```
/**
  * VerifyUser.java
  * Created by Matthew A. Crist on March 28, 2013.
  * This file is designed to replace the verify-user.sh script for more flexibility.
  * PREVIOUS DOCUMENTATION:
   # verify-user.sh
        # Created on March 23, 2013 by Matthew A. Crist.
        # This file will invoke the script required to verify that a user should
        # have access to an inbox and open a connection to send those files to the
        # client.
        #
        # NO LONGER VALID vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
        # THIS MODULE RELIES HEAVILY ON THE CREATED DIRECTORIES FOR MAIL FOLDERS
        # ON USER REGISTRATION.  IF THIS FOLDER DOES NOT EXIST, THE VERIFICATION
        # PROCESS IS POINTLESS.  A RESPONSE WILL BE USED ON THE SERVER SIDE
        # ACCEPT(user.verify) or REJECT(user.verify) TO RESPOND BACK TO THE CLIENT
        # IF IT IS ACCEPTABLE TO SEND/RECIEVE THE EMAIL IN THEIR INBOX.  WHEN THAT
        # EMAIL IS SENT, IT IS REMOVED FROM THE SERVER ENTIRELY.
        # ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
        #
        # CHANGE LOG:
        # ------------------------------------------------------------------------
        # 2013-03-31    -       Changed the purpose of the script to acquire emails.
        # 2013-03-29    -       Converted file from shell script to java program.
        # 2013-03-23    -       Initial conception of this file.
*/

package modules.user.verify;

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

```
public class VerifyUser {

        public static void main(String[] args) {

                boolean listening = true;


                try {

                        ServerSocket server = new ServerSocket(20002);


                        while(listening) {

                                System.out.println("User verification bound to post 20002.\n");

                                Socket client = server.accept();

                                System.out.println("User verification request accepted.  Processing...");


                                BufferedWriter out = new BufferedWriter(new OutputStreamWriter(client.      ↙
    getOutputStream()));

                                BufferedReader in  = new BufferedReader(new InputStreamReader(client.       ↙
    getInputStream()));


                                String input, store = "", request = "";


                                // For all input received, write it to the request buffer.

                                while((input = in.readLine()) != null) {

                                        request += input;

                                }       // end while loop


                                // Need to acquire the contents of the address book

                                BufferedReader reader = new BufferedReader(new FileReader("store/address-   ↙
    book/address-book.xml"));

                                while((input = reader.readLine()) != null) {

                                        store += input;

                                }       // end while loop


                                // The ACL2 command that will be executed.

                                String acl2 = "(include-book \"modules/user/verify/verify-user\")" +

                                                                  "(in-package \"ACL2\")" +

                                         "(set-state-ok t)" +

                                                                  "(set-guard-checking :none)" +
                                              "(testUser \"" + request + "\" \"" + store + "\" state)";

                                // Proceed to spur the ACL2 process and place a wrapper on the IO
                                System.out.println("Executing ACL2 runtime for User Verification...");
                                ProcessBuilder processBuilder = new ProcessBuilder("acl2");
                                File log = new File("logs/user/verify/acl2_log.txt");
                                processBuilder.redirectErrorStream(true);
                                processBuilder.redirectOutput(log);
                                Process process = processBuilder.start();
                                PrintWriter procIn = new PrintWriter(process.getOutputStream());

                                // Write the ACL2 to the process, exit ACL2 and close the socket
                                procIn.println(acl2);
                                procIn.println("(good-bye)");
                                procIn.flush();
                                procIn.close();

                                // Flag for the security check
                                boolean proceed = false;

                                // Read in the contents of the file and see if one line contains ACCEPT
                                BufferedReader tRead = new BufferedReader(new FileReader("incoming/user/    ↙
    verify/server-action.xml"));
                                String failBuffer = "";

                                System.out.println("Determining if login information is correct.");

                                while((input = tRead.readLine()) != null) {
                                        // Because I am lazy and don't want to parse the XML
                                        if(input.contains("ACCEPT")) {
                                                proceed = true;
                                                System.out.println("User verified successfully!");
                                        } else {
                                                failBuffer += request;
                                        }       // end if-else
                                }       // end while


                                tRead.close();


                                // Determine if the login ws good!
                                if(proceed) {
                                        String name   = request.substring(request.indexOf("<name>")+6,      ↙
    request.indexOf("</name>"));
                                        String domain = request.substring(request.indexOf("<domain>")+8,    ↙
    request.indexOf("</domain>"));


                                        File emailDirectory = new File("store/email/" + domain + "/" + name↙
     + "/");


                                        System.out.println("Sending emails from " + emailDirectory.getPath  ↙
    ());


                                        // It better be a damn directory, but incase someone has leet hacks
                                        if(emailDirectory.isDirectory()) {
                                                File[] emails = emailDirectory.listFiles();


                                                String transmit = "";


                                                System.out.println("Writing emails to client.");
                                                // Read the contents of each email and transmit them to the↙
     client.
                                                for(int i = 0; i < emails.length; i++) {
                                                   if(!emails[i].isHidden()){
                                                           BufferedReader eRead = new BufferedReader(new    ↙
    FileReader(emails[i]));
                                                           String eTmp = "";
                                                           while((eTmp = eRead.readLine()) != null) {
                                                                   transmit += eTmp;
                                                           }    // end while


                                                           eRead.close();
                                                           // Write email to client
                                                           out.write(transmit);
                                                           out.newLine();
                                                           // Reset the buffer
                                                           transmit = "";
                                                   }    // end if
                                                }       // end for

                                                out.write("END");
                                        } else {
                                                // Create the directory since it should be there!!!
                                                emailDirectory.mkdirs();

                                                System.out.println("There was an internal server error:    ↙
    Inbox does not exist!\n");
                                                out.write("END");
                                        }       // end if-else
                                } else {
                                        out.write(failBuffer);
                                        out.newLine();
                                        out.write("END");
                                }       // end if-else

                                // Close our connections
                                out.close();
                                in.close();
                                client.close();
                        }       // end while loop

                        server.close();
                        System.exit(0);
                } catch(Exception e) {
                        System.out.println(e.getMessage());
                        e.printStackTrace();
```

```
                }       // end try/catch

        }       // end function main

}       // end class VerifyUser
```