```
/**
 * RegisterUser.java
 * Created by Matthew A. Crist on March 29, 2013.
 *
 * This class is a revamp of the old shell scripts to allow for better
 * server integration.
 *
###############################################################################
# Created by Matthew A. Crist on March 8, 2013.
# This file will provide the actions required to register a user to the address
# book.  Folders will be created for a store.
#
# Server listening port: 20001
#
# CHANGE LOG:
#------------------------------------------------------------------------------
# 2013-03-29    -   Migrated file into java source.
# 2013-03-17    -   Added directory creation on user registration.
# 2013-03-09    -   Removed grep validation for request structure added if check
# 2013-03-09    -   Added grep validation for request structure
# 2013-03-09    -   Encapsulated all values in function register_user
# 2013-03-08    -   Added support for network connectivity.
# 2013-03-08    -   Initial conception of this file.
###############################################################################*/

package modules.user.register;

import java.io.*;
import java.util.*;
import java.net.*;
import java.nio.channels.FileChannel;

public class RegisterUser {
    public static void main(String[] args) {
        boolean listening = true;

        try {
            // Acquire the listening port for connection to client.
            ServerSocket server = new ServerSocket(20003);

            while(listening) {
                // Wait until the client connects
                Socket client = server.accept();

                // Handles for input and output streams relating to the socket connection
                PrintWriter out = new PrintWriter(client.getOutputStream(), true);
                BufferedReader in = new BufferedReader(new InputStreamReader(client.getInputStream()));

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

                // Read the input from the connection
                while((input = in.readLine()) != null) {
                    System.out.println(input);
                    request += input;
                }   // end while

                // Read the contents of the address-book currently stored
                BufferedReader reader = new BufferedReader(new FileReader("store/address-book/address-book.↙
    xml"));
                while((input = reader.readLine()) != null) {
                    store += input;
                }   // end while

                // The ACL2 code to execute.
                String acl2 = "(include-book \"modules/user/register/register-user\")" +
                              "(in-package \"ACL2\")"            +
                              "(registerUser \"" + request + "\" \"" + store + "\" state)";
            // Initialize ACL2 and dump its output to the log
            System.out.println("Executing ACL2 runtime for RegisterUser...");
            ProcessBuilder processBuilder = new ProcessBuilder("acl2");
            File log = new File("logs/user/register/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, close ACL2
            procIn.println(acl2);
            procIn.println("(good-bye)");
            procIn.flush();
            procIn.close();
            System.out.println("FinishedACL2");
            // Old store is old address-book file and new store is newly generated
            File oldStore = new File("store/address-book/address-book.xml");
            File newStore = new File("store/address-book/temp_address-book.xml");

            // Response header information
            String response = "<?xml version='1.0'?>"                        +
                              "<!DOCTYPE response SYSTEM 'dtd/reponse.dtd'>" +
                              "<response>";

            //System.out.println("Old Store Size: " + oldStore.length());
            //System.out.println("New Store Size: " + newStore.length());

            // Determine if there was a change.
            // If entry was added, the length > that old length.
            if(oldStore.length() < newStore.length()) {
                // Replace old file with new file
                FileChannel src = new FileInputStream(newStore).getChannel();
                FileChannel dest = new FileOutputStream(oldStore).getChannel();
                dest.transferFrom(src, 0, src.size());
                src.close();
                dest.close();

                // Extract data from request XML
                String name   = request.substring(request.indexOf("<name>")+6, request.indexOf("</name> ↙
"));
                String domain = request.substring(request.indexOf("<domain>")+8, request.indexOf("</    ↙
domain>"));

                // Create the store directory for the user's emails
                File storeDirectory = new File("store/email/" + domain + "/" + name + "/");
                storeDirectory.mkdirs();

                response += "<message>ACCEPT</message>";
            } else {
                // Remove new file as it is pointless
                newStore.delete();
                response += "<message>REJECT</message>";
            }   // end if-else

            response += "</response>";

            // Writeback the response to the client
            //out.print(response);
            out.flush();

            // Close all streams
            out.close();
            in.close();
            client.close();
        }   // end while
    } catch(Exception e) {
            e.printStackTrace();
        }   // end try/catch
    }   // end function main
}   // end class RegisterUser
```