```
// joi/10/juno/GUIShellConsole.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

/**
 *  The GUI to the Juno system Shell.
 */

public class GUIShellConsole extends JFrame
    implements OutputInterface
{
    private static final int FIELDWIDTH  = 50;
    private static final int FIELDHEIGHT = 10;

    // the components on the window

    private JLabel promptLabel = new JLabel();
    private JTextField commandLine = new JTextField( FIELDWIDTH );
    private JButton doIt       = new JButton( "Do It" );
    private JButton logout     = new JButton( "Logout" );
    private JTextArea stdout   =
                          new JTextArea( FIELDHEIGHT, FIELDWIDTH );
    private JTextArea stderr   =
                          new JTextArea( FIELDHEIGHT/2, FIELDWIDTH );

    private Shell sh;    // for interpreting shell commands
    private WindowCloser closeMe; // for logging out.

    private boolean echoInput;

    /**
     * Construct a GUI console for a shell.
     *
     * @param title the title to  display in the frame.
     * @param sh the shell to interpret commands.
     * @param echoInput is input to be echoed?
     */

    public GUIShellConsole( String title,
                            Shell sh,
                            boolean echoInput )
    {
        this.sh = sh;
        this.echoInput   = echoInput;

        setTitle( title );
        setPrompt( sh.getPrompt() );

        // set up console's look and feel

```



```
        JPanel outerPanel = new JPanel();
        outerPanel.setLayout( new BorderLayout() );

        Box box = Box.createVerticalBox();

        JPanel commandPanel = new JPanel();
        commandPanel.setLayout( new BorderLayout() );
        commandPanel.add( promptLabel, BorderLayout.NORTH );
        commandPanel.add( commandLine, BorderLayout.CENTER );
        box.add( commandPanel );
        box.add( Box.createVerticalStrut( 10 ) );

        Box buttons = Box.createHorizontalBox();
        buttons.add( Box.createGlue() );
        buttons.add( doIt );
        buttons.add( Box.createGlue() );
        buttons.add( logout );
        buttons.add( Box.createGlue() );
        box.add( buttons );
        box.add( Box.createVerticalStrut( 10 ) );

        JPanel stdoutPanel = new JPanel();
        stdoutPanel.setLayout( new BorderLayout() );
        stdoutPanel.add( new JLabel( "Standard output:"),
                         BorderLayout.NORTH );

        stdoutPanel.add( new JScrollPane( stdout ),
                         BorderLayout.CENTER );

        box.add( stdoutPanel );
        box.add( Box.createVerticalStrut( 10 ) );
        stdout.setEditable( false );

        JPanel stderrPanel = new JPanel();
        stderrPanel.setLayout( new BorderLayout() );
        stderrPanel.add( new JLabel( "Error output:" ),
                         BorderLayout.NORTH );
        stderrPanel.add( new JScrollPane( stderr ),
                         BorderLayout.CENTER );
        box.add( stderrPanel );
        box.add( Box.createVerticalStrut( 10 ) );
        stderr.setEditable( false );

        outerPanel.add( box, BorderLayout.CENTER );
        this.getContentPane().add( outerPanel, BorderLayout.CENTER );

        // Install menus and tool bar.

        JMenuBar menuBar  = new JMenuBar();
        JMenu commandMenu = new JMenu( "Command" );
        JMenu helpMenu    = new JMenu( "Help" );

        JToolBar toolBar  = new JToolBar();

        // Create menu items and tool buttons for each shell command

```

```
113          ShellCommandTable table = sh.getSystem().getCommandTable();
114          String [] commandNames = table.getCommandNames();
115          for ( int i = 0; i < commandNames.length; i++ ) {
116
117              String commandName = commandNames[i];
118              ShellCommand command =
119                  table.lookup( commandName );
120
121              CommandMenuAction commandAction =
122                  new CommandMenuAction( commandName,
123                                         command.getArgString() );
124
125              HelpMenuAction helpAction =
126                  new HelpMenuAction( commandName,
127                                      command.getArgString(),
128                                      command.getHelpString() );
129
130              JMenuItem item1 = commandMenu.add( commandAction );
131              JMenuItem item2 = helpMenu.add( helpAction );
132              JButton button  = toolBar.add( commandAction );
133              button.setToolTipText( command.getHelpString() );
134          }
135
136          this.setJMenuBar( menuBar );
137          this.getContentPane().add( toolBar,
138                                     BorderLayout.NORTH );
139          menuBar.add( commandMenu );
140          menuBar.add( helpMenu );
141
142          pack();
143          show();
144
145          // add Listener to the Do It button
146
147          doIt.addActionListener( new Interpreter() );
148
149          // add listener to the Logout button and window closer
150
151          closeMe = new WindowCloser( this );
152          logout.addActionListener( closeMe );
153          this.addWindowListener( closeMe );
154      }
155
156      // Set the GUI prompt
157
158      private void setPrompt(String prompt)
159      {
160          this.promptLabel.setText(prompt);
161      }
162
163      // Implementing the OutputInterface.
164      // Everything goes to the single message area.
165
166      public void println( String str )
167      {
168          stdout.append(str + "\n");
```



```
169      }
170
171      public void errPrintln( String str )
172      {
173          stderr.append(str + "\n");
174      }
175
176      public boolean isGUI()
177      {
178          return true;
179      }
180
181      public boolean isRemote()
182      {
183          return false;
184      }
185
186      public boolean isEchoInput()
187      {
188          return echoInput;
189      }
190
191      // An inner class for the semantics when the user submits
192      // a ShellCommand for execution.
193
194      private class Interpreter
195          implements ActionListener
196      {
197          public void actionPerformed( ActionEvent e )
198          {
199              String str = commandLine.getText();
200              stdout.append( sh.getPrompt() + str + '\n');
201              if (sh.interpret( str )) {
202                  setPrompt( sh.getPrompt() );
203              }
204              else {
205                  closeMe.actionPerformed(null);
206              }
207          }
208      }
209
210      private class CommandMenuAction extends AbstractAction
211      {
212          private String argString;
213          private String helpString;
214
215          public CommandMenuAction( String text, String argString )
216          {
217              super( text );
218              this. argString = argString;
219          }
220
221          public void actionPerformed( ActionEvent e )
222          {
223              commandLine.setText( getValue( Action.NAME ) +
224                                   " " + argString );
```

```
225         }
226     }
227
228     private class HelpMenuAction extends AbstractAction
229     {
230         private String argString;
231         private String helpString;
232
233         public HelpMenuAction( String text, String argString,
234                                String helpString )
235         {
236             super( text );
237             this. argString = argString;
238             this.helpString = helpString;
239         }
240
241         public void actionPerformed( ActionEvent e )
242         {
243             stdout.append( getValue( Action.NAME ) + ": " +
244                            helpString );
245         }
246     }
247
248     //  A WindowCloser instance handles close events generated
249     //  by the underlying window system with its windowClosing
250     //  method, and close events from buttons or other user
251     //  components with its actionPerformed method.
252     //
253     //  The action is to logout and dispose of this window.
254
255     private static class WindowCloser extends WindowAdapter
256         implements ActionListener
257     {
258         Frame myFrame;
259
260         public WindowCloser( Frame frame ) {
261             myFrame = frame;
262         }
263
264         public void windowClosing (WindowEvent e)
265         {
266             this.actionPerformed( null );
267         }
268
269         public void actionPerformed(ActionEvent e)
270         {
271             myFrame.dispose();
272         }
273     }
274 }
```