```
 1  // joi/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          // Everthing is placed on a panel (using BorderLayout)
55
56          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
87      // Implementing the OutputInterface. Everything goes to the
88      // single message area.
89
90      /**
91       * Write a String followed by a newline
92       * to message area.
93       *
94       * @param str - the string to write
95       */
96
97      public void println(String str )
98      {
99          messages.append( str + "\n" );
100     }
101
102     /**
103      * Write a String followed by a newline
104      * to message area.
105      *
106      * @param str - the String to write
107      */
108
109     public void errPrintln(String str )
110     {
111         println( str );
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             // Everything will go into a vertical Box, a container
170             // whose contents are laid out using BoxLayout
171
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             box.add( p );
187             box.add( Box.createVerticalStrut( 15 ) );
188
189             // Then a horizontal Box containing the two buttons
190
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         // An inner inner class for the semantics
214         // when the user clicks OK.
215
216         private class LoginProcessor implements ActionListener
217         {
218             public void actionPerformed(ActionEvent e)
219             {
220                 String str = nameField.getText() +  " " +
221                     passwordField.getText();
222                 nameField.setText("");
223                 passwordField.setText("");
224                 messages.setText(str+'\n');  // for debugging
```

```
                interpreter.interpret( str );
            }
        }
    }

    // The Register pane is specified in a private inner class,
    // visible only here.

    private class  RegisterPane extends JPanel
    {
        // The register pane has four textfields and two buttons.
        private JTextField chosenName;
        private JTextField fullName;
        private JTextField password1;
        private JTextField password2;

        private JButton register;
        private JButton clear;

        public RegisterPane( InterpreterInterface interpreter,
                             boolean echoInput)
        {
            super();

            // Define the look and feel
            // Everything goes into a vertical Box
            Box box = Box.createVerticalBox();

            // First a panel containing the text fields

            JPanel p  = new JPanel();
            p.setLayout( new GridLayout( 0 , 1 ) );

            p.add( new JLabel( "Choose login name:" ) );
            chosenName = new JTextField( FIELDWIDTH );
            p.add( chosenName );

            p.add( new JLabel( "Give full name:" ) );
            fullName = new JTextField( FIELDWIDTH );
            p.add( fullName );

            p.add( new JLabel( "Choose password:" ) );
            password1  = new JPasswordField( FIELDWIDTH );
            p.add( password1 );

            p.add( new JLabel( "Retype password:" ) );
            password2  = new JPasswordField( FIELDWIDTH );
            p.add( password2 );

            box.add( p );
            box.add( Box.createVerticalStrut( 15 ) );

            // Then a horizontal Box containing the buttons

            Box row = Box.createHorizontalBox();
            row.add( Box.createGlue() );
```



```
            register = new JButton( "Register" );
            row.add( register );
            row.add( Box.createGlue() );
            clear = new JButton( "Clear" );
            row.add( clear );
            row.add( Box.createGlue() );
            box.add( row );
            box.add( Box.createVerticalStrut( 15 ) );

            this.setLayout( new BorderLayout() );
            this.add( box, BorderLayout.CENTER );

            // Set up the listeners (the semantics)

            register.addActionListener( new Registration() );
            clear.addActionListener( new Cleanser() );
        }

        // An inner inner class for the semantics when the user
        // clicks Register.

        private class Registration implements ActionListener
        {
            public void actionPerformed(ActionEvent e)
            {
                if ( password1.getText().trim().equals(
                     password2.getText().trim() ) ) {
                    String str = "register " +
                        chosenName.getText() +  " " +
                        password1.getText() + " " +
                        fullName.getText() ;
                    chosenName.setText("");
                    fullName.setText("");
                    messages.setText(str+'\n');  // for debugging
                    interpreter.interpret(str);
                }
                else {
                    messages.setText(
                        "Sorry, passwords don't match.\n" );
                }
                password1.setText("");
                password2.setText("");
            }
        }

        // An inner inner class for the semantics when the user
        // clicks Clear.

        private class Cleanser implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                chosenName.setText("");
                fullName.setText("");
                password1.setText("");
                password2.setText("");
            }
        }
```

```
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
```