Mail User Agent Project Additional Classes

This page contains the code for all the other classes in the `MailClient` lab: `MailClient.java`, `Message.java`, and `Envelope.java`. **All of these classes will require modification in order to handle the actual project specifications.**

`MailClient.java`

```
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;

/* $Id: MailClient.java,v 1.7 1999/07/22 12:07:30 kangasha Exp $ */

/**
 * A simple mail client with a GUI for sending mail.
 *
 * @author Jussi Kangasharju
 */
public class MailClient extends Frame {
    /* The stuff for the GUI. */
    private Button btSend = new Button("Send");
    private Button btClear = new Button("Clear");
    private Button btQuit = new Button("Quit");
    private Label serverLabel = new Label("Local mailserver:");
    private TextField serverField = new TextField("", 40);
    private Label fromLabel = new Label("From:");
    private TextField fromField = new TextField("", 40);
    private Label toLabel = new Label("To:");
    private TextField toField = new TextField("", 40);
    private Label subjectLabel = new Label("Subject:");
    private TextField subjectField = new TextField("", 40);
    private Label messageLabel = new Label("Message:");
    private TextArea messageText = new TextArea(10, 40);

    /**
     * Create a new MailClient window with fields for entering all
     * the relevant information (From, To, Subject, and message).
     */
    public MailClient() {
super("Java Mailclient");
```

```
/* Create panels for holding the fields.  To make it look nice,
   create an extra panel for holding all the child panels. */
Panel serverPanel = new Panel(new BorderLayout());
Panel fromPanel = new Panel(new BorderLayout());
Panel toPanel = new Panel(new BorderLayout());
Panel subjectPanel = new Panel(new BorderLayout());
Panel messagePanel = new Panel(new BorderLayout());
serverPanel.add(serverLabel, BorderLayout.WEST);
serverPanel.add(serverField, BorderLayout.CENTER);
fromPanel.add(fromLabel, BorderLayout.WEST);
fromPanel.add(fromField, BorderLayout.CENTER);
toPanel.add(toLabel, BorderLayout.WEST);
toPanel.add(toField, BorderLayout.CENTER);
subjectPanel.add(subjectLabel, BorderLayout.WEST);
subjectPanel.add(subjectField, BorderLayout.CENTER);
messagePanel.add(messageLabel, BorderLayout.NORTH);
messagePanel.add(messageText, BorderLayout.CENTER);
Panel fieldPanel = new Panel(new GridLayout(0, 1));
fieldPanel.add(serverPanel);
fieldPanel.add(fromPanel);
fieldPanel.add(toPanel);
fieldPanel.add(subjectPanel);

/* Create a panel for the buttons and add listeners to the
   buttons. */
Panel buttonPanel = new Panel(new GridLayout(1, 0));
btSend.addActionListener(new SendListener());
btClear.addActionListener(new ClearListener());
btQuit.addActionListener(new QuitListener());
buttonPanel.add(btSend);
buttonPanel.add(btClear);
buttonPanel.add(btQuit);

/* Add, pack, and show. */
add(fieldPanel, BorderLayout.NORTH);
add(messagePanel, BorderLayout.CENTER);
add(buttonPanel, BorderLayout.SOUTH);
pack();
show();
    }

    static public void main(String argv[]) {
new MailClient();
    }

    /* Handler for the Send-button. */
    class SendListener implements ActionListener {
```

```
public void actionPerformed(ActionEvent event) {
    System.out.println("Sending mail");

    /* Check that we have the local mailserver */
    if ((serverField.getText()).equals("")) {
System.out.println("Need name of local mailserver!");
return;
    }

    /* Check that we have the sender and recipient. */
    if((fromField.getText()).equals("")) {
System.out.println("Need sender!");
return;
    }
    if((toField.getText()).equals("")) {
System.out.println("Need recipient!");
return;
    }

    /* Create the message */
    Message mailMessage = new Message(fromField.getText(),
      toField.getText(),
      subjectField.getText(),
      messageText.getText());

    /* Check that the message is valid, i.e., sender and
       recipient addresses look ok. */
    if(!mailMessage.isValid()) {
return;
    }

    /* Create the envelope, open the connection and try to send
       the message. */
    try {
Envelope envelope = new Envelope(mailMessage,
 serverField.getText());
    } catch (UnknownHostException e) {
/* If there is an error, do not go further */
return;
    }
    try {
SMTPConnection connection = new SMTPConnection(envelope);
connection.send(envelope);
connection.close();
    } catch (IOException error) {
System.out.println("Sending failed: " + error);
return;
    }
```

```
    System.out.println("Mail sent succesfully!");
}
    }

    /* Clear the fields on the GUI. */
    class ClearListener implements ActionListener {
public void actionPerformed(ActionEvent e) {
    System.out.println("Clearing fields");
    fromField.setText("");
    toField.setText("");
    subjectField.setText("");
    messageText.setText("");
}
    }

    /* Quit. */
    class QuitListener implements ActionListener {
public void actionPerformed(ActionEvent e) {
    System.exit(0);
}
    }
}
```

Message.java

```
import java.util.*;
import java.text.*;

/* $Id: Message.java,v 1.5 1999/07/22 12:10:57 kangasha Exp $ */

/**
 * Mail message.
 *
 * @author Jussi Kangasharju
 */
public class Message {
    /* The headers and the body of the message. */
    public String Headers;
    public String Body;

    /* Sender and recipient.  With these, we don't need to extract them
       from the headers. */
    private String From;
    private String To;

    /* To make it look nicer */
    private static final String CRLF = "\r\n";
```

```
    /* Create the message object by inserting the required headers from
       RFC 5322 (From, To, Date). */
    public Message(String from, String to, String subject, String text) {
/* Remove whitespace */
From = from.trim();
To = to.trim();
Headers = "From: " + From + CRLF;
Headers += "To: " + To + CRLF;
Headers += "Subject: " + subject.trim() + CRLF;

/* The RFC 5322-required format. */
SimpleDateFormat format =
    new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss Z");
String dateString = format.format(new Date());
Headers += "Date: " + dateString + CRLF;
Body = text;
    }

    /* Two functions to access the sender and recipient. */
    public String getFrom() {
return From;
    }

    public String getTo() {
return To;
    }

    /* Check whether the message is valid.  In other words, check that
       both sender and recipient contain only one @-sign. */
    public boolean isValid() {
int fromat = From.indexOf('@');
int toat = To.indexOf('@');

if(fromat < 1 || (From.length() - fromat) <= 1) {
    System.out.println("Sender address is invalid");
    return false;
}
if(toat < 1 || (To.length() - toat) <= 1) {
    System.out.println("Recipient address is invalid");
    return false;
}
if(fromat != From.lastIndexOf('@')) {
    System.out.println("Sender address is invalid");
    return false;
}
if(toat != To.lastIndexOf('@')) {
    System.out.println("Recipient address is invalid");
    return false;
```

```
}
return true;
    }

    /* For printing the message. */
    public String toString() {
String res;

res = Headers + CRLF;
res += Body;
return res;
    }
}
```

Envelope.java

```
import java.io.*;
import java.net.*;
import java.util.*;

/* $Id: Envelope.java,v 1.8 1999/09/06 16:43:20 kangasha Exp $ */

/**
 * SMTP envelope for one mail message.
 *
 * @author Jussi Kangasharju
 */
public class Envelope {
    /* SMTP-sender of the message (in this case, contents of From-header. */
    public String Sender;

    /* SMTP-recipient, or contents of To-header. */
    public String Recipient;

    /* Target MX-host */
    public String DestHost;
    public InetAddress DestAddr;

    /* The actual message */
    public Message Message;

    /* Create the envelope. */
    public Envelope(Message message, String localServer)
       throws UnknownHostException {
/* Get sender and recipient. */
Sender = message.getFrom();
Recipient = message.getTo();
```

```
/* Get message.  We must escape the message to make sure that
   there are no single periods on a line.  This would mess up
   sending the mail. */
Message = escapeMessage(message);

/* Take the name of the local mailserver and map it into an
 * InetAddress */
DestHost = localServer;
try {
    DestAddr = InetAddress.getByName(DestHost);
} catch (UnknownHostException e) {
    System.out.println("Unknown host: " + DestHost);
    System.out.println(e);
    throw e;
}
return;
    }

    /* Escape the message by doubling all periods at the beginning of
       a line. */
    private Message escapeMessage(Message message) {
String escapedBody = "";
String token;
StringTokenizer parser = new StringTokenizer(message.Body, "\n", true);

while(parser.hasMoreTokens()) {
    token = parser.nextToken();
    if(token.startsWith(".")) {
token = "." + token;
    }
    escapedBody += token;
}
message.Body = escapedBody;
return message;
    }

    /* For printing the envelope.  Only for debug. */
    public String toString() {
String res = "Sender: " + Sender + '\n';
res += "Recipient: " + Recipient + '\n';
res += "MX-host: " + DestHost + ", address: " + DestAddr + '\n';
res += "Message:" + '\n';
res += Message.toString();

return res;
    }
}
```