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

import java.util.*;

/**
 * Models a shell (command interpreter)
 *
 * The Shell knows the (Juno) system it’s working in,
 * the User who started it,
 * and the console to which to send output.
 *
 * It keeps track of the the current working directory (.) .
 *
 * @version 7
 */

public class Shell
{
    private Juno system;       // the operating system object
    private User user;         // the user logged in
    private Terminal console;  // the console for this shell
    private Directory dot;     // the current working directory

    /**
     * Construct a login shell for the given user and console.
     *
     * @param system a reference to the Juno system.
     * @param user the User logging in.
     * @param console a Terminal for input and output.
     */

    public Shell( Juno system, User user, Terminal console )
    {
        this.system  = system;
        this.user    = user;
        this.console = console;
        dot          = user.getHome(); // default current directory
        CLIShell();
    }

    // Run the command line interpreter

    private void CLIShell()
    {
        boolean moreWork = true;
        while(moreWork) {
            moreWork = interpret( console.readLine( getPrompt() ) );
        }
        console.println("goodbye");
    }

    // Interpret a String of the form
    //     shellcommand command-arguments
```

```java
    //
    // return true, unless shell command is logout.

    private boolean interpret( String inputLine )
    {
        StringTokenizer st = stripComments(inputLine);
        if (st.countTokens() == 0) {           // skip blank line
            return true;
        }
        String commandName = st.nextToken();
        ShellCommand commandObject =
            system.getCommandTable().lookup( commandName );
        if (commandObject == null ) {                             // EEE
            console.errPrintln("Unknown command: " + commandName);// EEE
            return true;
        }                                                       // EEE
        try {                                                   // EEE
            commandObject.doIt( st, this );                     // EEE
        }                                                       // EEE
        catch (ExitShellException e) {
            return false;
        }
        catch (BadShellCommandException e) {                     // EEE
            console.errPrintln( "Usage: " + commandName + " " +  // EEE
                                e.getCommand().getArgString() ); // EEE
        }                                                        // EEE
        catch (JunoException e) {                                // EEE
            console.errPrintln( e.getMessage() );                // EEE
        }                                                        // EEE
        catch (Exception e) {                                    // EEE
            console.errPrintln( "you should never get here" );   // EEE
            console.errPrintln( e.toString() );                  // EEE
        }                                                        // EEE
        return true;
    }

    // Strip characters from ’#’ to end of line, create and
    // return a StringTokenizer for what’s left.

    private StringTokenizer stripComments( String line )
    {
        int commentIndex = line.indexOf(’#’);
        if (commentIndex >= 0) {
            line = line.substring(0,commentIndex);
        }
        return new StringTokenizer(line);
    }

    /**
     * The prompt for the CLI.
     *
     * @return the prompt string.
     */

    public String getPrompt()
    {
        return system.getHostName() + "> ";
    }

    /**
     * The User associated with this shell.
     *
     * @return the user.
     */

    public User getUser()
    {
        return user;
    }

    /**
     * The current working directory for this shell.
     *
     * @return the current working directory.
     */

    public Directory getDot()
    {
        return dot;
    }

    /**
     * Set the current working directory for this Shell.
     *
     * @param dot the new working directory.
     */

    public void setDot(Directory dot)
    {
        this.dot = dot;
    }

    /**
     * The console associated with this Shell.
     *
     * @return the console.
     */

    public Terminal getConsole()
    {
        return console;
    }

    /**
     * The Juno object associated with this Shell.
     *
     * @return the Juno instance that created this Shell.
     */

    public Juno getSystem()
    {
        return system;
```

```
    }
}
```