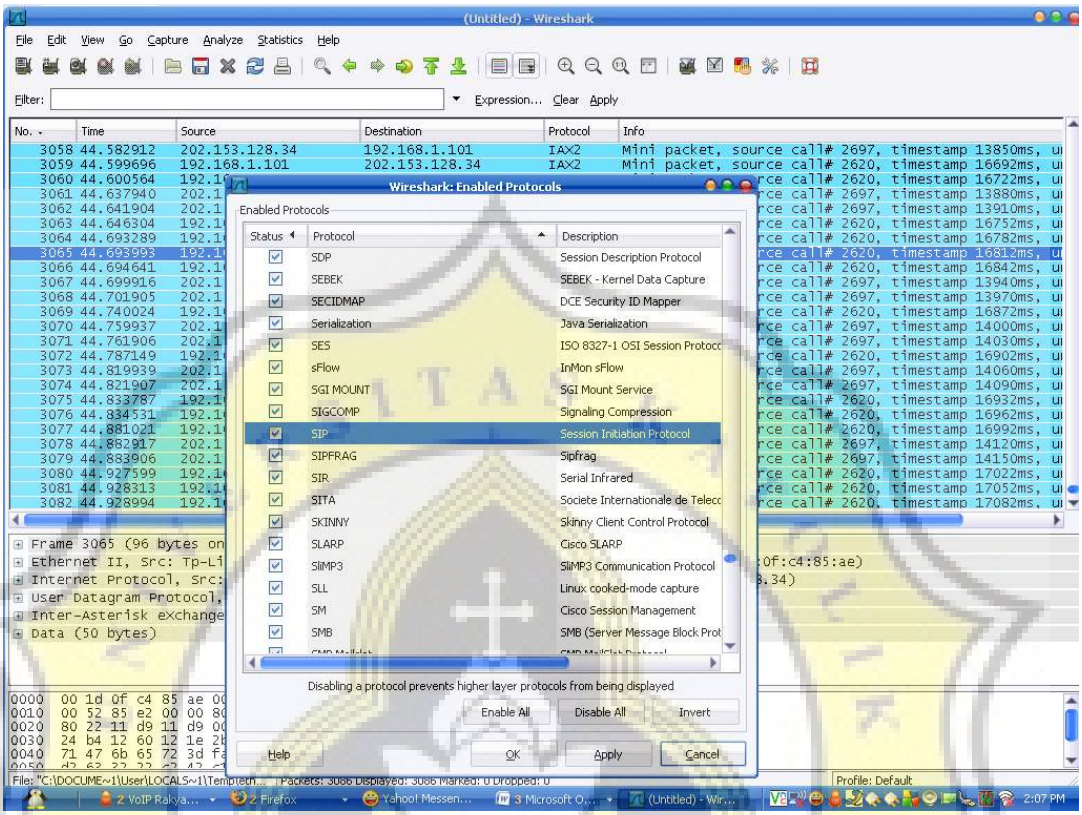
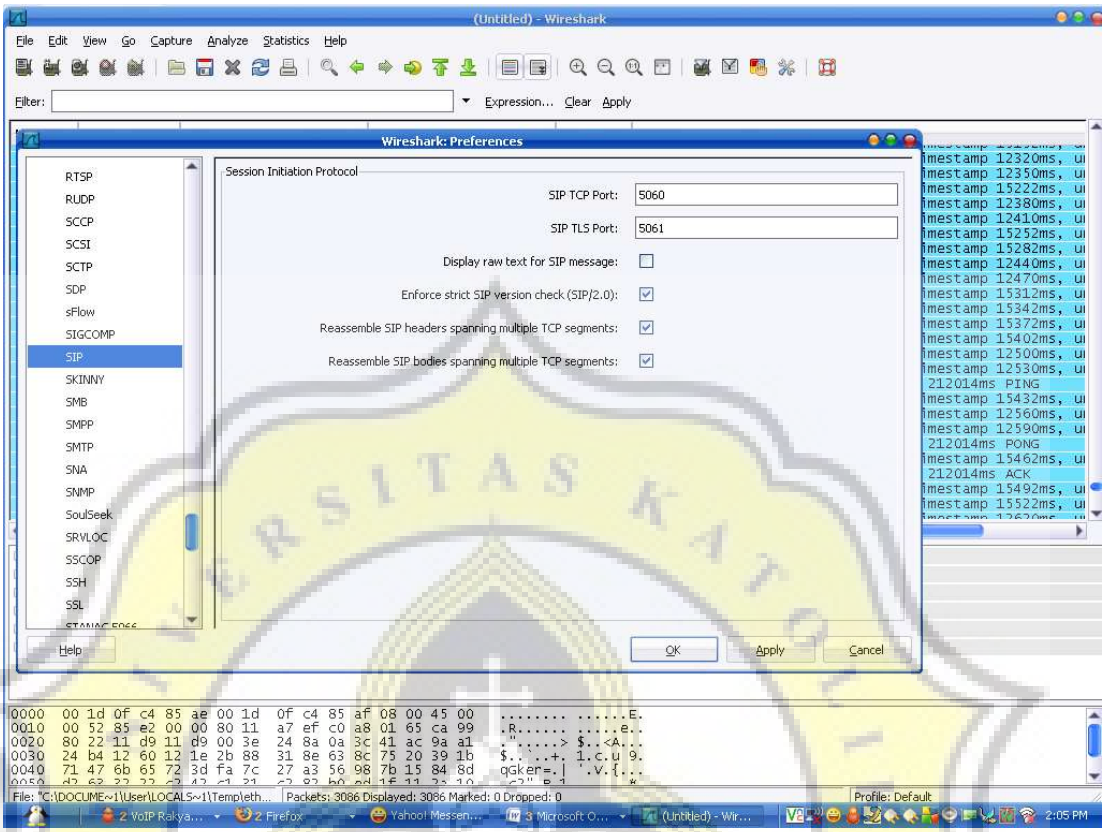
LAMPIRAN





Listing progam

```
*/
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.sip.*;

// import paket


public class SIPDemo extends MIDlet implements CommandListener {
    //inisialisasi variabel
    private static final String appTitle = "SIP Demo";
    private String sipAddress = "Somebody@127.0.0.1";
```

```
private int portReceive = 5070;
private int portSend = 5070;
private String message = "Some Message Body";
private String msgSubject = "Test Message";
private Display display;
private Form form = new Form(appTitle);
private List list;
private Command goCommand = new Command("Go", Command.ITEM, 1);
private Command exitCommand = new Command("Exit", Command.EXIT, 0);
private Command sendCommand = new Command("Send", Command.OK, 1);
private Command receiveCommand = new Command("Receive", Command.OK, 1);
private TextField tfAddress;
private TextField tfPort;
private TextField tfSubject;
private TextField tfMessage;
private SipClientConnection sc = null;
private SipConnectionNotifier scn = null;
private SipServerConnection ssc = null;

public SIPDemo() {
   //konstruktor = menetapkan status awal dmn objek dibuat
   super();
   display = Display.getDisplay(this);
   initList();
   display.setCurrent(list);
}

private void initList() {
   //method initList
   //mengisikan daftar apa saja yg mau d masukkan dalam aplikasi
   list = new List(appTitle, Choice.IMPLICIT);
   list.append("Send Message", null);
   list.append("Receive Message", null);

   list.addCommand(exitCommand);
   list.addCommand(goCommand);
   list.setCommandListener(this);
```

```
    }

    private void composeMessage() {
        //method composeMessage, membuat text field yg akan d tampilkan di layar
        //bila list menu pertama dipilih "Send Message" maka akan muncul method ini
        tfAddress = new TextField("Address ", sipAddress, 50, TextField.LAYOUT_LEFT);
        tfPort = new TextField("Port ", String.valueOf(portSend), 6,
TextField.LAYOUT_LEFT);
        tfSubject = new TextField("Subject ", msgSubject, 50, TextField.LAYOUT_LEFT);
        tfMessage = new TextField("Message Body ", message, 150,
TextField.LAYOUT_LEFT);
        form.append(tfAddress);
        form.append(tfPort);
        form.append(tfSubject);
        form.append(tfMessage);

        form.addCommand(sendCommand);
        form.addCommand(exitCommand);
        form.setCommandListener(this);
        display.setCurrent(form);
    }

    public void receiveMessage() {
        //method receiveMessage, membuat text field yg akan d tampilkan di layar
        //bila list menu pertama dipilih "Receive Message" maka akan muncul method ini
        tfPort = new TextField("Port ", String.valueOf(portReceive), 6,
TextField.LAYOUT_LEFT);
        form.append(tfPort);
        form.addCommand(receiveCommand);
        form.addCommand(exitCommand);
        form.setCommandListener(this);
        display.setCurrent(form);
    }

    public void receiveTextMessage() {
        //method receivedTextMessage, membuat object Thread dengan nama receiveThread
        Thread receiveThread = new ReceiveThread();
```

```
    receiveThread.start();
}

public void commandAction(Command c, Displayable s) {
    //method commandAction, semua command d atur d dalam method ini
    if (c == exitCommand) {
        destroyApp(true);
        notifyDestroyed();
    } else if (((s == list) && (c == List.SELECT_COMMAND)) || (c == goCommand)) {
        int i = list.getSelectedIndex();

        if (i == 0) { // Send Msg
            composeMessage();
        } else if (i == 1) { // Receive Msg
            receiveMessage();
        }
    } else if (s == form) {
        if (c == sendCommand) {
            message = tfMessage.getString();
            msgSubject = tfSubject.getString();
            sipAddress = tfAddress.getString();
            portSend = getPort(tfPort.getString());

            if (portSend == -1) {
                return;
            }

            sendTextMessage(message);
        } else if (c == receiveCommand) {
            portReceive = getPort(tfPort.getString());

            if (portReceive == -1) {
                return;
            }

            form.deleteAll();
            form.removeCommand(receiveCommand);
```

```
            receiveTextMessage();
        }
    }
}

/**
 * Converts ASCII to int and ensures a positive number.
 * -1 indicates an error.
 */
public int getPort(String s) {
    int i = -1;

    try {
        i = Integer.valueOf(s).intValue();
    } catch (NumberFormatException nfe) {
        // don't do anything, the number will be -1
    }

    if (i < 0) {
        Alert alert = new Alert("Error");
        alert.setType(AlertType.ERROR);
        alert.setTimeout(3000); // display the alert for 3 secs
        alert.setString("The port is not a positive number.\n" +
            "Please enter a valid port number.");
        display.setCurrent(alert);

        return -1;
    }

    return i;
}

public void startApp() {
    System.out.println("Starting SIP Demo...\n");
}

public void sendTextMessage(String msg) {
```

```
        SendThread sendThread = new SendThread();
        sendThread.start();
    }

    public void destroyApp(boolean b) {
        //method destroyApp, bila aplikasi d close maka
        //SipClientConnection, SipConnectionNotifier, SipServerConnection akan d close bila isinya bukan null
        //peintah closing akan d tampilkan d console
        System.out.println("Destroying app...\n");

        try {
            if (sc != null) {
                System.out.println("Closing Client Connection...");
                sc.close();
            }

            if (ssc != null) {
                System.out.println("Closing Server Connection...");
                ssc.close();
            }

            if (scn != null) {
                System.out.println("Closing Notifier Connection...");
                scn.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            notifyDestroyed();
        }
    }

    public void pauseApp() {
        //method pause, d dalam method ini tidak berisi apa
        //method ini tidak d panggil dalam aplikasi, lebih baik d hilangkan
        System.out.println("Paused...\n");
```

```
}

/** Send Message Thread */
class SendThread extends Thread implements SipClientConnectionListener {
   //class SendThread, class ini dpanggil d method send TextMessage
   private int recTimeout = 0; // do not wait, just poll

   public void notifyResponse(SipClientConnection sc) {
      try {
         sc.receive(recTimeout);
         form.append("Response received: " + sc.getStatusCode() + " " +
            sc.getReasonPhrase());
         sc.close();
      } catch (Exception ex) {
         form.append("MIDlet: exception " + ex.getMessage());
         ex.printStackTrace();
      }
   }

   public void run() {
      try {
         sc = (SipClientConnection)Connector.open("sip:" + sipAddress + ":" + portSend);
         sc.setListener(this);

         sc.initRequest("MESSAGE", null);
         sc.setHeader("From", "sip:" + sipAddress);
         sc.setHeader("Subject", msgSubject);
         sc.setHeader("Content-Type", "text/plain");
         sc.setHeader("Content-Length", Integer.toString(message.length()));

         OutputStream os = sc.openContentOutputStream();
         os.write(message.getBytes());

         os.close(); // close the stream and send the message

         form.deleteAll();
         form.removeCommand(sendCommand);
```

```
            form.append("Message sent...\n");
        } catch (IllegalArgumentException iae) {
            Alert alert = new Alert("Error");
            alert.setType(AlertType.ERROR);
            alert.setTimeout(3000); // display the alert for 3 secs
            alert.setString("Some of the submitted data is invalid.\n" +
                "Please enter valid information.");
            display.setCurrent(alert);
        } catch (Exception ex) {
            form.append("MIDlet: exception " + ex.getMessage());
            ex.printStackTrace();
        }
    }
}

/** Receive Message Thread */
class ReceiveThread extends Thread {
    //class ReceiveThread meng-extends class Thread yg ada dalam java
    //dia d gunakan pada method receiveTextMessage
    private byte[] buffer = new byte[0xFF];

    public void run() {
        try {
            scn = (SipConnectionNotifier)Connector.open("sip:" + portReceive);
            form.append("Listening at " + portReceive + "...");

            // block and wait for incoming request.
            // SipServerConnection is established and returned
            // when a new request is received.
            ssc = scn.acceptAndOpen();

            if (ssc.getMethod().equals("MESSAGE")) {
                String contentType = ssc.getHeader("Content-Type");

                if ((contentType != null) && contentType.equals("text/plain")) {
                    InputStream is = ssc.openContentInputStream();
                    int bytesRead;
```

```
                String msg = new String("");

                while ((bytesRead = is.read(buffer)) != -1) {
                   msg += new String(buffer, 0, bytesRead);
                }

                form.append("\n...Message Received\n");
                form.append("Subject: \"" + ssc.getHeader("Subject") + "\"\n");
                form.append("Body: \"" + msg + "\"\n\n");
              }

              // initialize SIP 200 OK and send it back
              ssc.initResponse(200);
              ssc.send();
              form.append("Response sent...\n");
            }

            ssc.close();
          } catch (Exception ex) {
            // IOException
            // InterruptedIOException
            // SecurityException
            // SipException
            form.append("MIDlet: exception " + ex.getMessage());
            ex.printStackTrace();
          }
        }
      }
    }
```