```java
// joi/7/juno/LoginInterpreter.java
//
//
// Copyright 2003 Ethan Bolker and Bill Campbell

import java.util.*;

/**
 * Interpreter for Juno login commands.
 *
 * There are so few commands that if-then-else logic is OK.
 *
 * @version 7
 */

public class LoginInterpreter
{
    private static final String LOGIN_COMMANDS =
        "help, register, <username>, exit";

    private Juno     system;  // the Juno object
    private Terminal console; // for i/o

    /**
     * Construct a new LoginInterpreter for interpreting
     * login commands.
     *
     * @param system the system creating this interpreter.
     * @param console the Terminal used for input and output.
     */

    public LoginInterpreter( Juno system, Terminal console)
    {
        this.system  = system;
        this.console = console;
    }

    /**
     * Set the console for this interpreter.  Used by the
     * creator of this interpreter.
     *
     * @param console the Terminal to be used for input and output.
     */

    public void setConsole( Terminal console )
    {
        this.console = console;
    }

    /**
     * Simulates behavior at login: prompt.
     * CLI stands for "Command Line Interface".
     */

    public void CLILogin()
    {
```

```java
        welcome();
        boolean moreWork = true;
        while( moreWork ) {
            moreWork = interpret( console.readLine( "Juno login: " ) );
        }
    }

    // Parse user’s command line and dispatch appropriate
    // semantic action.
    //
    // return true unless "exit" command or null inputLine.

    private boolean interpret( String inputLine )
    {
        if (inputLine == null) return false;
        StringTokenizer st =
            new StringTokenizer( inputLine );
        if (st.countTokens() == 0) {
            return true; // skip blank line
        }
        String visitor = st.nextToken();
        if (visitor.equals( "exit" )) {
            return false;
        }
        if (visitor.equals( "register" )) {
            register( st );
        }
        else if (visitor.equals( "help" )) {
            help();
        }
        else {
            User user = system.lookupUser(visitor);
            new Shell( system, user, console );
        }
        return true;
    }

    // Register a new user, giving him or her a login name and a
    // home directory on the system.
    //
    // StringTokenizer argument contains the new user’s login name
    // followed by full real name.

    private void register( StringTokenizer st )
    {
        String userName = st.nextToken();
        String realName = st.nextToken("").trim();
        Directory home  = new Directory( userName, null,
                                         system.getUserHomes() );
        User user = system.createUser( userName, home, realName );
        home.setOwner( user );
    }

    // Display a short welcoming message, and remind users of
    // available commands.

    private void welcome()
    {
        console.println( "Welcome to " + system.getHostName() +
                         " running " + system.getOS() +
                         " version " + system.getVersion() );
        help();
    }

    // Remind user of available commands.

    private void help()
    {
        console.println( LOGIN_COMMANDS );
        console.println("");
    }
}
```