

```

1 // fo1/7/juno/Juno.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5
6 import java.io.*;
7 import java.util.*;
8 import java.lang.*;
9
10 /**
11  * Juno (Juno's Unix NOC) mimics a command line operating system
12  * like Unix.
13  * <p>
14  * A Juno system has a name, a set of Users, a JFile system,
15  * a login process and a set of shell commands.
16
17  * @see User
18  * @see JFile
19  * @see ShellCommand
20
21  * @version 7
22  */
23
24 public class Juno
25 {
26     private final static String os      = "Juno";
27     private final static String version = "7";
28
29     private String  hostName; // host machine name
30     private Map    users; // lookup table for Users
31     private Terminal console; // for input and output
32
33     private Directory slash; // root of JFile system
34     private Directory userHomes; // for home directories
35
36     private ShellCommandTable commandTable; // shell commands
37
38     /**
39      * Construct a Juno (operating system) object.
40
41      * @param hostName the name of the host on which it's running.
42      * @param echoInput should all input be echoed as output?
43      */
44
45     public Juno( String hostName, boolean echoInput )
46     {
47         // initialize the Juno environment ...
48
49         this.hostName = hostName;
50         console       = new Terminal( echoInput );
51         users         = new TreeMap(); // for registered Users
52         commandTable = new ShellCommandTable(); // for shell commands
53
54         // the file system
55         slash = new Directory( "", null, null );
56

```

```

57     User root = new User( "root", slash, "Rick Martin" );
58     users.put( "root", root );
59     slash.setOwner( root );
60     userHomes = new Directory( "users", root, slash );
61
62     // create, then start a command line login interpreter
63     LoginInterpreter interpreter
64     = new LoginInterpreter( this, console );
65     interpreter.CLIlogin();
66
67 }
68
69 /**
70  * The name of the host computer on which this system
71  * is running.
72
73  * @return the host computer name.
74
75  */
76
77     public String getHostName()
78     {
79         return hostName;
80     }
81
82     /**
83      * The name of this operating system.
84
85      * @return the operating system name.
86
87  */
88
89     public String getOS()
90     {
91         return os;
92     }
93
94     /**
95      * The version number for this system.
96
97      * @return the version number.
98
99  */
100
101     public String getVersion()
102     {
103         return version;
104     }
105
106     /**
107      * The directory containing all user homes for this system.
108
109      * @return the directory containing user homes.
110
111  */
112
113     public Directory getUserHomes()
114     {
115         return userHomes;
116     }

```

```

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

```