message.substring(message.indexOf(":") +
1);
               // mIRC "NOTICE"
               if (message.charAt(0) != magicChar) {
                   if (message.charAt(0) == noticeChar) {
                       notice = message.substring(1,
message.length() - 2);

                       if (notice.indexOf(" ") != -1) {
                           noticeType = notice.substring(0,
notice.indexOf(" "));
                       }
                       else {
```

```
                            noticeType = notice;
                        }

                        if (noticeType.equals("PING")) {
                            pingID =
notice.substring(notice.indexOf(" ") + 1);
                            noticeReply = "NOTICE " + senderNick + "
:" +
                                noticeChar + "PING " + pingID +
noticeChar + "\n";
                            sConnection.writeln(noticeReply);
                        }
                        else if (noticeType.equals("TIME")) {
                            noticeReply = "NOTICE " + senderNick + "
:" +
                                noticeChar + "TIME " + "no-time-here
;o" + noticeChar + "\n";
                            sConnection.writeln(noticeReply);
                        }
                        else if (noticeType.equals("VERSION")) {
                            noticeReply = "NOTICE " + senderNick + "
:" +
                                noticeChar + "VERSION " +
this.VERSION + noticeChar + "\n";
                            sConnection.writeln(noticeReply);
                        }
                    }
                    // bukan mIRC "NOTICE", masukkan ke queue
                    else {
                        pmConnection = new IrcMessage(senderNick,
target, message);
                        pmConnection.MessageType =
pmConnection.MSG_PRIVMSG;
                        vQueue.addElement(pmConnection);
                    }
                }
    }

    /*
     * Melakukan decode protokol
     **/
    private void decodeIrcProtocol(String rawdata) {
        // server melakukan "PING" ke client
        if (rawdata.startsWith("PING")) {
            onPing(rawdata);
        }
        // JOIN
        else if (rawdata.indexOf("JOIN") > 0) {
            onJoin(rawdata);
        }
        // PART
        else if (rawdata.indexOf("PART") > 0) {
            onPart(rawdata);
        }
        // NAMES
        else if (rawdata.indexOf("353") > 0) {
            onNames(rawdata);
        }
        // PRIVMSG
        else if (rawdata.indexOf("PRIVMSG") > 0) {
            onMessages(rawdata);
        }
        // Closing Link
```

```
        else if (rawdata.indexOf("Closing Link") > 0) {
            onClosingLink();
        }
        // Error on registering
        else if (rawdata.indexOf("451") > 0) {
            new IrcFunction().msgError(dConnection, "Registration
failed!");
            currentError = ERR_REGISTER;
        }
        // Error on setting nickname: "Erroneus nickname"
        else if (rawdata.indexOf("432") > 0) {
            new IrcFunction().msgError(dConnection, "FAILED:
Erroneus nickname!");
            currentError = ERR_NICKNAME;
            this.nickname = this.lastnickname;
        }
        // Error on setting nickname: "Nickname is already in use"
        else if (rawdata.indexOf("433") > 0) {
            new IrcFunction().msgError(dConnection, "FAILED:
Nickname is already in use!");
            currentError = ERR_NICKNAME;
            this.nickname = this.lastnickname;
        }
        // Register done, start MOTD
        else if (rawdata.indexOf("001") > 0) {
            new IrcFunction().msgInfo(dConnection, "Registration
success!");
            currentError = ERR_NONE;
            currentState = STATE_MOTD;
        }
        // No such nick/channel
        else if (rawdata.indexOf("401") > 0) {
            currentError = ERR_PRIVMSG;
            new IrcFunction().msgError(dConnection, "No such
nick/channel");
        }
        // Cannot join channel (+k)
        else if (rawdata.indexOf("475") > 0) {
            currentError = ERR_JOIN;
            new IrcFunction().msgError(dConnection, "Channel key
required to join!");
        }

        // log to TextBox
        int len = logstatus.getString().length() + rawdata.length();
        if (len > logstatus.getMaxSize()) {
            // hapus bagian awal
            int start = len - logstatus.getMaxSize() + 1;

logstatus.setString(logstatus.getString().substring(start));
        }
        // tambahkan di bagian akhir
        if (!rawdata.endsWith(magicChar + "")) {
            logstatus.insert("\n" + rawdata,
logstatus.getString().length() + 1);
        }
    }

    /*
     * Membaca antrian pesan masuk untuk diproses ke tampilan
     **/
    public void readMessageQueue() {
        pmConnection = null;
```

```
        if (vQueue.size() != 0) {
            pmConnection = (IrcMessage) vQueue.elementAt(0);
            vQueue.removeElementAt(0);
        }
    }

    /*
     * Mengetahui server yang digunakan
     **/
    public String getHostname() {
        return this.hostname;
    }

    /*
     * Mengetahui nickname yang digunakan
     **/
    public String getUsername() {
        return this.username;
    }

    /*
     * Mengetahui nickname yang digunakan
     **/
    public String getNickname() {
        return this.nickname;
    }

    /*
     * Mengetahui realname yang digunakan
     **/
    public String getRealname() {
        return this.realname;
    }

    /*
     * Mengirim data ke socket per baris
     **/
    public void ircSendRawData(String rawdata) {
        if (connected == true) {
            sConnection.writeln(rawdata);
        }
    }

    /*
     * Membaca data dari socket per baris
     **/
    public String ircReadRawData() {
        String s = "";
        if (connected == true) {
            s = sConnection.readln();
        }
        return s;
    }
}
```

IrcFunction.java

```
/* Nama file:    IrcFunction.java
 * Author:       (c) 2005 Novareza Klifartha
 **/

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.*;

public class IrcFunction {
    public IrcFunction() {
        // do nothing
    }

    /*
     * Menampilkan pesan error ke display
     **/
    public void msgError(Display d, String msg) {
        d.setCurrent(new       Alert("Error",       msg,       null,
AlertType.ERROR));
    }

    /*
     * Menampilkan pesan info ke display
     **/
    public void msgInfo(Display d, String msg) {
        d.setCurrent(new Alert("Info", msg, null, AlertType.INFO));
    }

    /*
     * Menunggu selama selang waktu tertentu
     **/
    public void delay(int miliseconds) {
        try {
            Thread.sleep(miliseconds);
        }
        catch (InterruptedException ie) {}
    }

    /*
     * Melakukan parsing data
     **/
    public String getParsedString(String data, String delimiter, int
 index) {
        String s = data;

        if (index > 0) {
            int inc = 1;
            int begin = 0;
            int end = getParsedCount(data, delimiter);

            while ((inc != index) && (inc < end)) {
                inc++;
                s = s.substring(s.indexOf(delimiter) + 1);
            }
            // jika substring yang dicari BUKAN di index terakhir
            if (inc != end) {
                s = s.substring(0, s.indexOf(delimiter) + 1);
            }
            //System.out.println("data[" + index + "] = " + s);
        }
```

```
        return s;
    }

    /*
     * Menghitung jumlah indeks dari data yang di-parsing
     **/
    public int getParsedCount(String data, String delimiter) {
        int count = 0;
        String s = null;
        // ulangi hingga delimiter tidak ditemukan
        s = data;
        while (s.indexOf(delimiter) != -1) {
            count++;
            s = s.substring(s.indexOf(delimiter) + 1);
        };
        count++;
        return count;
    }

    /*
     * Menambahkan isi data di TextBox
     **/
    public void appendTextBox(TextBox tb, String s) {
        if ( (tb != null) && (!s.equals("")) && (!s.equals("\n")) )
{
            int len = tb.getString().length() + s.length();
            if (len > tb.getMaxSize()) {
                // hapus bagian awal
                int start = len - tb.getMaxSize() + 1;
                tb.setString(tb.getString().substring(start));
            }
            // tambahkan di bagian akhir
            if (tb.getString().length() == 0) {
                tb.setString(s);
            }
            else {
                tb.insert("\n" + s, tb.getString().length() + 1);
            }
        }
    }
}
```

IrcMain.java

```
/* Nama file:    IrcMain.java
 * Author:       (c) 2005 Novareza Klifartha
 **/

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class IrcMain {
    // midlet
    private MIDlet mMain;
    // display
    private Display dMain;
    private IrcConnection cMain;
    private IrcChannelUser chMain;
    // text box
    private TextBox tbStatus, tbExpertMode;
```

```
    // text field
    private TextField tfSendMsg;
    // event handler tombol
    private Command cBack, cQuit;
    private Command cClearStatus, cExpertMode, cExpertModeSend,
            cJoinChannel, cChatUser, cChangeNickname, cShowDisplay;
    // thread utama
    private Thread tMain;

    /*
     * Constructor
     **/
    public IrcMain(MIDlet m, IrcConnection c) {
        // pemindahan kendali variabel
        this.mMain = m;
        this.dMain = Display.getDisplay(m);
        this.cMain = c;
        // pengaktifan dynamic channel/user window
        chMain = new IrcChannelUser(dMain, cMain, this);
        // pembuatan window-window yang diperlukan
        createStatusWindow();
        createExpertModeWindow();
        // pengaktifan display ke status window
        dMain.setCurrent(tbStatus);
        // registering...
        appendStatus("+ Registering...");
        cMain.ircRegister();
        // baca MOTD (Message of The Day)
        appendStatus("+ Getting MOTD...");
        cMain.ircReadMOTD(tbStatus);
        // pengaktifan thread socket listener
        cMain.startListener();
        // pengaktifan thread Main
        appendStatus("\n+ Ready to chat on the network.");
        startMain();
        // change status title [<nickname> on <IRC server>]
        tbStatus.setTitle("Status [" + cMain.getNickname() +
            " on " + cMain.getHostname() + "]");
    }

    /*
     * Mengaktifkan thread utama
     **/
    public void startMain() {
        Thread main = new Thread() {
            public void run() {
                while (true) {
                    // memproses interface window sesuai protokolnya
                    cMain.readMessageQueue();
                    if (cMain.pmConnection != null) {

chMain.showDisplayToTarget(cMain.pmConnection);
                    }
                    // periksa ClosingLink
                    if (cMain.getLastState() ==
cMain.STATE_CLOSINGLINK) {
                        cMain.close();
                    }
                    // update nickname
                    String newtitle = "Status [" +
cMain.getNickname() +
                        " on " + cMain.getHostname() + "]";
                    if (!tbStatus.getTitle().equals(newtitle)) {
```

```
                    tbStatus.setTitle(newtitle);
                }
                newtitle = null;
                // sleep
                try {
                    Thread.sleep(1000);
                }
                catch (InterruptedException ie) {}
                // run garbage collector
                System.gc();
            }
        }
    };
    main.start();
}

/*
 * Pembuatan status window
 **/
private void createStatusWindow() {
    // pembuatan command
    cJoinChannel = new Command("Join channel", Command.ITEM, 1);
    cChatUser = new Command("Chat user", Command.ITEM, 1);
    cChangeNickname = new Command("Change nickname",
Command.ITEM, 1);
    cShowDisplay = new Command("Display list", Command.ITEM, 2);
    cExpertMode = new Command("Expert mode", Command.ITEM, 3);
    cClearStatus = new Command("Clear status", Command.ITEM, 4);
    cQuit = new Command("Quit", Command.EXIT, 1);
    // pembuatan tampilan "status"
    tbStatus = new TextBox("Status", "", 1024, TextField.ANY);
    tbStatus.addCommand(cJoinChannel);
    tbStatus.addCommand(cChatUser);
    tbStatus.addCommand(cChangeNickname);
    tbStatus.addCommand(cShowDisplay);
    tbStatus.addCommand(cExpertMode);
    tbStatus.addCommand(cClearStatus);
    tbStatus.addCommand(cQuit);
    // pengaktifan command listener
    tbStatus.setCommandListener(
        new CommandListener() {
            public void commandAction(Command c, Displayable d)
{
                if (c == cClearStatus) {
                    tbStatus.setString("");
                }
                else if (c == cJoinChannel) {
                    onJoin();
                }
                else if (c == cChatUser) {
                    onChatUser();
                }
                else if (c == cChangeNickname) {
                    onChangeNickname();
                }
                else if (c == cShowDisplay) {
                    onShowDisplay();
                }
                else if (c == cExpertMode) {
                    onExpertMode();
                }
                else if (c == cQuit) {
```

```
                        cMain.ircQuit("J2ME IRC Client:
bye..bye...");
                        mMain.notifyDestroyed();
                    }
                }
            }
        );
    }

    /*
     * Pembuatan Expert Mode window
     **/
    private void createExpertModeWindow() {
        // pembuatan command
        cExpertModeSend = new Command("Send", Command.OK, 2);
        cBack = new Command("Back", Command.BACK, 1);
        // pembuatan tampilan "write status"
        tbExpertMode = new TextBox("Expert mode", "", 255,
TextField.ANY);
        tbExpertMode.addCommand(cExpertModeSend);
        tbExpertMode.addCommand(cBack);
        tbExpertMode.setCommandListener(
            new CommandListener() {
                public void commandAction(Command c, Displayable d)
{
                    // tombol "Send raw data"
                    if (c == cExpertModeSend) {
                        // jika tidak kosong
                        if (!tbExpertMode.getString().equals("")) {

cMain.ircSendRawData(tbExpertMode.getString());
                            tbExpertMode.setString("");
                            showStatus();
                        }
                    }
                    // tombol "Back"
                    else if (c == cBack) {
                        showStatus();
                    }
                }
            }
        );
    }

    /*
     * Menampilkan antarmuka Expert Mode
     **/
    private void onExpertMode() {
        dMain.setCurrent(tbExpertMode);
    }

    /*
     * Menampilkan antarmuka Chat User
     **/
    private void onChatUser() {
        chMain.showChatUser();
    }

    /*
     * Menampilkan antarmuka untuk penggantian nickname
     **/
    private void onChangeNickname() {
        chMain.showChangeNickname();
```

```
    }

    /*
     * Menampilkan antarmuka Display List
     **/
    private void onShowDisplay() {
        chMain.showDisplay();
    }

    /*
     * Menampilkan antarmuka untuk Join
     **/
    private void onJoin() {
        chMain.showJoin();
    }

    /*
     * Menampilkan antarmuka Status
     **/
    public void showStatus() {
        dMain.setCurrent(tbStatus);
    }

    /*
     * Menambahkan data ke Status
     **/
    private void appendStatus(String s) {
        if (tbStatus != null) {
            int len = tbStatus.getString().length() + s.length();
            if (len > tbStatus.getMaxSize()) {
                // hapus bagian awal
                int start = len - tbStatus.getMaxSize() + 1;

tbStatus.setString(tbStatus.getString().substring(start));
            }
            // tambahkan di bagian akhir
            if (tbStatus.getString().length() == 0) {
                tbStatus.setString(s);
            }
            else {
                tbStatus.insert("\n" + s,
tbStatus.getString().length() + 1);
            }
        }
    }
}
```

IrcMessage.java

```
/* Nama file:    IrcMessage.java
 * Author:       (c) 2005 Novareza Klifartha
 **/
public class IrcMessage {
    // data-data message
    private String from, to, message;
    // konstanta message
    public final int MSG_NONE = -1;
    public final int MSG_PING = 0;
    public final int MSG_PRIVMSG = 1;
    public final int MSG_NOTICE = 2;
    public final int MSG_JOIN = 3;
    public final int MSG_NAMES = 4;
    public final int MSG_PART = 5;
    public final int MSG_QUIT = 6;
    // tipe message
    public int MessageType = MSG_NONE;

    /*
     * Constructor dengan parameter
     **/
    public IrcMessage(String from, String to, String message) {
        this.from = from;
        this.to = to;
        this.message = message;
    }

    /*
     * Constructor tanpa parameter
     **/
    public IrcMessage() {
        this.from = null;
        this.to = null;
        this.message = null;
    }

    /*
     * Menulis satu baris data ke socket
     **/
    public void setData(String from, String to, String message) {
        this.from = from;
        this.to = to;
        this.message = message;
    }

    /*
     * Mengetahui pengirim message
     **/
    public String getSender() {
        return from;
    }

    /*
     * Mengetahui target
     **/
    public String getTarget() {
        return to;
    }

    /*
```

```
     * Membaca message
     **/
    public String getMessage() {
        return message;
    }
}
```

IrcSocket.java

```
/* Nama file:    IrcSocket.java
 * Author:      (c) 2005 Novareza Klifartha
 **/

import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import java.io.*;
import java.util.*;

public class IrcSocket {
    // url untuk koneksi
    private String url;
    // port IRC
    private final String port = "6667";
    // koneksi stream
    private StreamConnection socket;
    // input
    private InputStream in;
    // output
    private OutputStream out;
    // untuk menampung data komunikasi
    private String request, response;
    // status koneksi
    private boolean connected = false;
    // status error
    private boolean error = false;
    // karakter untuk membatasi akhir data input
    public final char magicChar = '¤';
    // status pemakaian magicChar
    public boolean magicCharActive = false;

    public IrcSocket(String host) {
        try {
            url = "socket://" + host + ":" + port;
            socket = (StreamConnection) Connector.open(url,
Connector.READ_WRITE);
            in = socket.openInputStream();
            out = socket.openOutputStream();
            connected = true;
        }
        catch (IOException ie) {
            error = true;
        }
    }

    /*
     * Membaca data input yang masuk ke socket
     **/
    private String readInputStream() {
  int ch  = 0;
```

```
java.util.Vector respon = new Vector();
StringBuffer sb = new StringBuffer();
int idx = 0;
while ((ch != -1) && ((char)ch != magicChar))  {
          // baca per byte
          try {
              ch = in.read();
          }
          catch (IOException ie) {}
          // baca perpindahan baris
          if ((char)ch == '\n'){
              respon.addElement(sb.toString());
              break;
          }
          // magic char
          if ((char)ch == magicChar) {
              magicCharActive = false;
          }
          sb.append((char)ch);
          idx++;
}
String dataIn = sb.toString();
return dataIn;
   }

   /*
    * Menulis satu baris data ke socket
    **/
   public void writeln(String data) {
       try {
           request = data + "\n";
           out.write(request.getBytes());
           out.flush();
       }
       catch (IOException ie) {}
   }

   /*
    * Membaca satu baris data dari socket
    **/
   public String readln() {
       String data = readInputStream();
       return data;
   }

   /*
    * Menutup socket
    **/
   public void close() {
       try {
           if (in != null) {
               in.close();
           }
           if (out != null) {
               out.close();
           }
           if (socket != null) {
               socket.close();
           }
       }
       catch (IOException ie) {}
   }
```

```
    /*
     * Memeriksa status koneksi socket
     **/
    public boolean isConnected() {
        return connected;
    }

    /*
     * Memeriksa status kesalahan
     **/
    public boolean isError() {
        return error;
    }
}
```