Programming Assignment 1: E-mail Client

Due Nov. 16, 2011 (before 11:59PM)

In this programming assignment, you are asked to implement a simple e-mail client named *EmailClient* in Java. You will test *EmailClient* against a POP3 and SMTP server running on one of our server machines. So you won't implement the server side of the POP3 and SMTP protocols. For this programming assignment, you are **not** allowed to use any third party POP3 and SMTP client libraries or the POP3 and SMTP specific core or non-core APIs supplied with the JDK including but not limited to `javax.mail`. Since the goal of this assignment is to understand the operation of POP3 and SMTP protocols, using any (third party, core or non-core JDK supplied) APIs providing any level of abstraction specific to the POP3 and SMTP protocol is prohibited. In short, you have to implement the POP3 and SMTP client using just the Socket API of the JDK. If you have any doubt about what to use or not to use, please contact the TAs.

POP3 and SMTP client that you will implement in this programming assignment should be capable of listing, retrieving and sending e-mails using specific POP3 and SMTP requests, which will be explained later. POP3 (Post Office Protocol version 3) is used to retrieve e-mails from an e-mail server using TCP/IP connection and SMTP (Simple Mail Transfer Protocol) is used to send e-mails.

For this programming assignment, you are asked to implement very limited functionality of a typical POP3 and SMTP client. For the details of POP3 and SMTP protocols you can refer to RFC 1939 (http://tools.ietf.org/html/rfc1939) and RFC 821 (http://tools.ietf.org/html/rfc821), respectively. The following sections describe which functions of the POP3 and SMTP protocols should be supported by the *EmailClient* that you will implement.

POP3 Commands:

- **User and Password:**

In order to use the functions below, the *EmailClient* should first log-in. To do this, *EmailClient* should send **"USER *userName*\r\n"** and **"PASS *password*\r\n"** commands to the POP3 Server. The *userName* and *password* should be taken from the user.

- **List:**

To list the messages in the INBOX, you should send **"LIST\r\n"** command to POP3 Server by using Socket API after being authenticated, and the response from the server contains the list of the incoming e-mails.

EmailClient should be able to list the messages in the following format:

INBOX of userName
3 messages

MailId	*Mail Size (bytes)*
4	*6523*
12	*347*

23 125

- **Retrieve (Read) a message:**

To retrieve a messages in the INBOX, you should send **"RETR *mailId*\r\n"** command to POP3 Server by using Socket API, and the response from the server contains the contents of the mail with the id *mailId* which is 23 in the example given below.

EmailClient should be able to show the contents of a specific e-mail in the following format:

MailId: 23
Subject: Welcome
From: user2@wlab.cs.bilkent.edu.tr
Date: 6.10.2011
Message:
Welcome to our POP3 Client.

Have a nice day!

- **Quit:**

EmailClient should be able to quit from the POP3 Server by entering the command **"QUIT\r\n".**

SMTP Commands for sending an e-mail:

EmailClient should be able to send e-mails to other people. An example session is as follows:

To:
>yyy@wlab.cs.bilkent.edu.tr
Subject:
>Happy New Year
Body:
>We wish you a happy new year...
>Take care,
>Alice and Bob.
>.

The input lines (lines that start with ">") should be entered by the user by using Scanner API of the JDK and the other lines should be written by the *EmailClient* using System.out.println(). To send an e-mail, first the *EmailClient* should send HELO message to the SMTP Server by using Socket API of the JDK by sending the command **"HELO *domain\r\n*",** where the sending host identifies its domain. Please refer to "Implementation Hints" section to learn how you can get your domain name in Java. After the HELO message, the *EmailClient* should send the command **"MAIL FROM: *userName@domain\r\n*"** where the *userName is also* a command-line argument. Then the *EmailClient* should send the command **"RCPT TO: *recipient'sMailAddress*\r\n"** You may write any e-mail address for *recipient'sMailAddress,* which the e-mail will be sent to that address. To send the data (content) of the message, the *EmailClient* should send the command **"DATA\r\n"** followed by the headers and the body of the message to the SMTP server. After getting the response like the following from the server **“354 Enter message, ending with "." on a line by itself”,** anything that is sent to the server will be taken as DATA. To end the DATA, the user should enter the line **“.\r\n”**. The “Subject” header will be sent preceding the body of the message, as shown in the following client-server interaction example:

```
    EmailClient                                                  SMTP Server

 -----   HELO mylaptop                                          ---------->
 <---    250 wlab.cs.bilkent.edu.tr Hello localhost [127.0.0.1] ----------
 -----   MAIL FROM: userName@mylaptop                           ---------->
 <---    250 OK                                                 -----------
 -----   RCPT TO: xxx@gmail.com                                 ---------->
 <---    250 Accepted                                           ----------
 -----   DATA\r\n                                               ---------->
 <----   354 Enter message, ending …..                          ----------
 -----   Subject: Happy New Year\r\n                            ---------->
 -----   We wish you a happy new year...\r\n                    ---------->
 -----   Take care,\r\n                                         ---------->
 -----   Alice and Bob.\r\n                                     ---------->
 -----   .\r\n                                                  ---------->
 <---    250 OK id=1RIHm2-0004Kp-IC                             -----------
 -----   QUIT\r\n                                               ---------->
 <---    221 wlab.cs.bilkent.edu.tr closing connection          -----------
```

The lines after “Subject: Happy New Year\r\n” are the body part of the message.

To indicate that the contents of the DATA is finished, i.e., the mail is ready to be sent, the *EmailClient* should send **“.\r\n”** command to the SMTP server as in the above example.

At last, *EmailClient* should terminate the session by sending **“QUIT\r\n”** command to the SMTP Server.

Program Specifications

EmailClient must be a console application (no GUI (Graphical User Interface) is needed). Your program should be run with the following command in a shell:

java EmailClient <userName> <password> <POP3ServerAddress><SMTPServerAddress>

userName, password, POP3ServerAddress and SMTPServerAddress are command line arguments to *EmailClient* and you can find further information about command line arguments from http://download.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html

When *EmailClient* starts, it should connect to the POP3 Server by using **“USER *userName*\r\n”** and **“PASS *password*\r\n”** commands. The *userName* and *password* are given as command line arguments.

If there is an Authentication Error when connecting, *EmailClient* should print **"-ERR Authentication failed."** message, else it should print "**+OK Logged in."** message. If the user can successfully connect to POP3 Server, *EmailClient* should print the list of its functions as follows:

Main Functions:	Explanation:
L	: List the INBOX
R <*MailId*>	: Retrieve the mail with id *MailId*
C	: Compose a new message
F	: Refresh the INBOX
Q	: Quit the program

Main Functions:

L: When user enters "L", *EmailClient* should list the INBOX in the format explained above. If the INBOX is empty, *EmailClient* should print **"INBOX is empty"** message. If there is an error, *EmailClient* should print an appropriate error message.

R <*MailId*>: When the user uses this function, the e-mail with *MailId* should be retrieved in the format explained before. If e-mail with *MailId* cannot be found, the *EmailClient* should print **"-ERR There's no message <*MailId*>."**

C: When the user enters "C", he/she will send the message as explained in the "**SMTP Commands for sending an e-mail:"** section. User should enter the line **".\r\n"** to indicate that the DATA message is finished. If there is an error in any step of the sending i.e. HELO, MAIL FROM, RCPT TO or DATA, *EmailClient* should print an appropriate error message, else it should print **"E-mail is successfully sent."**

F: In order to refresh the INBOX i.e. to see new incoming messages, the user should enter "F". Because the LIST command does not do this. To refresh the INBOX, the *EmailClient* should first quit the POP3 session by sending **"QUIT\r\n"** to the POP3 Server and restart the session by sending **"USER *userName*\r\n"** and **"PASS *password*\r\n"** commands to the POP3 Server. At last, *EmailClient* should do the same operation with "L" command i.e. it lists the INBOX.

Q: This function terminates the program by sending **"QUIT\r\n"** message to POP3 Server and closing Sockets, Readers, Writers etc. , if the QUIT command is successful, *EmailClient* will print "**+OK Logging out. "** and the program terminates, if there is an unexpected error, *EmailClient* will print an appropriate error message.

Important: If the *EmailClient* receives any error message from POP3/SMTP Servers in any step, it should print an appropriate error message to inform the user.

References and Some Implementation Hints:

- Please do not use hard-coded values for any data to be entered by the user and let the users enter such data via standard input. For instance the user should be able to request resources from servers other than `wlab.cs.bilkent.edu.tr` by entering appropriate URLs in your client program -> connect to POP3/SMTP servers other than wlab.cs.bilkent.edu.tr.

- You can learn your *domain* in Java using the following commands:

```
InetAddress addr = InetAddress.getLocalHost();

String cname = addr.getCanonicalHostName();
```

Some Testing Hints:

- When you establish a TCP connection with the POP3 or SMTP Servers, you should first wait for the greeting message from the server before sending any other commands, otherwise you may encounter a synchronization error and during the session after each command is sent, you should wait for the server's response before proceeding with next commands.

- You may use the following POP3 and SMTP Servers to test your client:

  POP3 Server Address: wlab.cs.bilkent.edu.tr
  POP3 Port: 110
  userName: cs421pa1
  password: pa1

  SMTP Server Address: wlab.cs.bilkent.edu.tr
  SMTP Port: 25
  Test user: cs421pa1@test.com

  While testing your SMTP client, if you specify cs421pa1@test.com as the recipient's address, then the e-mail will be delivered to the cs421pa1 POP3 account.

- In order to access to the servers, you should be in the Bilkent University Network, If you are not in the Bilkent University, you should use VPN. http://vpn.bcc.bilkent.edu.tr/

- You may send the e-mails to your own e-mail accounts for testing and you may need to check your Spam Folder to see your e-mail.

Submission Rules:

You need to apply all the following rules in your submission. Any violation of these rules will result in discarding of your submission.

❑ You can do the assignment either individually or in groups of two. You can also choose your partner from the other section. If you implement the assignment as a group, please make a single submission for the group using the *CS421 File Submission System* as explained below.

❑ The assignment should be submitted via the *CS421 File Submission System* available at http://wlab.cs.bilkent.edu.tr/SubmissionWeb/. Please note that the submission system may not accept those files that do not conform to the submission rules explained in this section or to the programming rules explained in previous sections. The submission system shall be running even after the submission deadline. If you have done the assignment on your own, then you should leave the second text box (titled as "Student id #2") empty and just put your Bilkent ID in the text box titled as "Student id #1". If you have done the assignment as a group, then you should fill in both text boxes with the participating students' Bilkent IDs.

❑ Please use your **Bilkent student ID**s (and not your TC IDs) at *CS421 File Submission System.*

❑ All the files must be submitted in a **zip** file whose name is obtained by concatenating your name, your surname and your Bilkent ID. If your name is Ali Velioglu and your Bilkent ID is 20209999 then the zip archive's file name should be "AliVelioglu20209999" (without the quotation marks). For groups of two people, the name and the student ID of the second participant of the group should be concatenated to the first student's name and Bilkent ID. Following the example given above, if the second student's name is Veli Alioglu and his ID is 20208888, then the name of the zip archive should be **AliVelioglu20209999VeliAlioglu20208888.zip**.

❑ The file name should not contain any Turkish characters or spaces. The file must be a .zip file, not a .rar file or any other compressed file.

❑ Any other methods (e-mail/disk/CD/DVD) of submission will not be accepted.

❑ Please try to make a single submission. In case of duplicate submissions, the last one shall be graded.

❑ The programming language used for implementation must be Java. The virtual machine used to test the code will be version 1.6. Your sources should compile cleanly (warnings are OK). If your code does not compile, you will either receive **no credit** or your grade will be penalized **severely**.

❑ All of the files must be in the root of the zip file, directory structures are not allowed.

❑ Please note that this also disallows organizing your code into Java packages.

❑ In addition to the source code, the archive should contain a README file explaining your implementation, detailing any issues and/or other implementation details. If your program is incomplete, please clearly describe what works and what does not. This file should also serve as a user manual.

❑ The archive should NOT contain:

- Any class files or other executables
- Any third party library archives (i.e. jar files)
- Any text files other than plain text files
- Project files used by *IDE*s (e.g. JCreator, JBuilder, SunOne, Eclipse, Idea or NetBeans...) You may, and are encouraged to, use these programs while developing, but the end result must be a clean, IDE-independent program.

❑ The standard rules for plagiarism and academic honesty apply, if in doubt refer to "Student Disciplinary Rules and Regulation", items 7.j, 8.l and 8.m.