COMP2121 Assignment 1

Due: September 1st, 2013

The goal of this project is to implement an SMTP server in Java that supports multi-threading and stores emails into local files. The main method should be located in `MySMTPServer.java` file. For the submission, all necessary files should be put into a directory named with your own unikey, and a zip archive of this directory should be submitted on the eLearning website of COMP2121.

No optional packages that are not part by default of the JDK, like `JavaMail` can be used.

Task 1 Multithreaded server

The first task consists in writing a multi-threaded server, `MySMTPServer.java`. (Although you will probably have multiple classes located in multiple java files, do not use specific java packages.) The server executes an infinite loop to accept new connections requested by clients. At each new connection, the server spawns a new thread that will handle the incoming request.

Each request consists in a client passing an email information. This information is passed using various commands as described in RFC 2821 (http://www.ietf.org/rfc/rfc2821.txt). The server should at least support commands `HELO`, `MAIL FROM`, `RCPT TO`, `DATA`, and `QUIT`. The communication is both ways: the server must respond appropriately to the command executed by the client upon success or failure.

At the end of a communication the server thread handling the request must appropriately close the connection.

Task 2 Message encoding

Each email must be formatted in MIME format as specified in RFC 2045 (http://tools.ietf.org/html/rfc2045). In particular, the client must have the ability to at least specify the sender, the recipient, the date, the subject of the email and a body in plain-text format.

Task 3 Storing information

Each message will be given a unique sequence number – the sequence number should be monotonically increasing. Each server thread handling a request records the email information into a separate file with a unique filename located in an `emails` directory. (This `emails` directory should be located in the same “unikey” directory as the file `MySMTPServer.java`.) Typically, the sequence number can be used to generate a new filename. The file must represents the email information in text as follows:

```
Message <sequence-number>
From: <sender>
To: <receiver>
Date: <date>
Subject: <subject>
Body: <body>
```

Task 4 Test

The main method should be located in `MySMTPServer.java` file, so that the server can be run using the following command (on UNIX-like command line):

```
$ java MySMTPServer &
```

During the development phase, the server could be tested with a client using the usual `telnet` command:

```
$ telnet <mysmtpserver.mymachine.usyd.edu.au> <port-number>
```

Note that 127.0.0.1 or `localhost` can be used instead of `mysmtpserver.mymachine.usyd.edu.au` to refer to the local machine. The client and server can be tested on the same machine, however, it would be good to make sure that the client can successfully connect to the server running on a remote machine (within the same network). To this end, you would need to obtain the IP address or hostname of the remote machine: one can use the `ifconfig` and `hostname` commands (for UNIX-like environment) or `ipconfig` command (Windows).

Write a README text file with three sections COMPILATION, RUN, TEST indicating how the server can be compiled, run and tested.

Not only should the server pass the series of commands provided in the Lab 2 about "Routing and communicating" but it should also support erroneous series of commands given by the clients and return appropriate error message.

To do some basic testing of your program, use the archive located at http://sydney.edu.au/engineering/it/~gramoli/doc/ens/smtp/testing.tgz. First, make sure your server is running on your machine and listens to port 6013. Then unarchive the files, compile and run the test client as follows:

```
tar -xvzf testing.tgz
cd testing
javac -cp junit.jar:org.hamcrest.core_1.1.0.jar SMTPTest.java
java -cp .:junit.jar:org.hamcrest.core_1.1.0.jar SMTPTest 6013 PATH_TO_YOUR_SERVER
```

The following output indicates that your server successfully passed the two basic tests:

```
Successes:
    + checkInitiallyApproximateContent
    + labMessageCorrect
```

This does not mean you assignment is successful, as more elaborate tests will be used for marking.

Task 5 Address verification

Once the above steps are completed, verify that the email address given as the sender is correct in that it belongs to the domain of the University of Sydney. Make sure the server tells the client whether the address is accepted (after the verification).

For the sake of simplicity, the server should return successfully if the email address if of the form local-part “@” domain where domain ends with "usyd.edu.au". (The server could arbitrarily return a success or an error when domain ends with other acceptable USyd domain suffix, like "sydney.edu.au".)

More information about the local-part can be found in http://tools.ietf.org/html/rfc5322#section-3.4.1