

```

1 // fo1/10/juno/GUILoginConsole.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5
6 import javax.swing.*;
7 import javax.swing.event.*;
8 import java.awt.*;
9 import java.awt.event.*;
10
11 /**
12  * The graphical user interface to Juno.
13  */
14
15 public class GUILoginConsole extends JFrame
16 implements OutputInterface
17 {
18     private static final int FIELDWIDTH = 30;
19     private static final int FIELDHEIGHT = 5;
20
21     private final Juno junoSystem;
22     private WindowCloser closeMe; // to shut down Juno
23
24     private String title; // title for the windows
25
26     // The interpreter interprets one-line commands.
27     private InterpreterInterface interpreter;
28     private boolean echoInput;
29
30     // All output goes to messages.
31     private JTextArea messages;
32
33     /**
34      * Construct a GUI console for Juno.
35      *
36      * @param title the title for this window.
37      * @param junoSystem the Juno system for which this is a GUI
38      * @param interpreter the object to which to send user input.
39      * @param echoInput true when input echoes to this console.
40      */
41
42     public GUILoginConsole( String title, Juno junoSystem,
43                           InterpreterInterface interpreter,
44                           boolean echoInput)
45     {
46         super( title );
47         this.title = title;
48         this.junoSystem = junoSystem;
49         this.interpreter = interpreter;
50         this.echoInput = echoInput;
51         this.closeMe = new WindowCloser( junoSystem );
52
53         // Set up the look and feel;
54         // Everything is placed on a panel (using BorderLayout)
55         JPanel panel = new JPanel();

```

```

57     panel.setLayout( new BorderLayout() );
58
59     // First a tabbed pane, with two tabs:
60     // one for login, one for registration
61
62     JTabbedPane tabs = new JTabbedPane();
63     tabs.addTab( "Login", null,
64               new LoginPane( interpreter, echoInput, closeMe ) );
65     tabs.addTab( "Register", null,
66               new RegisterPane( interpreter, echoInput ) );
67     tabs.setSelectedIndex( 0 ); // Login selected by default
68     panel.add( tabs, BorderLayout.NORTH );
69
70     // and the output messages area.
71     panel.add( new JLabel( "Messages:" ), BorderLayout.CENTER );
72     messages = new JTextArea( FIELDHEIGHT, FIELDWIDTH );
73     panel.add( messages, BorderLayout.SOUTH );
74
75     // Add the panel to this JFrame
76     this.getContentPane().add( panel );
77
78     // Closing this window
79     this.setDefaultCloseOperation( JFrame.DO_NOTHING_ON_CLOSE );
80     this.addWindowListener( closeMe );
81
82     // Size and display this JFrame
83     pack();
84     show();
85
86     // Implementing the OutputInterface. Everything goes to the
87     // single message area.
88
89     /**
90      * Write a String followed by a newline
91      * to message area.
92      *
93      * @param str - the string to write
94      */
95
96     public void println( String str )
97     {
98         messages.append( str + "\n" );
99     }
100
101     /**
102      * Write a String followed by a newline
103      * to message area.
104      *
105      * @param str - the String to write
106      */
107
108     public void errPrintln( String str )
109     {
110         println( str );
111     }
112

```

```

113
114 /**
115  * Query what kind of console this is.
116  *
117  * @return true if and only if echoing input.
118  */
119
120 public boolean isEchoInput()
121 {
122     return echoInput;
123 }
124
125 /**
126  * Query what kind of console this is.
127  *
128  * @return true if and only if GUI
129  */
130
131 public boolean isGUI()
132 {
133     return true;
134 }
135
136 /**
137  * Query what kind of console this is.
138  *
139  * @return true if and only if remote
140  */
141
142 public boolean isRemote()
143 {
144     return false;
145 }
146
147 // The Login pane is specified in a private inner class,
148 // visible only here.
149
150 private class LoginPane extends JPanel
151 {
152     // The login pane has two text fields and two buttons.
153     private JTextField nameField;
154     private JTextField passwordField;
155
156     private JButton ok;
157     private JButton exit;
158
159     private WindowCloser closeMe; // to shut down Juno
160     // Construct the login pane and set up its listeners.
161
162     public LoginPane( InterpreterInterface interpreter,
163                     boolean echoInput, WindowCloser closeMe )
164     {
165         super();
166         this.closeMe = closeMe;
167     }
168     // Set up the look and feel.

```

```

169
170 // Everything will go into a vertical Box, a container
171 // whose contents are laid out using BoxLayout
172 Box box = Box.createVerticalBox();
173
174 // First a panel, containing the two text fields
175
176 JPanel p = new JPanel();
177 p.setLayout( new GridLayout( 4, 1 ) );
178
179 p.add( new JLabel( "Login:" ) );
180 nameField = new JTextField( FIELDWIDTH );
181 p.add( nameField );
182
183 p.add( new JLabel( "Password:" ) );
184 passwordField = new JPasswordField( FIELDWIDTH );
185 p.add( passwordField );
186
187 box.add( p );
188 box.add( Box.createVerticalStrut( 15 ) );
189
190 // Then a horizontal Box containing the two buttons
191 Box row = Box.createHorizontalBox();
192 row.add( Box.createGlue() );
193
194 ok = new JButton( "OK" );
195 row.add( ok );
196 row.add( Box.createGlue() );
197
198 exit = new JButton( "Exit" );
199 row.add( exit );
200 row.add( Box.createGlue() );
201 box.add( row );
202 box.add( Box.createVerticalStrut( 15 ) );
203
204 this.setLayout( new BorderLayout() );
205 this.add( box, BorderLayout.CENTER );
206
207 // Set up the listeners (the semantics)
208
209 ok.addActionListener( new LoginProcessor() );
210 exit.addActionListener( closeMe ); // shuts down Juno
211
212 }
213
214 // An inner class for the semantics
215 // when the user clicks OK.
216
217 private class LoginProcessor implements ActionListener
218 {
219     public void actionPerformed( ActionEvent e )
220     {
221         String str = nameField.getText() + " " +
222             passwordField.getText();
223         passwordField.setText( "" );
224         messages.setText( str + "\n" ); // For debugging

```

```

225         interpreter.interpret( str );
226     }
227 }
228 }
229 // The Register pane is specified in a private inner class,
230 // visible only here.
231
232 private class RegisterPane extends JPanel
233 {
234     // The register pane has four textfields and two buttons.
235     private JTextField chosenName;
236     private JTextField fullName;
237     private JTextField password1;
238     private JTextField password2;
239
240     private JButton register;
241     private JButton clear;
242
243     public RegisterPane( InterpreterInterface interpreter,
244                         boolean echoInput )
245     {
246         super();
247
248         // Define the look and feel
249         // Everything goes into a vertical Box
250         Box box = Box.createVerticalBox();
251
252         // First a panel containing the text fields
253         JPanel p = new JPanel();
254         p.setLayout( new GridLayout( 0 , 1 ) );
255
256         p.add( new JLabel( "Choose login name:" ) );
257         chosenName = new JTextField( FIELDWIDTH );
258         p.add( chosenName );
259
260         p.add( new JLabel( "Give full name:" ) );
261         fullName = new JTextField( FIELDWIDTH );
262         p.add( fullName );
263
264         p.add( new JLabel( "Choose password:" ) );
265         password1 = new JTextField( FIELDWIDTH );
266         p.add( password1 );
267
268         p.add( new JLabel( "Retype password:" ) );
269         password2 = new JTextField( FIELDWIDTH );
270         p.add( password2 );
271
272         box.add( p );
273
274         box.add( Box.createVerticalStrut( 15 ) );
275
276         // Then a horizontal Box containing the buttons
277         Box row = Box.createHorizontalBox();
278         row.add( Box.createGlue() );
279
280

```

```

281         register = new JButton( "Register" );
282         row.add( register );
283         row.add( Box.createGlue() );
284         clear = new JButton( "Clear" );
285         row.add( clear );
286         row.add( Box.createGlue() );
287         box.add( row );
288         box.add( Box.createVerticalStrut( 15 ) );
289
290         this.setLayout( new BorderLayout() );
291         this.add( box, BorderLayout.CENTER );
292
293         // Set up the listeners (the semantics)
294         register.addActionListener( new Registration() );
295         clear.addActionListener( new Cleanser() );
296     }
297
298     // An inner class for the semantics when the user
299     // clicks Register.
300     private class Registration implements ActionListener
301     {
302         public void actionPerformed( ActionEvent e )
303         {
304             if ( password1.getText().trim().equals(
305                 password2.getText().trim() ) ) {
306                 String str = "register " +
307                     chosenName.getText() + " " +
308                     password1.getText() + " " +
309                     fullName.getText();
310                 chosenName.setText("");
311                 fullName.setText("");
312                 messages.setText( str + '\n' ); // for debugging
313                 interpreter.interpret( str );
314             }
315             else {
316                 messages.setText(
317                     "Sorry, passwords don't match.\n" );
318             }
319         }
320     }
321
322     // An inner class for the semantics when the user
323     // clicks Clear.
324     private class Cleanser implements ActionListener {
325         public void actionPerformed( ActionEvent e ) {
326             chosenName.setText("");
327             fullName.setText("");
328             password1.setText("");
329             password2.setText("");
330         }
331     }
332 }
333 }
334 }
335 }
336 }

```

```
337     }
338
339     // A WindowCloser instance handles close events generated
340     // by the underlying window system with its windowClosing
341     // method, and close events from buttons or other user
342     // components with its actionPerformed method.
343     //
344     // The action is to shut down Juno.
345
346     private static class WindowCloser extends WindowAdapter
347     implements ActionListener
348     {
349         Juno system;
350
351         public WindowCloser( Juno system )
352         {
353             this.system = system;
354         }
355
356         public void windowClosing (WindowEvent e)
357         {
358             this.actionPerformed( null );
359         }
360
361         public void actionPerformed(ActionEvent e)
362         {
363             if (system != null) {
364                 system.shutdown();
365             }
366             System.exit(0);
367         }
368     }
369
370     /**
371     * main() in GUILoginConsole class for
372     * unit testing during development.
373     */
374
375     public static void main( String[] args )
376     {
377         new GUILoginConsole( "GUItest", null, null, true ).show();
378     }
379 }
380
```