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

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

/**
 *  The graphical user interface to Juno.
 */

public class GUILoginConsole extends JFrame
    implements OutputInterface
{
    private static final int FIELDWIDTH = 30;
    private static final int FIELDHEIGHT = 5;

    private final Juno junoSystem;
    private WindowCloser closeMe; // to shut down Juno

    private String title;  // title for the windows

    // The interpreter interprets one-line commands.
    private InterpreterInterface interpreter;
    private boolean echoInput;

    // All output goes to messages.
    private JTextArea messages;

    /**
     *  Construct a GUI console for Juno.
     *
     *  @param title the title for this window.
     *  @param junoSystem the Juno system for which this is a GUI
     *  @param interpreter the object to which to send user input.
     *  @param echoInput true when input echoes to this console.
     */

    public GUILoginConsole( String title, Juno junoSystem,
                            InterpreterInterface interpreter,
                            boolean echoInput)
    {
        super( title );
        this.title       = title;
        this.junoSystem  = junoSystem;
        this.interpreter = interpreter;
        this.echoInput   = echoInput;
        this.closeMe = new WindowCloser( junoSystem );

        // Set up the look and feel;
        // Everthing is placed on a panel (using BorderLayout)

        JPanel panel = new JPanel();
```



```
        panel.setLayout( new BorderLayout() );

        // First a tabbed pane, with two tabs:
        //   one for login, one for registration

        JTabbedPane tabs = new JTabbedPane();
        tabs.addTab( "Login" , null,
                     new LoginPane( interpreter, echoInput, closeMe ) );
        tabs.addTab( "Register", null,
                     new RegisterPane( interpreter, echoInput ) );
        tabs.setSelectedIndex( 0 );  // Login selected by default
        panel.add( tabs, BorderLayout.NORTH );

        // and the output messages area.
        panel.add( new JLabel( "Messages:" ), BorderLayout.CENTER );
        messages = new JTextArea( FIELDHEIGHT, FIELDWIDTH );
        panel.add( messages, BorderLayout.SOUTH );

        // Add the panel to this JFrame
        this.getContentPane().add( panel );

        // Closing this window
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        this.addWindowListener( closeMe );

        // Size and display this JFrame
        pack();
        show();
    }

    // Implementing the OutputInterface. Everything goes to the
    // single message area.

    /**
     * Write a String followed by a newline
     * to message area.
     *
     * @param str - the string to write
     */

    public void println(String str )
    {
        messages.append( str + "\n" );
    }

    /**
     * Write a String followed by a newline
     * to message area.
     *
     * @param str - the String to write
     */

    public void errPrintln(String str )
    {
        println( str );
    }
```

```
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
```

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



```
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
295             register.addActionListener( new Registration() );
296             clear.addActionListener( new Cleanser() );
297         }
298
299         // An inner inner class for the semantics when the user
300         // clicks Register.
301
302         private class Registration implements ActionListener
303         {
304             public void actionPerformed(ActionEvent e)
305             {
306                 if ( password1.getText().trim().equals(
307                      password2.getText().trim() ) ) {
308                     String str = "register " +
309                         chosenName.getText() +  " " +
310                         password1.getText() + " " +
311                         fullName.getText() ;
312                     chosenName.setText("");
313                     fullName.setText("");
314                     messages.setText(str+'\n');  // for debugging
315                     interpreter.interpret(str);
316                 }
317                 else {
318                     messages.setText(
319                         "Sorry, passwords don't match.\n" );
320                 }
321                 password1.setText("");
322                 password2.setText("");
323             }
324         }
325
326         // An inner inner class for the semantics when the user
327         // clicks Clear.
328
329         private class Cleanser implements ActionListener {
330             public void actionPerformed(ActionEvent e) {
331                 chosenName.setText("");
332                 fullName.setText("");
333                 password1.setText("");
334                 password2.setText("");
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
```