```
//  joi/6/juno/Juno.java
//
//
//  Copyright 2003 Bill Campbell and Ethan Bolker

import java.io.*;
import java.util.*;
import java.lang.*;

/**
 * Juno (Juno's Unix NOt) mimics a command line operating system
 * like Unix.
 * <p>
 * A Juno system has a name, a set of Users, a JFile system,
 * a login process and a set of shell commands.
 *
 * @see User
 * @see JFile
 * @see ShellCommand
 *
 * @version 6
 */

public class Juno
{
    private final static String os      = "Juno";
    private final static String version = "6";

    private String    hostName;   // host machine name
    private Map       users;      // lookup table for Users
    private Terminal  console;    // for input and output

    private Directory slash;      // root of JFile system
    private Directory userHomes;  // for home directories

    private ShellCommandTable commandTable; // shell commands

    /**
     * Construct a Juno (operating system) object.
     *
     * @param hostName  the name of the host on which it's running.
     * @param echoInput should all input be echoed as output?
     */

    public Juno( String hostName, boolean echoInput )
    {
        // initialize the Juno environment ...

        this.hostName = hostName;
        console       = new Terminal( echoInput );
        users         = new TreeMap();           // for registered Users
        commandTable  = new ShellCommandTable(); // for shell commands

        // the file system

        slash     = new Directory( "", null, null );
```



```
        User root = new User( "root", slash, "Rick Martin" );
        users.put( "root", root );
        slash.setOwner(root);
        userHomes = new Directory( "users", root, slash );

        // create, then start a command line login interpreter

        LoginInterpreter interpreter
            = new LoginInterpreter( this, console );
        interpreter.CLILogin();
    }

    /**
     * The name of the host computer on which this system
     * is running.
     *
     * @return the host computer name.
     */

    public String getHostName()
    {
        return hostName;
    }

    /**
     * The name of this operating system.
     *
     * @return the operating system name.
     */

    public String getOS()
    {
        return os;
    }

    /**
     * The version number for this system.
     *
     * @return the version number.
     */

    public String getVersion()
    {
        return version;
    }

    /**
     * The directory containing all user homes for this system.
     *
     * @return the directory containing user homes.
     */

    public Directory getUserHomes()
    {
        return userHomes;
    }
```

```
113
114     /**
115      * The shell command table for this system.
116      *
117      * @return the shell command table.
118      */
119
120     public ShellCommandTable getCommandTable()
121     {
122         return commandTable;
123     }
124
125     /**
126      * Look up a user by user name.
127      *
128      * @param username the user's name.
129      * @return the appropriate User object.
130      */
131
132     public User lookupUser( String username )
133     {
134         return (User) users.get( username );
135     }
136
137     /**
138      * Create a new User.
139      *
140      * @param userName the User's login name.
141      * @param home her home Directory.
142      * @param realName her real name.
143      * @return newly created User.
144      */
145
146     public User createUser( String userName, Directory home,
147                             String realName )
148     {
149         User newUser = new User( userName, home, realName );
150         users.put( userName, newUser );
151         return newUser;
152     }
153
154     /**
155      * The Juno system may be given the following command line
156      * arguments.
157      * <pre>
158      *
159      * -e:         Echo all input (useful for testing).
160      *
161      * -version:   Report the version number and exit.
162      *
163      * [hostname]: The name of the host on which
164      *             Juno is running (optional).
165      * </pre>
166      */
167
168     public static void main( String[] args )
```



```
169     {
170         // Parse command line options
171
172         boolean echoInput = false;
173         String  hostName  = "mars";
174
175         for (int i=0; i < args.length; i++) {
176             if (args[i].equals("-version")) {
177                 System.out.println( os + " version " + version );
178                 System.exit(0);
179             }
180             if (args[i].equals("-e")) {
181                 echoInput = true;
182             }
183             else {
184                 hostName = args[i];
185             }
186         }
187
188         // create a Juno instance, which will start itself
189
190         new Juno( hostName, echoInput );
191     }
192 }
```