```
 1  // joi/6/juno/Shell.java
 2  //
 3  //
 4  // Copyright 2003, Ethan Bolker and Bill Campbell
 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 6
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(); // start the command line interpreter
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) {
64              return true;             // skip blank line
65          }
66          String commandName = st.nextToken();
67          if (commandName.equals( "logout" )) {
68              return false;            // user is done
69          }
70          ShellCommand commandObject =
71              system.getCommandTable().lookup( commandName );
72          if (commandObject == null ) {
73              console.errPrintln( "Unknown command: " + commandName );
74          }
75          else {
76              commandObject.doIt( st, this );
77          }
78          return true;
79      }
80
81      // Strip characters from '#' to end of line, create and
82      // return a StringTokenizer for what's left.
83
84      private StringTokenizer stripComments( String line )
85      {
86          int commentIndex = line.indexOf('#');
87          if (commentIndex >= 0) {
88              line = line.substring(0,commentIndex);
89          }
90          return new StringTokenizer(line);
91      }
92
93      /**
94       * The prompt for the CLI.
95       *
96       * @return the prompt string.
97       */
98
99      public String getPrompt()
100     {
101         return system.getHostName() + "> ";
102     }
103
104     /**
105      * The User associated with this Shell.
106      *
107      * @return the user.
108      */
109
110     public User getUser()
111     {
112         return user;
```

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