PROGRAM CHAT BERBASIS JAVA

Ibnu Pradipta – 33237
Hanindito H.P – 33308
Firman Nanda – 33529
Jurusan Teknik Elektro FT UGM,
Yogyakarta

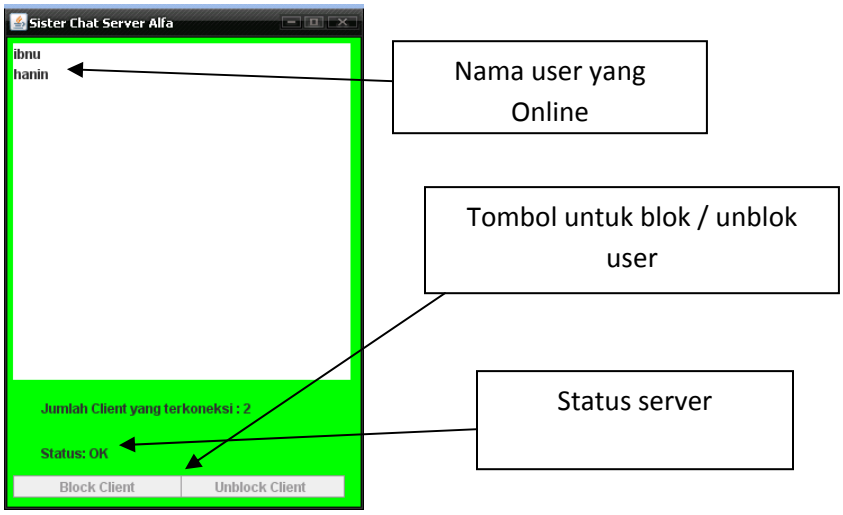
1.1 PENDAHULUAN

Bahasa pemrograman java banyak diaplikasikan dalam berbagai program maupun aplikasi yang berbasis PC atau mobile. Salah satu penerapannya adalah pada program chatting, dipilihnya bahasa java sebagai bahasa pembuatan program ini dikarenakan dalam java memiliki kelas thread sebagai kelas default yang tidak dimiliki oleh bahasa pemrograman lain. Program chat ini terdiri dari dua sisi, yaitu sisi client dan server. Server berfungsi untuk mengatur lalu lintas komunikasi data, sedangkan client berfungsi sebagai alat komunikasi pengguna.

1.2 PEMBAHASAN

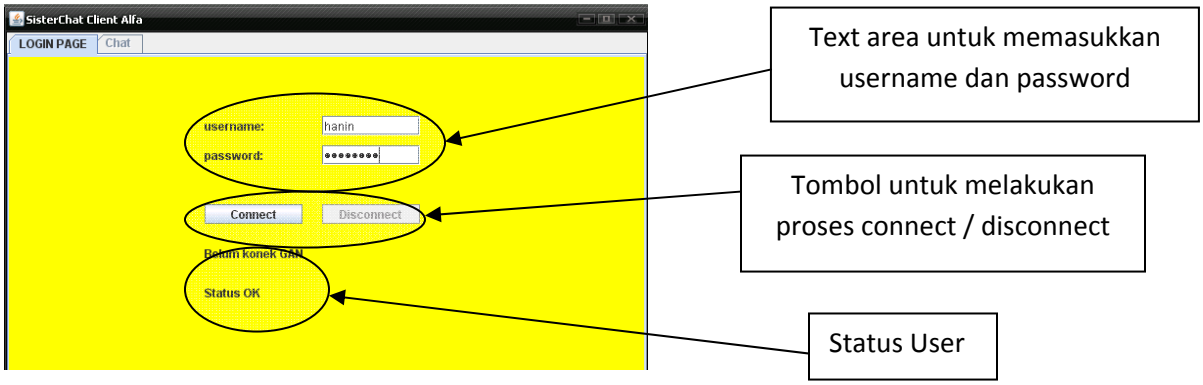
1.2.1 Hasil run program

1.2.1.1 Server

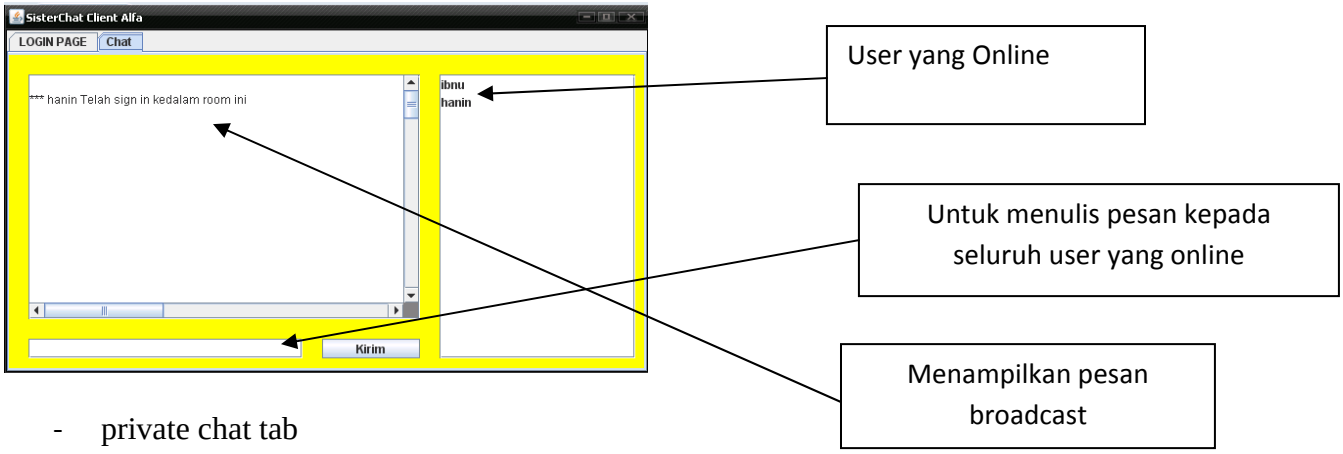


1.2.1.2 Client

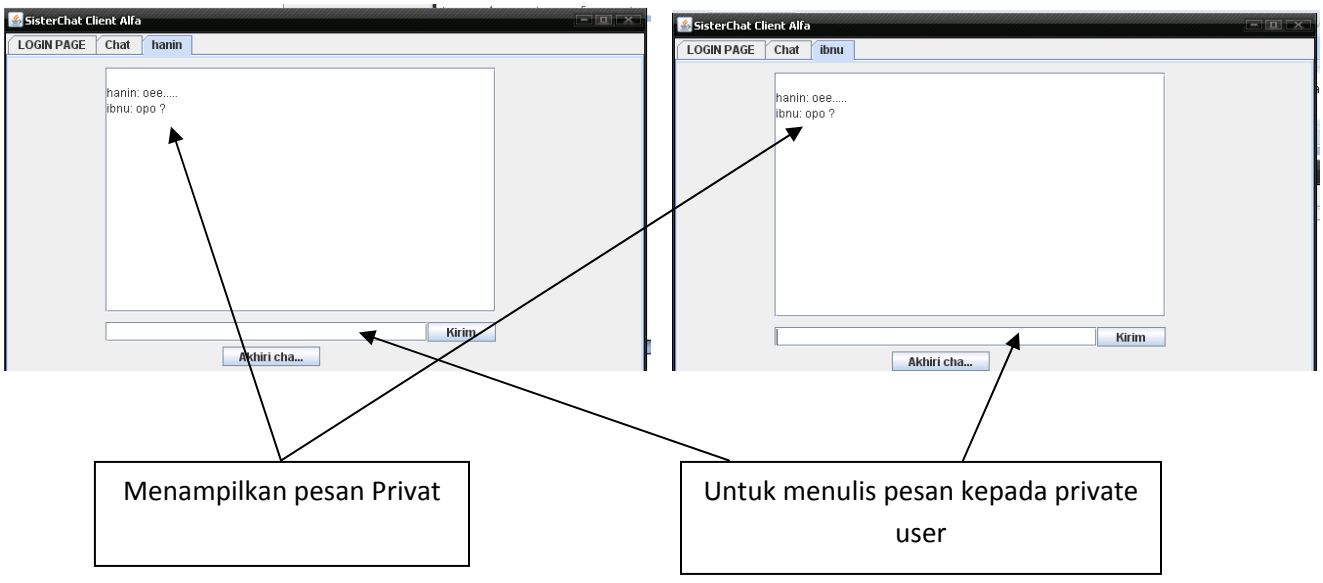
- Login tab



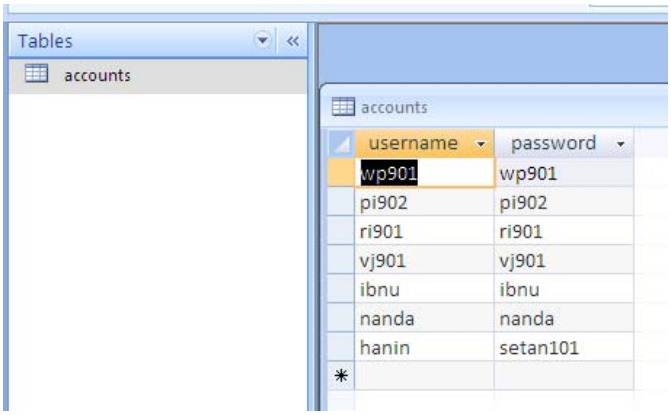
- Chat tab



- private chat tab



1.2.1.3 Database username dan password



Program kami menggunakan database Access. Database ini terhubung dengan program, database ini berisi data username dan password. Apabila ada user yang tidak terdaftar yang ingin melakukan chat, maka user tersebut tidak bisa melakukan chat, yang dapat melakukan hanyalah user yang sudah terdaftar.

1.2.2 Pembahasan Source Code

1.2.2.1 Pembahasan kelas Client

Kelas ini memiliki banyak fungsi yang berhubungan dengan client pada sistem chat. Pada kelas ini mengatur tampilan dan bagaimana sistem client berhubungan dengan sistem server. Berikutnya akan dijelaskan beberapa method penting yang ada pada kelas ini dan fungsi dari method tersebut dalam menunjang sistem chat ini.

```
public Client()   {
      super("SisterChat Client Alfa");
      addWindowListener(
            new WindowAdapter()
            {
                  public void windowClosing(WindowEvent evt)
                  {
                        System.exit(0);
                  }
            }
      );
      setupGUI();
}
```

Program diatas merupakan constructor dari kelas client, pada constructor berfungsi untuk memanggil method setupGUI () yang merupakan method yang berisi tampilan GUI pada sisi client**.**

```
private void setupGUI()
      {
            this.addWindowListener(new         WindowAdapter(){public         void
windowClosing(WindowEvent e){Exit();}});
        this.getContentPane().setLayout(new GridLayout(1, 1));
            this.setSize(660, 380);
        this.setResizable(false);

            // For each tab create a new panel to hold the labels, textfields and
buttons
            Component panel1 = makeNetworkPanel();
        tabbedPane.addTab("LOGIN PAGE", panel1);
        tabbedPane.setSelectedIndex(0);

        Component panel2 = makeChatPanel();
        tabbedPane.addTab("Chat", panel2);
            tabbedPane.setEnabledAt(1, false);

            this.getContentPane().add(tabbedPane);
        this.setVisible(true);
    }
```

Method ini merupakan method yang berfungsi untuk mengatur tampilan GUI awal pada tampilan pada sisi client. Apabila kita ingin mengubah tampilan GUI awal kita maka kita harus mengubah variabel-variabel yang ada pada method ini.

```
protected Component makeNetworkPanel()
      {
            JPanel networkPanel = new JPanel();
                networkPanel.setBackground(Color.getHSBColor(5, 2, 5));
            networkPanel.setLayout(null);

        labelUsernName = new JLabel("username:");
            networkPanel.add(labelUsernName);
            labelUsernName.setBounds(200, 60, 100, 20);

        labelPassword = new JLabel("password:");
            networkPanel.add(labelPassword);
            labelPassword.setBounds(200, 90, 100, 20);

            inputUsername = new JTextField(15);
            networkPanel.add(inputUsername);
            inputUsername.setBounds(320, 60, 100, 20);

            inputPassword = new JPasswordField(15);
            networkPanel.add(inputPassword);
            inputPassword.setBounds(320, 90, 100, 20);

            connect = new JButton("Connect");
            networkPanel.add(connect);
            connect.setBounds(200, 150, 100, 20);

            disconnect = new JButton("Disconnect");
            disconnect.setEnabled(false);
            networkPanel.add(disconnect);
            disconnect.setBounds(320, 150, 100, 20);
```

```
            connectionStatus = new JLabel("Belum konek GAN");
        networkPanel.add(connectionStatus);
        connectionStatus.setBounds(200, 190, 220, 20);

            errorMessages = new JLabel("Status OK");
        networkPanel.add(errorMessages);
        errorMessages.setBounds(200, 230, 220, 20);

            connect.addActionListener(
                  new ActionListener()
                  {
                        public void actionPerformed(ActionEvent e)
                        {
                              username = inputUsername.getText();
                              password = inputPassword.getText();
                              if (username.length() > 0 && password.length() > 0)
                              {
                                    Connect(username, password);
                              }
                              else
                                    JOptionPane.showMessageDialog(new      Frame(),
"Kamu belum memasukkan Username dan password");
                        }
            }
            );

      disconnect.addActionListener(
            new ActionListener()
            {
                  public void actionPerformed(ActionEvent e)
                  {
                        Disconnect();
                  }
            }
      );
        return networkPanel;
    }
```

Method diatas merupakan method yang berfungsi untuk menampilkan halaman LOGIN pada sisi client. Pada method ini juga kita dapat menginput username dan password, membuat fungsi untuk connect dan disconnect dari server, dan pada method ini juga dibuat label untuk pesan apakah kita telah terkoneksi ke server atau belum dan label untuk menampilkan pesan error.

```
      protected Component makeChatPanel()
      {
            JPanel chatPanel = new JPanel();
                chatPanel.setBackground(Color.yellow);
            chatPanel.setLayout(null);

            displayMessages = new JTextArea(100, 100);
            displayMessages.setEditable(false);
            JScrollPane          displayMessagesScrollPane          =          new
JScrollPane(displayMessages);
            chatPanel.add(displayMessagesScrollPane);
```

```
            displayMessagesScrollPane.setBounds(20, 20, 400, 250);
            displayMessagesScrollPane.setBackground(Color.gray);

            onlineList = new JList();
            JScrollPane onlineListScrollPane = new JScrollPane(onlineList);
            chatPanel.add(onlineListScrollPane);
            onlineListScrollPane.setBounds(440, 20, 200, 290);
            onlineListScrollPane.setBackground(Color.gray);

            inputText = new JTextField(100);
        chatPanel.add(inputText);
        inputText.setBounds(20, 290, 280, 20);

            send = new JButton("Kirim");
        chatPanel.add(send);
        send.setBounds(320, 290, 100, 20);

            inputText.addActionListener(
            new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                              String text = inputText.getText();

                    if(text.length() > 0)
                    {
                                  if(!blocked)
                                        writeToServer("PuM",   username  +  ":  "+
text);
                                        else
                                              JOptionPane.showMessageDialog(new
Frame(), "Peringatan! Kamu telah diblock oleh server");
                                  inputText.setText("");
                    }
                }
                  }
            );

            send.addActionListener(
                  new ActionListener()
                  {
                        public void actionPerformed(ActionEvent e)
                        {
                              String text = inputText.getText();

                    if(text.length() > 0)
                    {
                                  if(!blocked)
                                        writeToServer("PuM",   username  +  ":  "+
text);
                                        else
                                              JOptionPane.showMessageDialog(new
Frame(), "Peringatan! Kamu telah diblock oleh server");
                                  inputText.setText("");
                    }//end of if
                        }
                  }
```

```
            );

            onlineList.addMouseListener(
                  new MouseAdapter()
                  {
                        public void mouseClicked(MouseEvent e)
                        {
                              if (e.getClickCount() == 2)
                              {
                                    String                chatWith                =
(String)onlineList.getSelectedValue();
                                    privateMessageManager(chatWith);
                              }
                        }
                  }
            );
            return chatPanel;
      }
```

Method diatas merupakan method yang berfungsi untuk menampilkan halaman chat pada sisi client. Pada method ini juga kita dapat menginput pesan dan mengirimkannya, pada method ini juga terdapat panel untuk menampilkan pesan yang kita kirim dan yang kita terima. Pada method ini juga menyediakan daftar dari client yang terkoneksi dengan server.

```
public void Connect(String username, String password)
      {
            try
            {
                  client = new Socket("127.0.0.1",999);
                  in              =             new             BufferedReader(new
InputStreamReader(client.getInputStream()));
                  out = new PrintStream(client.getOutputStream());
                  String message = username + "$" + password;
                  writeToServer("Login" , message);
                  connect.setEnabled(false);
                  listening = true;
                  thread = new Thread(this);
                  thread.start();
            }
            catch(Exception e)
            {
                  errorMessages.setText(e.getMessage());
            }
      }
```

Method ini berfungsi untuk membuat socket baru, dan berfungsi untuk mengolah data username dan password yang tadi telah kita inputkan pada text area yang ada pada login page. Pada method ini juga akan dibuat instan dari method thread.

```
public void Disconnect()
	{
		closeAllPMs();
		writeToServer("Logoff", username);
		displayMessages.setText("");
		tabbedPane.setEnabledAt(1,false);
		connectionStatus.setText("Belum terkoneksi");
		disconnect.setEnabled(false);
		connect.setEnabled(true);
		try
		{
			thread.stop();
			thread=null;
			listening = false;
			in.close();
			out.close();
			client.close();
		errorMessages.setText("Status: OK");
		}
		catch(Exception e)
		{
			errorMessages.setText(e.getMessage());
		}
	}
```

Pada method ini akan menutup semua private message, semua pada method ini harus mengirimkan pesan ke server dengan subjek logoff setelah itu akan menghentikan thread dan membuat variabel “listening” berubah menjadi false, setelah itu maka chat tab akan tidak bisa dibuka, dan membuat tombol disconnect tidak aktif, dan membuat tombol connect menjadi aktif, setelah itu akan akan membersihkan “displayMessages” textarea.

```
public void run()
	{
		while(listening)
		{
			try
			{
				String fromServer=null;
				if ((fromServer=in.readLine()) !=null)
				{

					String subject = verifySubject(fromServer);
					System.out.println(subject);
					subjectCategory(subject,fromServer);
				}
				else
				{
					listening=false;
					errorMessages.setText("Null message from Server");
				}
			}
			catch(Exception e)
			{
				listening = false;
				try
```

```
                        {
                              in.close();
                              out.close();
                              client.close();
                        }
                        catch(Exception e1)
                        {
                              errorMessages.setText(e1.getMessage());
                        }
      errorMessages.setText(e.getMessage());
                  }
            }
      }
```

Method ini akan terus-menerus mendengarkan socket untuk pesan dari Server. apabila ada pesan yang diterima, maka akan memverifikasi subjek, Setelah itu akan kembali ke mendengarkan socket lagi. Jika kesalahan terjadi maka socket akan tertutup dan sambungan terputus dengan server.

```
public String verifySubject(String message)
      {
            int first$ = message.indexOf('$');
            int second$ = message.indexOf('$',first$+1);
            return message.substring(first$+1,second$);
      }

      public void subjectCategory(String subject, String message)
      {
            if(subject.equalsIgnoreCase("InvalidUser"))
                  subjectInvalidUser(message);
            else if (subject.equalsIgnoreCase("ValidUser"))
                              subjectValidUser(message);
                              else if(subject.equalsIgnoreCase("UserExists"))
                                                subjectUserExists(message);
                                                else
if(subject.equalsIgnoreCase("ClientList"))

subjectClientList(message);
                                                                  else
if(subject.equalsIgnoreCase("ClientStatus"))

      subjectClientStatus(message);

      else if(subject.equalsIgnoreCase("PuM"))

                        subjectPuM(message);

                        else if(subject.equalsIgnoreCase("PrM"))

                                          subjectPrM(message);

                                          else
if(subject.equalsIgnoreCase("Block"))

                                                            subjectBlock();
```

```
                                                        else
if(subject.equalsIgnoreCase("Unblock"))


      subjectUnblock();
      }
```

Pesan yang dikirim oleh server selalu diawali oleh Subyek. Subjek ditandai oleh tanda '$'.Misalnya :

- $ PRM $ menunjukkan Private Message
- $ PUM $ menunjukkan pesan umum
- $ ValidUser $ menunjukkan login yang benar
- $ InvalidUser $ menunjukkan username atau password salah
- $ Userexists $ menunjukkan bahwa pengguna dengan nama user sudah terhubung dari tempat lain
- $ ClientStatus $ menunjukkan bahwa Client telah login atau log out
- $ ClientList $ menunjukkan daftar semua klien tersambung
- $ Blokir $ menunjukkan bahwa server telah memblokir pengguna ini
- $ Unblock $ mengindikasikan bahwa server telah mengunblock pengguna ini

```
public void subjectInvalidUser(String message)
      {
            inputUsername.setText("");
            inputPassword.setText("");
            connect.setEnabled(true);
            errorMessages.setText("Invalid User");
            thread.stop();
            thread=null;
            listening=false;
            try
            {
                  in.close();
                  out.close();
                  client.close();
            }
            catch(Exception e)
            {
                  errorMessages.setText(e.getMessage());
            }
      }
```

Method ini dipanggil ketika user salah memasukkan username dan password, ketika hal ini terjadi maka ada beberapa task yang dikerjakan, yaitu :

- ➔ Membersihkan textfield
- ➔ Tombol button akan aktif jadi user dapat login kembali
- ➔ Thread yang aktif akan berhenti bekerja, dan variabel “listening” akan dibuat dalam kondisi false jadi loop pada method run() akan berhenti.
- ➔ Semua komunikasi dengan server akan segera dihentikan.

```
public void subjectValidUser(String message)
	{
		inputUsername.setText("");
		inputPassword.setText("");
		disconnect.setEnabled(true);
		errorMessages.setText("Konek sebagai " + username);
		connectionStatus.setText("Telah terkoneksi");
		tabbedPane.setEnabledAt(1,true);
		tabbedPane.setSelectedIndex(1);
	}
```

Method ini berfungsi untuk memeriksa apakah username dan password yang diinputkan benar, apabila benar maka chat tab akan aktif dan kita dapat melakukan chat dengan user lain.

```
public void subjectClientList(String list)
	{
		int first$ = list.indexOf('$');
		int second$ = list.indexOf('$',first$+1);
		list = list.substring(second$+1);
		int start = 0;
		int end = start;
		Vector names = new Vector();
		for(int i=start;start<list.length();i++)
		{
			end = list.indexOf('$', start);
			String temp = list.substring(start,end);
			names.addElement(temp);
			start = end+1;
		}
		onlineList.setListData(names);
	}
```

Method ini berfungsi untuk menampilkan user yang online pada list yang berada pada chat tab, format tampilan pada list akan sesuai urutan login, jadi apabila user 1 login terlebih dahulu maka user 1 akan berada pada urutan paling atas.

```
public void subjectClientStatus(String message)
	{

		int first$ = message.indexOf('$');
		int second$ = message.indexOf('$',first$+1);
		message = message.substring(second$+1);

		int next$ = message.indexOf('$');
		String status = message.substring(0,next$);
		String username = message.substring(next$+1);
		if(status.equalsIgnoreCase("Connected"))
		     displayMessages.append("\n" + "*** " + username + " Telah sign in
kedalam room ini");
		 else if(status.equalsIgnoreCase("Disconnected"))
				displayMessages.append("\n" + "*** " + username + " Telah
sign out dari room ini");
	}
```

Pada method ini akan diolah pesan yang diterima, apabila pesan yang diterima adalah “connected” maka akan menampilkan pesan bahwa kita telah sign in ke dalam room, dan apabila pesan yang diterima adalah “Disconnected” maka akan menampilkan pesan bahwa kita telah sign out dari room.

```
public void subjectPuM(String message)
	{

		int first$ = message.indexOf('$');
		int second$ = message.indexOf('$', first$+1);
		message = message.substring(second$+1);
		displayMessages.append("\n" + message);
	}
```

Method ini memiliki fungsi untuk mengextract pesan dan kemudian menampilkan pesan tersebut.

```
public void subjectPrM(String message)
	{
		 int first$ = message.indexOf('$');
		 int second$ = message.indexOf('$', first$+1);
		 int third$ = message.indexOf('$', second$+1);
		 int fourth$ = message.indexOf('$', third$+1);
		 int fifth$ = message.indexOf('$', fourth$+1);
		 String end = message.substring(second$+1,third$);
		 String from = message.substring(third$+1, fourth$);
		 message = message.substring(fifth$+1);
		 if(end.equalsIgnoreCase("true"))
		 {
			for(int i=0;i<pms.size();i++)
			{
				PrivateMessage tmp = (PrivateMessage)pms.get(i);
				if((tmp.chattingWith).equals(from))
				{
					(tmp.displayMessages).append("\n" + "***" + from  +
" Telah mengakhiri chat");
					(tmp.ended) = true;
				}
			}
		 }
		 else if(end.equalsIgnoreCase("false"))
		 {
			if(!(alreadyChatting(from)))
				createNewPM(from);
			for(int i=0;i<pms.size();i++)
			{
				PrivateMessage tmp = (PrivateMessage)pms.get(i);
				if((tmp.chattingWith).equals(from))
				{
					(tmp.displayMessages).append("\n" + message);
				}
			}
		 }
	}
```

Method ini berfungsi untuk menampilkan private message, program akan mengextract pesan kedalam method “privateMessageManager” jika sebuah private chat akan dibuka maka sebuah tab akan segera muncul dengan nama tabnya adalah nama dari user lain yang kita ajak private chat.

```
public void subjectBlock()
	{
		blocked=true;
		JOptionPane.showMessageDialog(new Frame(), "Peringatan! Kamu telah diblock oleh server");
	}

	public void subjectUnblock()
	{
		blocked=false;
		JOptionPane.showMessageDialog(new Frame(), "Kamu telah diunblock oleh server");
	}
```

Kedua method diatas berfungsi untuk mengidentifikasi apakah kita telah diblock atau tidak oleh server, apabila kita telah di block maka akan muncul pesan bahwa kita telah diblock oleh server, dan apabila kita tidak lagi di block maka akan muncul pesan juga bahwa kita telah diunblock oleh server.

```
public void privateMessageManager(String chatWith)
	{
		if(chatWith.equalsIgnoreCase(username))
			return;
		if(!(alreadyChatting(chatWith)))
			createNewPM(chatWith);
	}
```

Method ini berfungsi sebagai pengatur pada private message, method ini juga akan memanggil method createNewPM(), yang berfungsi untuk membuat tab baru apabila kita memulai untuk menggunakan private chat.

```
public boolean alreadyChatting(String chatWith)
	{
		for(int i=0;i<pms.size();i++)
		{
			PrivateMessage tmp = (PrivateMessage)pms.get(i);
			if((tmp.chattingWith).equals(chatWith))
			{
				tabbedPane.setSelectedIndex(i+2);
				return true;
			}
		}
		return false;
	}
```

Method ini berfungsi untuk pengatur sistem pada saat kita memulai private chat, apabila ada pesan masuk maka method ini akan menampilkan pesan yang masuk tersebut pada label yang ada pada tab dengan nama user yang kita aja chat private.

```
public void createNewPM(String chatWith)
	{
		Component panel3 = new PrivateMessage(clientApplication, chatWith);
		tabbedPane.addTab(chatWith, panel3);
		tabbedPane.setEnabledAt(tabbedPane.getTabCount()-1, true);
		tabbedPane.setSelectedIndex(tabbedPane.getTabCount()-1);
		pms.addElement(panel3);
	}
```

Method ini berfungsi untuk membuat tab baru apabila kita memulai untuk melakukan private chat.

```
public void sendPrivateMessage(String end,String to, String message)
	{
		if(!blocked)
			writeToServer("PrM" , end + "$" + username + "$" + to + "$" +
username + ": " + message);
			else
				JOptionPane.showMessageDialog(new   Frame(),   "Peringatan!
Kamu telah diblock oleh server");
	}
```

Method ini berfungsi untuk mengirimkan pesan yang kita ketikkan kepada user yang kita ajak chat private, setelah dikirimkan maka pesan tersebut juga akan muncul pada tampilan chat kita. Dan apabila kita telah diblock oleh server maka kita tidak dapat mengirimkan pesan, dan akan muncul pesan bahwa kita telah diblock oleh server.

```
public void endPrivateChat(String withWhom)
	{
		for(int i=0;i<pms.size();i++)
		{
			PrivateMessage tmp = (PrivateMessage)pms.get(i);
			if((tmp.chattingWith).equals(withWhom))
			{
				tabbedPane.removeTabAt(i+2);
				pms.removeElementAt(i);
				if(!(tmp.ended))
					sendPrivateMessage("true", withWhom, "");
			}
		}
	}
```

Method ini berfungsi untuk proses pada saat kita mengakhiri suatu private chat, apabila kita mengakhiri chat maka tab akan ditutup dan pengiriman pesan private akan berakhir.

```
public void closeAllPMs()
	{
		int noOfPms = pms.size();
		for(int i=0;i<noOfPms;i++)
		{
			System.out.println(i);
			PrivateMessage tmp = (PrivateMessage)pms.get(i);
			tabbedPane.removeTabAt(i+2);
			pms.removeElementAt(i);
```

```
                        if(!(tmp.ended))
                              sendPrivateMessage("true", tmp.chattingWith, "");
                }
        }
}
```

Method ini berfungsi untuk mengakhiri seluruh private message chat, apabila method ini dipanggil maka method ini akan menutup semua tab yang berfungsi sebagai private chat dengan user lain.

1.2.2.2 Pembahasan Kelas ClientThread

Kelas ClientThread ini merupakan kelas turunan dari kelas default dari java yaitu kelas Thread. Di dalamnya terdapat fungsi untuk proses pengiriman informasi dari client kepada server.

```
public class ClientThread extends Thread
{
      public String userName = "$NotSet$";
      BufferedReader in;
      PrintStream out;
      Server serv;
      boolean listening = false;
      public ClientThread(Socket clientSocket, Server serv)
      {
            this.serv = serv;
            try
            {
                  in              =              new              BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                  out = new PrintStream(clientSocket.getOutputStream());
                  serv.addUser(this);
                  listening=true;
            }
            catch(Exception e)
            {
                  System.out.println(e);
            }
      }
```

Bagian di atas merupakan sebuah konstruktor dari kelas ClientThread yang mengatur proses pengiriman data dari client ke server. Di dalamnya terdapat variable in yang menginstansi method BufferedReader yang berfungsi untuk membaca data yang masuk, yang dikirimkan dari client yang lain.

Selain itu juga terdapat variable out yang menginstansi method PrintStream yang berfungsi dalam proses penampilan dari data yang terkirim. Kemudian variable serv yang menambahkan user pada tampilan GUI nya.

```
	public void run()
		{
			while(listening)
			{
				try
				{
					String message=null;
					if((message=in.readLine())!=null)
					{
						writeMessageToParent(message);
					}
					else
					{
						listening=false;
						System.out.println("r");
					}
				}
				catch(Exception e)
				{
					stop();
					listening=false;
					System.out.println(e);
				}
			}
		}

		public void writeMessageToParent(String message)
		{
			serv.messageFromClient(message);
		}
}
```

Kode di atas merupakan method run dan juga method writeMessagetoAllClients. Method run ini berfungsi pada saat menunggu adanya pesan yang masuk, jika tidak ada pesan, maka akan terus dilakukan penungguan. Jika mulai ada pesan yang masuk maka akan dilakukan penampilannya.

1.2.2.3 Pembahasan Kelas Server

Kelas Server ini merupakan kelas turunan dari kelas default JFrame, kelas ini akan mengatur jalannya sistem pada server. Berikut ini akan dibahas beberapa method penting yang terdapat pada kelas Server.java tersebut.

```
	public Server()
	{
		super("Sister Chat Server Alfa");
		addWindowListener(
			new WindowAdapter()
			{
				public void windowClosing(WindowEvent evt)
				{
					System.exit(0);
				}//end of method windowClosing()
			}//end of class WindowAdapter()
```

```
        );
        setupGUI();
        show();
        setSize(320,450);
        setResizable(false);

    }
```

Hal pertama yang akan dikerjakan pada kelas ini adalah pembuatan sebuah konstruktor seperti tampak pada source code di atas. Konstruktor ini akan memanggil suatu method show(), dan juga akan memanggil method setupGUI() yang merupakan method yang akan membuat suatu tampilan GUI seperti tombol, panel, dan lainnya. Method setupGUI() ini dapat dilihat pada lampiran.

```
    public void messageFromClient(String message)
    {
          String subject = verifySubject(message);
    subjectCategory(subject, message);
    }
```

Program di atas merupakan sebuah method yang berfungsi untuk memverifikasi Subject dari pesan yang dikirimkan oleh client, yang bernama method messageFromClient.

```
    public String verifySubject(String message)
    {
          int first$ = message.indexOf('$');
          int second$ = message.indexOf('$',first$+1);
          return message.substring(first$+1,second$);
    }
```

Program di atas merupakan sebuah method yang akan menverifikasi Subject pesan. Pesan atau Messages yang dikirim oleh client selalu diawali oleh suatu Subject. Subject ini dilampiri dengan tanda ‘$’. Diantaranya PrM mengindikasikan sebuah Private Message atau pesan pribadi, PuM mengindikasikan sebuah Public Message atau pesan public, $Login$ yang mengindikasikan sebuah login dari user yang baru, dan $Logoff$ mengindikasikan sebuah logoff dari user. Sehingga pada statement berikut, variable first$ menyimpan indeks dari tanda ‘$’ yang pertama (atau dengan kata lain ia menyimpan posisi dari permulaian suatu Subject). Kemudian ia melokasikan posisi dari tanda ‘$’ kedua dan menyimpannya pada variable second$. Kemudian dengan bantuan method substring, Subject akan di ekstrak dan dikembalikan.

```
    public void subjectCategory(String subject , String message)
    {
         if(subject.equalsIgnoreCase("PrM"))
                subjectPrM(message);
         else if(subject.equals("PuM"))
                     subjectPuM(message);
                     else if(subject.equals("Login"))
                                 subjectLogin(message);
                                 else if(subject.equals("Logoff"))
                                             subjectLogoff(message);
    }
```

Di atas merupakan program atau method untuk pembagian pesan berdasarkan type-nya (subject). Kategori ini akan di definisi ulang dan Client telah mengetahui kategori tersebut.

```
    public void subjectPrM(String message)
    {
           int first$ = message.indexOf('$');
```

```
        int second$ = message.indexOf('$', first$+1);
        int third$ = message.indexOf('$', second$+1);
        int fourth$ = message.indexOf('$', third$+1);
        int fifth$ = message.indexOf('$', fourth$+1);
        String to = message.substring(fourth$+1, fifth$);
```

Pesan yang dikirim oleh client memiliki banyak sekali informasi, tetapi Server tidak harus terfokus pada hal-hal tersebut, terkecuali untuk bagian dimana dikatakan kepada siapa pesan ini akan dikirimkan. Formatnya adalah sebagai berikut : PrM<end>$<from>$<to>$<message>

```
        for(int i=0;i<clients.size();i++)
        {
              ClientThread tmp = (ClientThread)clients.get(i);
              if((tmp.userName).equalsIgnoreCase(to))
              {
                    (tmp.out).println(message);
              }
        }
  }
```

Setelah server menemukan kepada siapa pesan tersebut harus dikirimkan, kemudian ia mengecek daftar-daftar client untuk memastikan ketepatannya client yang akan dituju tersebut. Dan langsung mengirimkan pesan kepadanya.

```
  public void subjectPuM(String message)
  {
        int first$ = message.indexOf('$');
        int second$ = message.indexOf('$', first$+1);
        message = message.substring(second$+1);
        for(int i=0;i<clients.size();i++)
        {
              ClientThread tmp = (ClientThread)clients.get(i);
              (tmp.out).println("$PuM$" + message);
        }
  }
```

Program di atas merupakan sebuah method subjectPuM, subject yang menyatakan bahwa pesan ini merupakan pesan publik. Fungsi dari method merupakan salah satu dari yang paling sederhana. Ia memindahkan subject dari pesan dan melanjutkannya kepada semua client, karena merupakan sebuah pesan publik. Formatnya adalah sebagai berikut : PuM<message>

```
  public void subjectLogin(String message)
  {
        int first$ = message.indexOf('$');
        int second$ = message.indexOf('$', first$+1);
        int third$ = message.indexOf('$', second$+1);
        String username = message.substring(second$+1, third$);
        String password = message.substring(third$+1);
        boolean validUser = verifyUserName(username,password);
```

Berikut merupakan sebuah method subjectLogin, yang merupakan pemrosesan login seorang user berdasarkan subject pesan yang diberikan. Format dari pesan ini adalah: $Login$<username>$<password>. Ia akan mengekstrak username dan password, kemudian mengecek apakah valid atau tidak valid.

```
if(validUser)
{
```

Jika program sampai disini maka username dan password yang diberikan valid. Tetapi sekarang ia akan mengecek apakah user tersebut sudah siap untuk terkoneksi dari suatu tempat lain.

```
for(int i=0;i<clients.size();i++)
{
      ClientThread tmp = (ClientThread)(clients.get(i));
      if((tmp.userName).equalsIgnoreCase("$NotSet$"))
      {
```

Jika program mencapai disini makan si user belum terkoneksi dan namanya harus di tambahkan ke threadnya, sebagai contoh untuk objek tmp dan kemudian objek ini harus diset kembali ke dalam vector. Dan client harus diinformasikan mengenai hal ini. Juga daftar nama client harus dikirimkan kepada semua user. Kemudian juga mengirim status user saat ini kepada semua client kecuali dirinya sendiri.

```
                        tmp.userName = username;
                        clients.set(i,tmp);
                        clientNames.addElement(username);
                        blockedUsers.addElement("false");
                        listArea.setListData(clientNames);
                        labelClientsCon.setText("Jumlah      Client      yang
terkoneksi : " + clientNames.size());
                        (tmp.out).println("$ValidUser$");
                        String list = getClientList();
                        for(int k=0;k<clientNames.size();k++)
                        {
                              ClientThread              temp              =
(ClientThread)clients.get(k);
                              (temp.out).println("$ClientList$" + list);
                              (temp.out).println("$ClientStatus$"           +
"Connected$" + username);
                        }
                        return;
                  }
                  else if((tmp.userName).equalsIgnoreCase(username))
                              {
```

Jika program telah mencapai disini kemudian ini berarti bahwa seseorang yang lain dengan nama username yang sama telah terkoneksi sebelumnya. Sehingga akan dikirimkan kembali sebuah pesan “UserExists” kepada client. Dan kontrol program akan dikembalikan setelah menghilangkan thread user dari koleksi atau daftar. Pada loop berikut, program berusaha untuk mencari object(ClientThread) yang memiliki username sebagai “$NotSet$”. Ia akan mencari salah satu dan kemudian menghilangkan thread tersebut dari Vector dan kemudian kembali lagi.

```
                                    for(int j=0;j<clients.size();j++)
                                    {
                                          ClientThread         temp         =
(ClientThread)(clients.get(j));  //THIS variable is temp and NOT tmp


      if((temp.userName).equalsIgnoreCase("$NotSet$"))
                                          {

      (temp.out).println("$UserExists$");
```

```
                                                      try
                                                      {

((ClientThread)(clients.get(j))).in.close();

((ClientThread)(clients.get(j))).out.close();
                                                      }//end of try
                                                      catch(Exception e)
                                                      {

labelErrorMessages.setText(e.getMessage());
                                                      }//end of catch
                                                      clients.removeElementAt(j);
                                                      return;
                                                }
                                          }
                                    }
                              }
                        }
                        else
                        {
```

User tidak memiliki izin sehingga akan dikirimkan pesan kepada Client tersebut mengenai hal ini. Juga akan ditutup thread yang sedang digunakan oleh user ini. Dan menghapus objeknya dari Vector.

```
                  for(int i=0;i<clients.size();i++)
                   {
                        ClientThread tmp = (ClientThread)(clients.get(i));
                        if((tmp.userName).equalsIgnoreCase("$NotSet$"))
                        {
                             (tmp.out).println("$InvalidUser$");
                             clients.removeElementAt(i);
                             return;
                        }//end of if
                   }//end of for
            }//end of else
      }//end of method subjectLogin()

      public String getClientList()
      {
            String list = "";
            for(int i=0;i<clientNames.size();i++)
            {
                  list = list + (String)clientNames.get(i);
                  list = list.concat("$");
            }//end of for
            return list;
      }//end of method getClientList()

      public boolean verifyUserName(String username, String password)
      {
```

Kerja dari fungsi ini adalah untuk mengkoneksikan ke database dan mengecek apakah username dan password terdapat di dalam database tersebut. Jika terdapat maka akan diberikan nilai boolean true , dan nilai false untuk hal sebaliknya.

```
            try
```

```
            {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                  Connection                              con                              =
DriverManager.getConnection("jdbc:odbc:accounts");
                  PreparedStatement  stat  =  con.prepareStatement("SELECT  *  FROM
accounts WHERE accounts.username = '" + username + "' AND accounts.password = '" +
password + "'");
                  ResultSet result = stat.executeQuery();
                  int rows=0;
                  while(result.next())
                  {
                        rows = result.getRow();
                  }//end of while

                  if(rows>0)
                  {
```

Jika kontrol sampai disini maka berarti bahwa user ini memiliki izin.

```
                        return true;
                  }
```

Jika kontrol mencapai disini maka berarti user ini tidak valid.

```
                  return false;
            }

            catch(Exception e)
            {
                  labelErrorMessages.setText(e.getMessage());
            }//end of catch
            return false;
      }//end of method verifyUserName()

      public void subjectLogoff(String message)
      {
```

Format dari pesan ini akan berupa $Logoff$<username>. Sehingga pertama dilakukan ekstrak terhadap username dari pesan. Method ini memiliki responsibility terhadap pencarian vector “client” yang mana menyimpan objek ClientThread. Dan mencari thread yang berasosiasi dengan user ini. Kemudian menghapus objek (ClientThread) dari vector dan menutup semua koneksi dengan user tersebut. Dan kemudian meng-update daftar nama client dan ListArea yang menampilkan jumlah dari client yang terkoneksi.

```
            int first$ = message.indexOf('$');
            int second$ = message.indexOf('$', first$+1);
            String username = message.substring(second$+1);
            for(int i=0;i<clients.size();i++)
            {
                  ClientThread tmp = (ClientThread)clients.get(i);
                  if(tmp.userName.equalsIgnoreCase(username))
                  {
```

Jika program telah sampai disini maka semua fungsi di atas yang telah didefinisikan komen di atas telah dikerjakan.

```
                        try
                        {
                              ((ClientThread)(clients.get(i))).in.close();
                              ((ClientThread)(clients.get(i))).out.close();
                        }
                        catch(Exception e)
```

```
                              {
                                    labelErrorMessages.setText(e.getMessage());
                              }
                         clients.removeElementAt(i);
                         boolean removed = clientNames.removeElement(username);
                         if(!removed)
                              labelErrorMessages.setText("Daftar     tidak     bisa
diupdate");
                        listArea.setListData(clientNames);
                        labelClientsCon.setText("Jumlah Client yang terkoneksi " +
String.valueOf(clientNames.size()));
                        String list = getClientList();
                        for(int k=0;k<clientNames.size();k++)
                              {
                                    ClientThread               temp               =
(ClientThread)clients.get(k);
                                    (temp.out).println("$ClientList$" + list);
                                    (temp.out).println("$ClientStatus$"           +
"Disconnected$" + username);
                              }
                        return;
                  }
            }
            labelErrorMessages.setText("Clien  yang  tidak  teridentifikasi  mencoba
untuk logoff");
            return;
      }
```

Berikut merupakan method yang berfungsi untuk melakukan block terhadap suatu client.

```
      private void blockUser()
      {
            int index = listArea.getSelectedIndex();
            if(index != -1)
            {
                  blockedUsers.setElementAt("true", index);
                  ClientThread tmp = (ClientThread)clients.get(index);
                  (tmp.out).println("$Block$");
                  buttonBlock.setEnabled(false);
                  buttonUnblock.setEnabled(true);
            }
      }
```

Berikut merupakan method yang berfungsi untuk melakukan unblock terhadap suatu client.

```
      private void unblockUser()
      {
            int index = listArea.getSelectedIndex();
            if(index != -1)
            {
                  blockedUsers.setElementAt("false", index);
                  ClientThread tmp = (ClientThread)clients.get(index);
                  (tmp.out).println("$Unblock$");
                  buttonBlock.setEnabled(true);
                  buttonUnblock.setEnabled(false);
            }
      }
}
```

1.2.2.4 Pembahasan Kelas ServList

Kelas ServList ini merupakan kelas turunan dari kelas default JFrame, kelas ini akan menampilkan daftar client yang terhubung, dan merupakan kelas yang dapat di run atau dijalankan. Pada kelas ini akan dibuat sebuah objek dari kelas Server(). Serta akan membuat suatu thread bagi client.

```
class ServList extends JFrame
{
	public static void main(String[] args)
	{
		Server serv = new Server();
		ServerSocket server;
		boolean listening=true;
		try
		{
			server = new ServerSocket(999);
			System.out.println(server.toString());
			while(listening)
			{
				Socket temp = server.accept();
				(new ClientThread(temp,serv)).start();
			}
		server.close();
		}
		catch(Exception e)
		{
			System.out.println("Exception while starting THREAD: " + e);
		}
	}
}
```

1.2.2.5 Pembahasan kelas PrivateMessage

Kelas PrivateMessage ini merupakan kelas yang berfungsi dalam proses pengiriman pesan antar client. Kelas ini merupakan kelas turunan dari JPanel.

```
public class PrivateMessage extends JPanel
{
	String chattingWith=null;
	boolean ended = false;

	private JScrollPane scroll;
	JTextArea displayMessages;
	private JTextField inputText;
	private JButton send, end;
	private Client parent;

	public PrivateMessage(Client parent, String chattingWith)
	{
		this.parent = parent;
	    this.chattingWith = chattingWith;
```

```
            setLayout(null);
            setupGUI();
            setSize(660,380);
      }
```

Di dalam kelas PrivateMessage tersebut terdaat konstruktor PrivateMessage yang berfungsi dalam proses setup GUI dan mengeset ukuran dari tampilannya menjadi 660x380.

```
      public void setupGUI()
      {
            displayMessages = new JTextArea();
            displayMessages.setEditable(false);
            scroll = new JScrollPane(displayMessages);
            inputText = new JTextField();
            send = new JButton("Kirim");
            end = new JButton("Akhiri chatting");

            inputText.addActionListener(
                  new ActionListener()
                  {
                        public void actionPerformed(ActionEvent evt)
                        {
                              String text = inputText.getText();
                              if(text.length() > 0 && !ended)
                              {
                                    inputText.setText("");
                                    parent.sendPrivateMessage("false", chattingWith
, text);
                                    displayMessages.append("\n" + parent.username +
": " + text);
                              }
                        }
                  }
```

Kode di atas berisi class ActionListener dan method actionPerformed. Pada setup GUI diatur tampilannya berupa JTextArea, JTextField, serta button send yang diberi label ‘kirim’ dan end dengan label ‘akhiri’. Pada input ActionListener, dilakukan pemasukan terhadap pesan, dan juga akan ditampilkan parent user name dan pembicaraanya.

```
            );

            send.addActionListener(
                  new ActionListener()
                  {
                        public void actionPerformed(ActionEvent evt)
                        {
                              String text = inputText.getText();
                              if(text.length() > 0 && !ended)
                              {
                                    inputText.setText("");
                                    parent.sendPrivateMessage("false", chattingWith
, text);
                                    displayMessages.append("\n" + parent.username +
": " + text);
                              }
                        }
```

```
            }
        );
```

Method di atas merupakan action listener untuk membaca di kliknya tombol send untuk pengiriman datanya. Dengan format di dalamnya yaitu parent user name, dan text dari messagenya kemudian akan dikirimkan.

```
        end.addActionListener(
            new ActionListener()
            {
                public void actionPerformed(ActionEvent evt)
                {
                        parent.endPrivateChat(chattingWith);
                }
            }
        );
```

Method endActionListener di atas berfungsi untuk membaca action yang dilakukan yaitu membaca di kliknya tombol end yang berfungsi untuk mengakhiri pembicaraan. Jika dilakukan action ini maka akan mengakhiri pembicaraan.

```
        add(scroll);
        add(inputText);
        add(send);
        add(end);

        scroll.setBounds(100, 10, 400, 250);
        inputText.setBounds(100,270,330,20);
        send.setBounds(430,270,70,20);
        end.setBounds(220,295,100,20);
    }
}
```

1.3 Lampiran

1.3.1 Kelas Client

```
import java.net.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.border.*;
import java.awt.*;
import java.io.*;
import java.util.*;

public class Client extends JFrame implements Runnable
{
      private JTabbedPane tabbedPane = new JTabbedPane();

      private JLabel labelUsernName, labelPassword, connectionStatus;
      protected JLabel errorMessages;
      private JTextField inputUsername;
      private JPasswordField inputPassword;
      private JButton connect, disconnect;

      private JTextArea displayMessages;
      private JTextField inputText;
      private JButton send;
      private JList onlineList;


      String username, password;

      BufferedReader in;
      PrintStream out;
      Socket client;

      Thread thread;
      boolean listening = false;

      boolean blocked = false;

      Vector pms = new Vector();

      static Client clientApplication;

      public static void main(String args[])
    {
            clientApplication = new Client();
    }

      public Client()
      {
            super("SisterChat Client Alfa");
            addWindowListener(
                  new WindowAdapter()
                  {
                        public void windowClosing(WindowEvent evt)
                        {
```

```
                        System.exit(0);
                    }
                }
            );
            setupGUI();
      }

      private void setupGUI()
      {
            this.addWindowListener(new         WindowAdapter(){public         void
windowClosing(WindowEvent e){Exit();}});
        this.getContentPane().setLayout(new GridLayout(1, 1));
            this.setSize(660, 380);
        this.setResizable(false);

            Component panel1 = makeNetworkPanel();
        tabbedPane.addTab("LOGIN PAGE", panel1);
        tabbedPane.setSelectedIndex(0);

        Component panel2 = makeChatPanel();
        tabbedPane.addTab("Chat", panel2);
            tabbedPane.setEnabledAt(1, false);

            this.getContentPane().add(tabbedPane);
        this.setVisible(true);
    }



      protected Component makeNetworkPanel()
      {
            JPanel networkPanel = new JPanel();
                networkPanel.setBackground(Color.getHSBColor(5, 2, 5));
            networkPanel.setLayout(null);

        labelUsernName = new JLabel("username:");
            networkPanel.add(labelUsernName);
            labelUsernName.setBounds(200, 60, 100, 20);

        labelPassword = new JLabel("password:");
            networkPanel.add(labelPassword);
            labelPassword.setBounds(200, 90, 100, 20);

            inputUsername = new JTextField(15);
            networkPanel.add(inputUsername);
            inputUsername.setBounds(320, 60, 100, 20);

            inputPassword = new JPasswordField(15);
            networkPanel.add(inputPassword);
            inputPassword.setBounds(320, 90, 100, 20);

            connect = new JButton("Connect");
            networkPanel.add(connect);
            connect.setBounds(200, 150, 100, 20);

            disconnect = new JButton("Disconnect");
            disconnect.setEnabled(false);
```

```
            networkPanel.add(disconnect);
            disconnect.setBounds(320, 150, 100, 20);

            connectionStatus = new JLabel("Belum konek GAN");
          networkPanel.add(connectionStatus);
          connectionStatus.setBounds(200, 190, 220, 20);

            errorMessages = new JLabel("Status OK");
          networkPanel.add(errorMessages);
          errorMessages.setBounds(200, 230, 220, 20);

            connect.addActionListener(
                  new ActionListener()
                  {
                        public void actionPerformed(ActionEvent e)
                        {
                              username = inputUsername.getText();
                              password = inputPassword.getText();
                              if (username.length() > 0 && password.length() > 0)
                              {
                                    Connect(username, password);
                              }
                              else
                                    JOptionPane.showMessageDialog(new      Frame(),
"Kamu belum memasukkan Username dan password");
                        }
            }
            );

      disconnect.addActionListener(
            new ActionListener()
            {
                  public void actionPerformed(ActionEvent e)
                  {
                        Disconnect();

            }
      );
          return networkPanel;
    }




      protected Component makeChatPanel()
      {
            JPanel chatPanel = new JPanel();
                chatPanel.setBackground(Color.yellow);
            chatPanel.setLayout(null);

            displayMessages = new JTextArea(100, 100);
            displayMessages.setEditable(false);
            JScrollPane          displayMessagesScrollPane          =          new
JScrollPane(displayMessages);
            chatPanel.add(displayMessagesScrollPane);
            displayMessagesScrollPane.setBounds(20, 20, 400, 250);
            displayMessagesScrollPane.setBackground(Color.gray);
```

```
            onlineList = new JList();
            JScrollPane onlineListScrollPane = new JScrollPane(onlineList);
            chatPanel.add(onlineListScrollPane);
            onlineListScrollPane.setBounds(440, 20, 200, 290);
            onlineListScrollPane.setBackground(Color.gray);

            inputText = new JTextField(100);
        chatPanel.add(inputText);
        inputText.setBounds(20, 290, 280, 20);

            send = new JButton("Kirim");
        chatPanel.add(send);
        send.setBounds(320, 290, 100, 20);

            inputText.addActionListener(
            new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                              String text = inputText.getText();

                    if(text.length() > 0)
                    {
                                  if(!blocked)
                                        writeToServer("PuM",   username  +  ":  "+
text);
                                        else
                                              JOptionPane.showMessageDialog(new
Frame(), "Peringatan! Kamu telah diblock oleh server");
                                  inputText.setText("");
                    }
                }
                  }
            );

            send.addActionListener(
                  new ActionListener()
                  {
                        public void actionPerformed(ActionEvent e)
                        {
                              String text = inputText.getText();

                    if(text.length() > 0)
                    {
                                  if(!blocked)
                                        writeToServer("PuM",   username  +  ":  "+
text);
                                        else
                                              JOptionPane.showMessageDialog(new
Frame(), "Peringatan! Kamu telah diblock oleh server");
                                  inputText.setText("");
                    }
                        }
                  }
            );
```

```
            onlineList.addMouseListener(
                  new MouseAdapter()
                  {
                        public void mouseClicked(MouseEvent e)
                        {
                              if (e.getClickCount() == 2)
                              {
                                    String                chatWith                =
(String)onlineList.getSelectedValue();
                                    privateMessageManager(chatWith);
                              }
                        }
                  }
            );
            return chatPanel;
      }

      public void Connect(String username, String password)
      {
            try
            {
                  client = new Socket("127.0.0.1",999);
                  in              =             new             BufferedReader(new
InputStreamReader(client.getInputStream()));
                  out = new PrintStream(client.getOutputStream());
                  String message = username + "$" + password;
                  writeToServer("Login" , message);
                  connect.setEnabled(false);
                  listening = true;
                  thread = new Thread(this);
                  thread.start();
            }
            catch(Exception e)
            {
                  errorMessages.setText(e.getMessage());
            }
      }


      public void Disconnect()
      {
            closeAllPMs();
            writeToServer("Logoff", username);
            displayMessages.setText("");
            tabbedPane.setEnabledAt(1,false);
            connectionStatus.setText("Belum terkoneksi");
            disconnect.setEnabled(false);
            connect.setEnabled(true);
            try
            {
                  thread.stop();
                  thread=null;
                  listening = false;
                  in.close();
                  out.close();
                  client.close();
            errorMessages.setText("Status: OK");
```

```
	}
	catch(Exception e)
	{
		errorMessages.setText(e.getMessage());
	}
}
public void writeToServer(String subject, String message)
{
	out.println("$" + subject + "$" + message);
}



public void Exit()
{
	Disconnect();
}


public void run()
{
	while(listening)
	{
		try
		{
			String fromServer=null;
			if ((fromServer=in.readLine()) !=null)
			{


				String subject = verifySubject(fromServer);
				System.out.println(subject);
				subjectCategory(subject,fromServer);
			}			else
			{
				listening=false;
				errorMessages.setText("Null message from Server");
			}
		}
		catch(Exception e)
		{
			listening = false;
			try
			{
				in.close();
				out.close();
				client.close();
			}
			catch(Exception e1)
			{
				errorMessages.setText(e1.getMessage());
			}
			errorMessages.setText(e.getMessage());
		}
	}
}
```

```
	public String verifySubject(String message)
	{
		int first$ = message.indexOf('$');
		int second$ = message.indexOf('$',first$+1);
		return message.substring(first$+1,second$);
	}


	public void subjectCategory(String subject, String message)
	{
		if(subject.equalsIgnoreCase("InvalidUser"))
			subjectInvalidUser(message);
		else if (subject.equalsIgnoreCase("ValidUser"))
					subjectValidUser(message);
					else if(subject.equalsIgnoreCase("UserExists"))
								subjectUserExists(message);
								else
if(subject.equalsIgnoreCase("ClientList"))

subjectClientList(message);
                                                                  else
if(subject.equalsIgnoreCase("ClientStatus"))

	subjectClientStatus(message);

	else if(subject.equalsIgnoreCase("PuM"))

					subjectPuM(message);

					else if(subject.equalsIgnoreCase("PrM"))

								subjectPrM(message);

								else
if(subject.equalsIgnoreCase("Block"))

                                                                  subjectBlock();

                                                                  else
if(subject.equalsIgnoreCase("Unblock"))


	subjectUnblock();
	}

	public void subjectInvalidUser(String message)
	{


		inputUsername.setText("");
		inputPassword.setText("");
		connect.setEnabled(true);
		errorMessages.setText("Invalid User");
		thread.stop();
		thread=null;
		listening=false;
```

```
	try
	{
		in.close();
		out.close();
		client.close();
	}
	catch(Exception e)
	{
		errorMessages.setText(e.getMessage());
	}
}

public void subjectValidUser(String message)
{
	inputUsername.setText("");
	inputPassword.setText("");
	disconnect.setEnabled(true);
	errorMessages.setText("Konek sebagai " + username);
	connectionStatus.setText("Telah terkoneksi");
	tabbedPane.setEnabledAt(1,true);
	tabbedPane.setSelectedIndex(1);
}

public void subjectUserExists(String message)
{
	inputUsername.setText("");
	inputPassword.setText("");
	connect.setEnabled(true);
	errorMessages.setText("User Telah konek");
	connectionStatus.setText("Belum terkoneksi");
	listening=false;
	try
	{
		in.close();
		out.close();
		client.close();
	}
	catch(Exception e)
	{
		errorMessages.setText(e.getMessage());
	}
}

public void subjectClientList(String list)
{


	 list = list.substring(second$+1);
	 int start = 0;
	 int end = start;

	 Vector names = new Vector();
	 for(int i=start;start<list.length();i++)
	 {
		end = list.indexOf('$', start);
		String temp = list.substring(start,end);
		names.addElement(temp);
```

```
                start = end+1;
            }

            onlineList.setListData(names);
      }

      public void subjectClientStatus(String message)
      {

            int next$ = message.indexOf('$');
            String status = message.substring(0,next$);
            String username = message.substring(next$+1);
            if(status.equalsIgnoreCase("Connected"))
                displayMessages.append("\n" + "*** " + username + " Telah sign in
kedalam room ini");
             else if(status.equalsIgnoreCase("Disconnected"))
                        displayMessages.append("\n" + "*** " + username + " Telah
sign out dari room ini");
      }//end of method subjectClientStatus()

      public void subjectPuM(String message)
      {

           int first$ = message.indexOf('$');
           int second$ = message.indexOf('$', first$+1);
           message = message.substring(second$+1);
           displayMessages.append("\n" + message);
      }
      public void subjectPrM(String message)
      {


            int first$ = message.indexOf('$');
            int second$ = message.indexOf('$', first$+1);
            int third$ = message.indexOf('$', second$+1);
            int fourth$ = message.indexOf('$', third$+1);
            int fifth$ = message.indexOf('$', fourth$+1);
            String end = message.substring(second$+1,third$);
            String from = message.substring(third$+1, fourth$);
            message = message.substring(fifth$+1);
            if(end.equalsIgnoreCase("true"))
            {
                  for(int i=0;i<pms.size();i++)
                  {
                        PrivateMessage tmp = (PrivateMessage)pms.get(i);
                        if((tmp.chattingWith).equals(from))
                        {
                              (tmp.displayMessages).append("\n" + "***" + from  +
" Telah mengakhiri chat");
                              (tmp.ended) = true;
                        }
                  }
            }
            else if(end.equalsIgnoreCase("false"))
            {
                  if(!(alreadyChatting(from)))
                        createNewPM(from);
```

```
                for(int i=0;i<pms.size();i++)
                {
                      PrivateMessage tmp = (PrivateMessage)pms.get(i);
                      if((tmp.chattingWith).equals(from))
                      {
                            (tmp.displayMessages).append("\n" + message);
                      }
                }
          }
      }

      public void subjectBlock()
      {
            blocked=true;
            JOptionPane.showMessageDialog(new  Frame(),  "Peringatan!  Kamu  telah
diblock oleh server");
      }

      public void subjectUnblock()
      {
            blocked=false;
            JOptionPane.showMessageDialog(new Frame(), "Kamu telah diunblock oleh
server");
      }

      public void privateMessageManager(String chatWith)
      {
            if(chatWith.equalsIgnoreCase(username))
                  return;
            if(!(alreadyChatting(chatWith)))
                  createNewPM(chatWith);
      }
      public boolean alreadyChatting(String chatWith)
      {
            for(int i=0;i<pms.size();i++)
            {
                  PrivateMessage tmp = (PrivateMessage)pms.get(i);
                  if((tmp.chattingWith).equals(chatWith))
                  {
                        tabbedPane.setSelectedIndex(i+2);
                        return true;
                  }
            }
            return false;
      }

      public void createNewPM(String chatWith)
      {
            Component panel3 = new PrivateMessage(clientApplication, chatWith);
            tabbedPane.addTab(chatWith, panel3);
            tabbedPane.setEnabledAt(tabbedPane.getTabCount()-1, true);
            tabbedPane.setSelectedIndex(tabbedPane.getTabCount()-1);
            pms.addElement(panel3);
      }

      public void sendPrivateMessage(String end,String to, String message)
      {
```

```
            if(!blocked)
                  writeToServer("PrM" , end + "$" + username + "$" + to + "$" +
username + ": " + message);
                  else
                        JOptionPane.showMessageDialog(new   Frame(),   "Peringatan!
Kamu telah diblock oleh server");
      }

      public void endPrivateChat(String withWhom)
      {
            for(int i=0;i<pms.size();i++)
            {
                  PrivateMessage tmp = (PrivateMessage)pms.get(i);
                  if((tmp.chattingWith).equals(withWhom))
                  {
                        tabbedPane.removeTabAt(i+2);
                        pms.removeElementAt(i);
                        if(!(tmp.ended))
                              sendPrivateMessage("true", withWhom, "");
                  }
            }
      }

      public void closeAllPMs()
      {
            int noOfPms = pms.size();
            for(int i=0;i<noOfPms;i++)
            {
                  System.out.println(i);
                  PrivateMessage tmp = (PrivateMessage)pms.get(i);
                  tabbedPane.removeTabAt(i+2);
                  pms.removeElementAt(i);
                        if(!(tmp.ended))
                              sendPrivateMessage("true", tmp.chattingWith, "");
            }
      }
}
```

1.3.2 Kelas ClientThread

```
import java.awt.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

public class ClientThread extends Thread
{
      public String userName = "$NotSet$";
      BufferedReader in;
      PrintStream out;
      Server serv;
      boolean listening = false;
      public ClientThread(Socket clientSocket, Server serv)
      {
            this.serv = serv;
```

```
            try
            {
                  in              =              new              BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                  out = new PrintStream(clientSocket.getOutputStream());
                  serv.addUser(this);
                  listening=true;
            }
            catch(Exception e)
            {
                  System.out.println(e);
            }
      }

      public void run()
            {
                  while(listening)
                  {
                        try
                        {
                              String message=null;
                              if((message=in.readLine())!=null)
                              {
                                    writeMessageToParent(message);
                              }
                              else
                              {
                                    listening=false;
                                    System.out.println("r");
                              }
                        }
                        catch(Exception e)
                        {
                              stop();
                              listening=false;
                              System.out.println(e);
                        }
                  }
            }

            public void writeMessageToParent(String message)
            {
                  serv.messageFromClient(message);
            }
}
```

1.3.3 Kelas Server

```
import java.awt.Color;
import java.io.*;
import java.net.*;
import java.util.*;
import javax.swing.*;
import java.awt.event.*;
import java.sql.*;
```

```
class Server extends JFrame
{
	private Vector clients = new Vector();
	private Vector clientNames = new Vector();
	private Vector blockedUsers = new Vector();
	private JList listArea;
	private JLabel labelErrorMessages;
	private JLabel labelClientsCon;
	private JButton buttonBlock, buttonUnblock;

	public Server()
	{
		super("Sister Chat Server Alfa");
		addWindowListener(
			new WindowAdapter()
			{
				public void windowClosing(WindowEvent evt)
				{
					System.exit(0);
				}
			}
		);
		setupGUI();
		show();
		setSize(320,450);
		    setResizable(false);

	}

	public void setupGUI()
	{
		getContentPane().setLayout(null);
		    getContentPane().setBackground(Color.green);

		labelErrorMessages = new JLabel("Status: OK");
		labelErrorMessages.setBounds(30,360,100,20);
		labelClientsCon = new JLabel("Jumlah Client yang terkoneksi : " +
clientNames.size());
		labelClientsCon.setBounds(30,320,200,20);
		getContentPane().add(labelErrorMessages);
		getContentPane().add(labelClientsCon);

		listArea = new JList(clientNames);
		getContentPane().add(listArea);
		listArea.setBounds(5,5,300,300);
		listArea.addMouseListener(
			new MouseAdapter()
			{
				public void mouseClicked(MouseEvent e)
				{
					int index = listArea.getSelectedIndex();
					if(index != -1)
					{
						String status =
(String)(blockedUsers.get(index));
						if(status.equalsIgnoreCase("false"))
```

```
                                        buttonBlock.setEnabled(true);
                                    else if(status.equalsIgnoreCase("true"))

      buttonUnblock.setEnabled(true);
                              }
                        }
                  }
            );

            buttonBlock = new JButton("Block Client");
            buttonUnblock = new JButton("Unblock Client");
            getContentPane().add(buttonBlock);
            buttonBlock.setBounds(5,390,150,20);
            buttonBlock.setEnabled(false);
            getContentPane().add(buttonUnblock);
            buttonUnblock.setBounds(150,390,150,20);
            buttonUnblock.setEnabled(false);

            buttonBlock.addActionListener(new     ActionListener(){public     void
actionPerformed(ActionEvent evt){blockUser();}});
            buttonUnblock.addActionListener(new    ActionListener(){public    void
actionPerformed(ActionEvent evt){unblockUser();}});
      }

      public void addUser(ClientThread thread)
      {
            clients.addElement(thread);
      }

      public void messageFromClient(String message)
      {
            String subject = verifySubject(message);
            subjectCategory(subject, message);
      }//end of method messageFromClient()

      public String verifySubject(String message)
      {

            int first$ = message.indexOf('$');
            int second$ = message.indexOf('$',first$+1);
            return message.substring(first$+1,second$);
      }

      public void subjectCategory(String subject , String message)
      {

           if(subject.equalsIgnoreCase("PrM"))
                 subjectPrM(message);
           else if(subject.equals("PuM"))
                        subjectPuM(message);
                        else if(subject.equals("Login"))
                                    subjectLogin(message);
                                    else if(subject.equals("Logoff"))
                                                subjectLogoff(message);
      }

      public void subjectPrM(String message)
```

```
	{


		for(int i=0;i<clients.size();i++)
		{
			ClientThread tmp = (ClientThread)clients.get(i);
			if((tmp.userName).equalsIgnoreCase(to))
			{
				(tmp.out).println(message);
			}
		}
	}

	public void subjectPuM(String message)
	{

		int first$ = message.indexOf('$');
		int second$ = message.indexOf('$', first$+1);
		message = message.substring(second$+1);
		for(int i=0;i<clients.size();i++)
		{
			ClientThread tmp = (ClientThread)clients.get(i);
			(tmp.out).println("$PuM$" + message);
		}
	}

	public void subjectLogin(String message)
	{

		int first$ = message.indexOf('$');
		int second$ = message.indexOf('$', first$+1);
		int third$ = message.indexOf('$', second$+1);
		String username = message.substring(second$+1, third$);
		String password = message.substring(third$+1);
		boolean validUser = verifyUserName(username,password);
		if(validUser)
		{

			for(int i=0;i<clients.size();i++)
			{
				ClientThread tmp = (ClientThread)(clients.get(i));
				if((tmp.userName).equalsIgnoreCase("$NotSet$"))
				{

					tmp.userName = username;
					clients.set(i,tmp);
					clientNames.addElement(username);
					blockedUsers.addElement("false");
					listArea.setListData(clientNames);
					labelClientsCon.setText("Jumlah	Client	yang
terkoneksi : " + clientNames.size());
					(tmp.out).println("$ValidUser$");

					String list = getClientList();
					for(int k=0;k<clientNames.size();k++)
					{
```

```
                                    ClientThread                temp                =
(ClientThread)clients.get(k);
                                    (temp.out).println("$ClientList$" + list);
                                    (temp.out).println("$ClientStatus$"            +
"Connected$" + username);
                              }
                              return;
                        }
                        else if((tmp.userName).equalsIgnoreCase(username))
                              {

                                    for(int j=0;j<clients.size();j++)
                                    {
                                          ClientThread         temp         =
(ClientThread)(clients.get(j));


      if((temp.userName).equalsIgnoreCase("$NotSet$"))
                                                {

      (temp.out).println("$UserExists$");

      //((ClientThread)(clients.get(j))).stop();
                                                try
                                                      {

      ((ClientThread)(clients.get(j))).in.close();

      ((ClientThread)(clients.get(j))).out.close();
                                                      }
                                                      catch(Exception e)
                                                      {

      labelErrorMessages.setText(e.getMessage());
                                                      }
                                                      clients.removeElementAt(j);
                                                      return;
                                                }
                                    }
                              }
                  }
            }
            else
            {

                  for(int i=0;i<clients.size();i++)
                   {
                        ClientThread tmp = (ClientThread)(clients.get(i));
                        if((tmp.userName).equalsIgnoreCase("$NotSet$"))
                        {
                             (tmp.out).println("$InvalidUser$");
                             //((ClientThread)(clients.get(i))).stop();
                             clients.removeElementAt(i);
                             return;
                        }
                   }
            }
```

```
	}

	public String getClientList()
	{
		String list = "";
		for(int i=0;i<clientNames.size();i++)
		{
			list = list + (String)clientNames.get(i);
			list = list.concat("$");
		}
		return list;
	}

	public boolean verifyUserName(String username, String password)
	{

		try
		{
			Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
			Connection                          con                          =
DriverManager.getConnection("jdbc:odbc:accounts");
			PreparedStatement  stat  =  con.prepareStatement("SELECT  *  FROM
accounts WHERE accounts.username = '" + username + "' AND accounts.password = '" +
password + "'");
			ResultSet result = stat.executeQuery();
			int rows=0;
			while(result.next())
			{
				rows = result.getRow();
			}//end of while

			if(rows>0)
			{

				return true;
			}

			return false;
		}
		catch(Exception e)
		{
			labelErrorMessages.setText(e.getMessage());
		}
		return false;
	}

	public void subjectLogoff(String message)
	{

		int first$ = message.indexOf('$');
		int second$ = message.indexOf('$', first$+1);
		String username = message.substring(second$+1);
		for(int i=0;i<clients.size();i++)
		{
			ClientThread tmp = (ClientThread)clients.get(i);
			if(tmp.userName.equalsIgnoreCase(username))
			{
```

```
                        try
                              {
                                    ((ClientThread)(clients.get(i))).in.close();
                                    ((ClientThread)(clients.get(i))).out.close();
                              }
                              catch(Exception e)
                              {
                                    labelErrorMessages.setText(e.getMessage());
                              }
                        clients.removeElementAt(i);
                        boolean removed = clientNames.removeElement(username);
                        if(!removed)
                              labelErrorMessages.setText("Daftar     tidak     bisa
diupdate");
                       listArea.setListData(clientNames);
                       labelClientsCon.setText("Jumlah Client yang terkoneksi " +
String.valueOf(clientNames.size()));
                      String list = getClientList();
                       for(int k=0;k<clientNames.size();k++)
                              {
                                    ClientThread               temp               =
(ClientThread)clients.get(k);
                                    (temp.out).println("$ClientList$" + list);
                                    (temp.out).println("$ClientStatus$"           +
"Disconnected$" + username);
                              }
                       return;
                  }
            }
            labelErrorMessages.setText("Clien  yang  tidak  teridentifikasi  mencoba
untuk logoff");
            return;
      }

      private void blockUser()
      {
            int index = listArea.getSelectedIndex();
            if(index != -1)
            {
                  blockedUsers.setElementAt("true", index);
                  ClientThread tmp = (ClientThread)clients.get(index);
                  (tmp.out).println("$Block$");
                  buttonBlock.setEnabled(false);
                  buttonUnblock.setEnabled(true);
            }
      }

      private void unblockUser()
      {
            int index = listArea.getSelectedIndex();
            if(index != -1)
            {
                  blockedUsers.setElementAt("false", index);
                  ClientThread tmp = (ClientThread)clients.get(index);
                  (tmp.out).println("$Unblock$");
                  buttonBlock.setEnabled(true);
```

```
                buttonUnblock.setEnabled(false);
            }
      }
}
```

1.3.4 Kelas ServList

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

class ServList extends JFrame
{
      public static void main(String[] args)
      {
            Server serv = new Server();
            ServerSocket server;
            boolean listening=true;
            try
            {
                  server = new ServerSocket(999);
                  System.out.println(server.toString());
                  while(listening)
                  {
                        Socket temp = server.accept();
                        (new ClientThread(temp,serv)).start();
                  }
            server.close();
            }
            catch(Exception e)
            {
                  System.out.println("Exception while starting THREAD: " + e);
            }
      }
}
```

1.3.5 Kelas PrivateMessage

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class PrivateMessage extends JPanel
{
      String chattingWith=null;
      boolean ended = false;

      private JScrollPane scroll;
      JTextArea displayMessages;
      private JTextField inputText;
      private JButton send, end;
      private Client parent;

      public PrivateMessage(Client parent, String chattingWith)
```

```
	{
		this.parent = parent;
	    this.chattingWith = chattingWith;
		setLayout(null);
		setupGUI();
		setSize(660,380);
	}

	public void setupGUI()
	{
		displayMessages = new JTextArea();
		displayMessages.setEditable(false);
		scroll = new JScrollPane(displayMessages);
		inputText = new JTextField();
		send = new JButton("Kirim");
		end = new JButton("Akhiri chatting");

		inputText.addActionListener(
			new ActionListener()
			{
				public void actionPerformed(ActionEvent evt)
				{
					String text = inputText.getText();
					if(text.length() > 0 && !ended)
					{
						inputText.setText("");
						parent.sendPrivateMessage("false", chattingWith
, text);
						displayMessages.append("\n" + parent.username +
": " + text);
					}
				}
			}
		);

		send.addActionListener(
			new ActionListener()
			{
				public void actionPerformed(ActionEvent evt)
				{
					String text = inputText.getText();
					if(text.length() > 0 && !ended)
					{
						inputText.setText("");
						parent.sendPrivateMessage("false", chattingWith
, text);
						displayMessages.append("\n" + parent.username +
": " + text);
					}
				}
			}
		);

		end.addActionListener(
			new ActionListener()
			{
				public void actionPerformed(ActionEvent evt)
```

```
                        {
                                parent.endPrivateChat(chattingWith);
                        }
                }
        );

        add(scroll);
        add(inputText);
        add(send);
        add(end);

        scroll.setBounds(100, 10, 400, 250);
        inputText.setBounds(100,270,330,20);
        send.setBounds(430,270,70,20);
        end.setBounds(220,295,100,20);
    }
}
```