Mail User Agent: Simplified Version

This lab is divided into two parts. In the first part, you use telnet to manually send mail through an SMTP mail server. In the second part, you write a Java program that performs the same action.

Part 1: Sending Email with Telnet

When you do this lab, you should try to send email to yourself. This means you need to know the host name of the mail server for your mail domain. To find out this information, you can query DNS for the MX record that maintains information about your mail domain. For example, bob@someschool.edu has mail domain *someschool.edu*. The following command queries DNS for the mail servers responsible for delivering mail to this domain:

```
nslookup -type=MX someschool.edu
```

For the response to this command, there may be several mail servers that will deliver mail to mailboxes in the domain *someschool.edu*. Suppose that the name of one of them is *mx1.someschool.edu*. In this case, the following command will establish a TCP connection to this mail server. (Notice that the port number 25 is specified on the command line.)

```
telnet mx1.someschol.edu 25
```

At this point, the telnet program will allow you to enter SMTP commands, and will display the responses from the mail server. For example, the following sequence of commands would send email to bob from alice

```
HELO alice
MAIL FROM: <alice@crepes.fr>
RCPT TO: <bob@someschool.edu>
DATA
SUBJECT: hello

Hi Bob, How's the weather? Alice.
.
QUIT
```

The SMTP protocol was originally designed to allow people to manually interact with mail servers in a conversational manner. For this reason, if you enter a command with incorrect syntax, or with unacceptable arguments, the server will return a message stating this, and will allow you to try again.

To complete this part of the lab, you should send an email message to yourself and verify that it was delivered.

Part 2: Sending Email with Java

Java provides an API for interacting with the Internet mail system, which is called JavaMail. However, we will not be using this API, because it hides the details of SMTP and socket programming. Instead, you should write a Java program that establishes a TCP connection with a mail server through the socket interface, and sends an email message.

You can place all of your code into the main method of a class called *EmailSender*. Run your program with the following simple command:

```
java EmailSender
```

This means you will include in your code the details of the particular email message you are trying to send.

Here is a skeleton of the code you'll need to write:

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

public class EmailSender
{
   public static void main(String[] args) throws Exception
   {
      // Establish a TCP connection with the mail server.


      // Create a BufferedReader to read a line at a time.
      InputStream is = socket.getInputStream();
      InputStreamReader isr = new InputStreamReader(is);
      BufferedReader br = new BufferedReader(isr);

      // Read greeting from the server.
      String response = br.readLine();
      System.out.println(response);
      if (!response.startsWith("220")) {
         throw new Exception("220 reply not received from server.");
      }

      // Get a reference to the socket's output stream.
      OutputStream os = socket.getOutputStream();

      // Send HELO command and get server response.
      String command = "HELO alice\r\n";
      System.out.print(command);
      os.write(command.getBytes("US-ASCII"));
      response = br.readLine();
      System.out.println(response);
      if (!response.startsWith("250")) {
         throw new Exception("250 reply not received from server.");
      }

      // Send MAIL FROM command.


      // Send RCPT TO command.


      // Send DATA command.


      // Send message data.

      // End with line with a single period.


      // Send QUIT command.

   }
}
```