Lab 11

Spam, Spam, Spam: part 2

Objective. To gain experience with Streams. **(This lab is due at the end of lab today.)**

POP Mail Connections. This week's assignment will give you the opportunity to practice working with Streams. You will write the networking part of last week's "Spam" program. This week you will complete the `MailConnection` class, which was provided to you as a library last week. The class implements the POP mail protocol by opening a socket and input and output streams, writing messages to the server, and interpreting the responses.

Before getting into the details of `MailConnection`, let us quickly review of the mail protocol. What follows is an example session involving connecting to a mail server.

```
moo:~/] telnet cortland.cs.williams.edu 110
Trying 137.165.8.5...
Connected to cortland.cs.williams.edu.
Escape character is '^]'.
+OK cortland.cs.williams.edu Cyrus POP3 v2.2.12-OS X 10.4.8 server ready (*)
USER freund
+OK Name is a valid mailbox
PASS --------
+OK Mailbox locked and ready
STAT
+OK 147 184742
TOP 13
-ERR Missing argument
TOP 13 0
+OK Message follows
Return-Path: <heeringa@cs.williams.edu>
Received: from durham.cs.williams.edu (durham.cs.williams.edu [137.165.8.108])
(authenticated bits=0)
by eringer.cs.williams.edu (8.13.8/8.13.8) with ESMTP id m3JEGqF3033894
(version=TLSv1/SSLv3 cipher=AES128-SHA bits=128 verify=NO);
Sat, 19 Apr 2008 10:16:52 -0400 (EDT)
(envelope-from freund@cs.williams.edu)
... other received info omitted ...
Message-Id: <C7FAEB46-A7AD-4915-8CBB-D140598A7DF7@cs.williams.edu>
From: Brent Heeringa <heeringa@cs.williams.edu>
To: freund@cortland.cs.williams.edu
Content-Type: text/plain; charset=US-ASCII; format=flowed
Content-Transfer-Encoding: 7bit
Mime-Version: 1.0 (Apple Message framework v919.2)
Subject: Programming Project 2
Date: Sat, 19 Apr 2008 10:16:51 -0400

.
QUIT
+OK
Connection closed by foreign host.
```

Your `MailConnection` class will need to encode most of this protocol. We will provide you with our solution to last week's `SpamFilter` class to use with your new `MailConnection` class.

See the handouts web page to obtain a copy of the starter code.

The Design

The `MailConnection` class manages the communication with a mail server. We begin with the descriptions of the private methods we have provided in the class for your use:

`private void netPrintln(String s)`: Writes a string to the mail server's output stream.

`private String netReadLine()`: Reads a line from the mail server.

`private void close()`: Closes the socket. Must be called if the connection was not established in the constructor.

Note: We have included a constant, `DEBUG` that is initially set to be true. It causes anything written to the mail server by `netPrintln` to be echoed to the console window in BlueJ. Similarly anything read from the mail server by `netReadLine` is also echoed to the console window. It should be set to false before you turn in your program.

We also provide two public methods: `disconnect()`, which closes the connection after a successful mail session, and `isConnected()`, which returns a boolean indicating whether the connection is currently active with the mail server. These are the methods that you called last week from your `SpamFilter` class.

The following is a description of the constructor and methods that you must complete.

`public MailConnection (String host, String userName, String password)`

Create a mail connection to a host for a specific user. If everything works, set connect to true and return. If it fails for any reason, pop up a dialog box with a helpful message and call the `close()` method to close the connection

We have provided the code to create a new socket as well as input and output streams (called `input` and `output`) that use that socket. You should read the response from the mail server (using the private `netReadLine()` method) starting with the line marked (*) in the sample above. If that line starts with "+OK" then go through the protocol identifying the USER and password. If all of the responses start with "+OK" then set boolean variable `connected` to true. Otherwise, pop up a dialog box providing information on what went wrong (see the provided code).

`public int getNumMsgs()`

Returns the number of messages in the mailbox you are connected to. This returns 0 if there is no active connection.

This method will send a STAT command to the server and extract the number of messages available from the line the server sends back. It will then return the integer obtained from that string.

Recall that the method `Integer.parseInt(s)` converts the digits in the `String s` to an integer and returns that value. For example, `Integer.parseInt("33")` returns the value 33.

`public String header (int msg)`

Returns the headers of a mail message identified by the number passed in. Unlike Java, mailboxes number messages beginning with 1 and go up to the number of messages contained in the mailbox. To obtain the header for message i, you must send "TOP i 0" to the mail server.

The `header` method returns an empty string if it is called when there is no connection or if the `msg` parameter is less than 1 or greater than the actual number of messages.

This method should use `readResponse()` (see below) to obtain the String consisting of the full header.

`private String readResponse()`

Reads a multi-line response from the mail server. Returns an empty string if there is no connection.

If the connection is alive then it should successively read new lines from the input stream and concatenate them together (inserting new line characters at the end of lines) until the results of a read are either null or ".".

Be sure to test the constructor and each method you are writing carefully before going on to the next one. To make this easy, the version of our solution to last week's lab which we have included does not perform the entire process of fetching and filtering the mail as soon as you click one button. Instead, the first time you click the button it merely connects to the server. The next time you click, it gets the number of available messages. After that, each message downloads one message until they have all been fetched. Once your program is working, you can change our program to behave more like last week's lab by removing one line in the begin method as indicated by the comments in our code.

Submitting Your Work

Once you have saved your work in BlueJ, please perform the following steps to submit your assignment. Don't forget to change the name of your folder to include your last name.

- First, return to the Finder.
- Click "Go" and select "Connect to Server."
- For the server address, type in "Cortland" and click "Connect."
- Select the button next to "Guest" and click "Connect."
- A selection box should appear. Select "Courses" and click "Ok."
- You should now see a Finder window with a "cs134" folder. Open this folder.
- You should now see the drop-off folders for the three labs sections. Drag your program folder into the appropriate lab section folder. When you do this, the Mac will warn you that you will not be able to look at this folder. That is fine. Just click "OK."
- Log off of the computer before you leave.

This lab is due at the end of lab today. If you submit and later discover that your submission was flawed, you can submit again. We will grade the latest submission made before the 11 p.m. deadline. The Mac will not let you submit again unless you change the name of your folder slightly. It does this to prevent another student from accidentally overwriting one of your submissions. Just add something to the folder name (like "v2") and the resubmission will work fine.

Grading Guidelines

As on labs, we will evaluate your program for both style and correctness. Here are some specific items to keep in mind and focus on while writing your program:

Style
Descriptive comments
Good names
Good use of constants
Appropriate formatting

Design
General correctness/design/efficiency issues
conditionals and loops
Parameters, variables, and scoping
Good correct use of arrays
Good use of private methods
Good use of strings

Correctness
Constructor
getNumMessages
header
readResponse