

```

1 // fo1/7/juno/Shell.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5
6 import java.util.*;
7
8 /**
9  * Models a shell (command interpreter)
10  *
11  * The Shell knows the (Juno) system it's working in,
12  * the User who started it,
13  * and the console to which to send output.
14  *
15  * It keeps track of the the current working directory ( . ) .
16  *
17  * @version 7
18  */
19
20 public class Shell
21 {
22     private Juno system; // the operating system object
23     private User user; // the user logged in
24     private Terminal console; // the console for this shell
25     private Directory dot; // the current working directory
26
27     /**
28      * Construct a login shell for the given user and console.
29      *
30      * @param system a reference to the Juno system.
31      * @param user the User logging in.
32      * @param console a Terminal for input and output.
33      */
34
35     public Shell( Juno system, User user, Terminal console )
36     {
37         this.system = system;
38         this.user = user;
39         this.console = console;
40         dot = user.getHome(); // default current directory
41         CLIShell();
42     }
43
44     // Run the command line interpreter
45
46     private void CLIShell()
47     {
48         boolean moreWork = true;
49         while(moreWork) {
50             moreWork = interpret( console.readLine( getPrompt() ) );
51         }
52         console.println("goodbye");
53     }
54
55     // Interpret a String of the form
56     // shellcommand command-arguments

```

```

57 //
58 // return true, unless shell command is logout.
59
60 private boolean interpret( String inputLine )
61 {
62     StringTokenizer st = stripComments(inputLine);
63     if (st.countTokens() == 0) { // skip blank line
64         return true;
65     }
66     String commandName = st.nextToken();
67     ShellCommand commandObject =
68         system.getCommandTable().lookup( commandName );
69     if (commandObject == null ) {
70         console.errPrintln("Unknown command: " + commandName); // EEE
71         return true;
72     }
73     try {
74         commandObject.doit( st, this );
75     }
76     catch (ExitShellException e) {
77         return false;
78     }
79     catch (BadShellCommandException e) {
80         console.errPrintln( "Usage: " + commandName + " " +
81             e.getCommand().getArgString() ); // EEE
82     }
83     catch (JunoException e) {
84         console.errPrintln( e.getMessage() ); // EEE
85     }
86     catch (Exception e) {
87         console.errPrintln( "you should never get here" ); // EEE
88         console.errPrintln( e.toString() ); // EEE
89     }
90     return true;
91 }
92
93 // Strip characters from '#' to end of line, create and
94 // return a StringTokenizer for what's left.
95
96 private StringTokenizer stripComments( String line )
97 {
98     int commentIndex = line.indexOf('#');
99     if (commentIndex >= 0) {
100         line = line.substring(0,commentIndex);
101     }
102     return new StringTokenizer(line);
103 }
104
105 /**
106  * The prompt for the CLI.
107  *
108  * @return the prompt string.
109  */
110
111 public String getPrompt()
112 {

```

```
113     }
114     return system.getHostName() + "> ";
115 }
116 /**
117  * The User associated with this shell.
118  *
119  * @return the user.
120  */
121
122 public User getUser()
123 {
124     return user;
125 }
126
127 /**
128  * The current working directory for this shell.
129  *
130  * @return the current working directory.
131  */
132
133 public Directory getDot()
134 {
135     return dot;
136 }
137
138 /**
139  * Set the current working directory for this shell.
140  *
141  * @param dot the new working directory.
142  */
143
144 public void setDot(Directory dot)
145 {
146     this.dot = dot;
147 }
148
149 /**
150  * The console associated with this shell.
151  *
152  * @return the console.
153  */
154
155 public Terminal getConsole()
156 {
157     return console;
158 }
159
160 /**
161  * The Juno object associated with this Shell.
162  *
163  * @return the Juno instance that created this Shell.
164  */
165
166 public Juno getSystem()
167 {
168     return system;
169 }
```

```
169     }
170 }
```