

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Main.java

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package project1;
7
8  import java.awt.*;
9  import javax.swing.*;
10
11 /**
12  *
13  * @author Brian Cullinan
14  */
15 public class Main {
16
17     /**
18     * @param args the command line arguments
19     */
20     public static void main(String[] args) {
21
22         // this is our main window frame
23         new Application();
24     }
25
26 }
27
28

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Application.java

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package project1;
7
8  import javax.swing.*;
9  import java.awt.*;
10 import javax.swing.event.*;
11 import java.awt.event.*;
12
13 /**
14  *
15  * @author Brian Cullinan
16  */
17 public class Application extends JFrame {
18
19     public Application()
20     {
21         super();
22         setTitle("Media Server");
23         setSize(640, 480);
24         setExtendedState(JFrame.MAXIMIZED_BOTH);
25         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26
27         new Desktop(this);
28     }
29 }
30
31

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Desktop.java

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package project1;
7
8  import javax.swing.*;
9  import java.awt.*;
10 import javax.swing.event.*;
11 import java.awt.event.*;
12

```

```
13 /**
14  *
15  * @author Brian Cullinan
16  */
17 public class Desktop extends JDesktopPane implements ComponentListener, InternalFrameListener
18 {
19     Application application;
20     Taskbar taskbar;
21     StartMenu startmenu;
22
23
24     public Desktop(Application application)
25     {
26         super();
27         //try {
28         //    UIManager.addAuxiliaryLookAndFeel(new project1.ui.NewLookAndFeel());
29         //} catch (Exception ex) {
30
31         //}
32         addComponentListener(this);
33         this.application = application;
34         application.setContentPane(this);
35         application.setVisible(true);
36
37         setTaskbar(new Taskbar(this));
38
39         setStartMenu(new StartMenu(this));
40     }
41
42     public JInternalFrame add(JInternalFrame frame)
43     {
44         frame.addInternalFrameListener(this);
45         frame.setIconifiable(false);
46         super.add(frame);
47         return frame;
48     }
49
50     public Module add(Module frame)
51     {
52         frame.addInternalFrameListener(this);
53         taskbar.add(frame);
54         frame.setIconifiable(false);
55         super.add(frame);
56         return frame;
57     }
58
59     public void setTaskbar(Taskbar taskbar)
60     {
61         this.taskbar = taskbar;
62     }
63
64     public void setStartMenu(StartMenu launcher)
65     {
66         this.startmenu = launcher;
67     }
68
69     public void componentHidden(ComponentEvent e)
70     {
71
72     }
73     public void componentMoved(ComponentEvent e)
74     {
75
76     }
77     public void componentResized(ComponentEvent e)
78     {
79         if(taskbar != null)
80             taskbar.setBounds(0,
81                 getHeight() - taskbar.getHeight() - getInsets().top - getInsets().bottom,
82                 getWidth() - getInsets().left - getInsets().right,
83                 taskbar.getHeight());
84     }
85     public void componentShown(ComponentEvent e)
86     {
87
88     }
89
90     public void internalFrameActivated(InternalFrameEvent e) {
91         try {
92             ((Module)e.getSource()).taskbutton.setSelected(true);
93         } catch (Exception ex) {
```

```

94     }
95     }
96 }
97 public void internalFrameClosed(InternalFrameEvent e) {
98     try {
99         taskbar.taskbar.remove(((Module)e.getSource()).taskbutton);
100     } catch (Exception ex) {
101     }
102 }
103 }
104 public void internalFrameClosing(InternalFrameEvent e) {
105 }
106 }
107 public void internalFrameDeactivated(InternalFrameEvent e) {
108     if(e.getSource() == startmenu)
109     {
110         startmenu.setVisible(false);
111     }
112     try {
113         ((Module)e.getSource()).taskbutton.setSelected(false);
114     } catch (Exception ex) {
115     }
116 }
117 }
118 public void internalFrameDeiconified(InternalFrameEvent e) {
119 }
120 }
121 public void internalFrameIconified(InternalFrameEvent e) {
122 }
123 }
124 public void internalFrameOpened(InternalFrameEvent e) {
125 }
126 }
127 }
128 }
129 }
130 }

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Taskbar.java

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package project1;
7
8  import javax.swing.*;
9  import java.awt.*;
10 import javax.swing.event.*;
11 import java.awt.event.*;
12 import project1.ui.*;
13
14 /**
15  *
16  * @author Brian Cullinan
17  */
18 public class Taskbar extends JInternalFrame implements ActionListener, MouseListener {
19
20     //private static final String uiClassID = "TaskbarUI";
21
22     Desktop desktop;
23
24     JPanel taskbar;
25     // the start button on the taskbar
26     JButton start;
27
28     JPopupMenu popup;
29
30     GridLayout layout;
31
32     public Taskbar(Desktop desktop)
33     {
34         super();
35         this.desktop = desktop;
36         ((javax.swing.plaf.basic.BasicInternalFrameUI)getUI()).setNorthPane(null);
37         setBounds(0,
38             desktop.getHeight() - 50 - desktop.getInsets().top - desktop.getInsets().bottom,
39             desktop.getWidth() - desktop.getInsets().left - desktop.getInsets().right,
40             50);
41         setLayout(new BorderLayout());

```

```
42     setLayer(9999);
43
44     start = new JButton("Start");
45     start.addActionListener(this);
46     add(start, BorderLayout.WEST);
47
48     taskbar = new JPanel();
49     layout = new GridLayout(1, 4);
50     taskbar.setLayout(layout);
51     add(taskbar, BorderLayout.CENTER);
52
53     popup = new JPopupMenu();
54     JMenuItem close = new JMenuItem("Close");
55     close.addActionListener(this);
56     popup.add(close);
57
58     desktop.add(this);
59     setVisible(true);
60 }
61
62 public TaskButton add(Module frame) {
63
64     TaskButton taskbutton = new TaskButton(frame);
65     if(taskbar.getComponents().length == layout.getColumns())
66         layout.setColumns(layout.getColumns()+1);
67     taskbar.add(taskbutton);
68     frame.taskbutton = taskbutton;
69     taskbutton.addActionListener(this);
70     taskbutton.addMouseListener(this);
71
72     return taskbutton;
73 }
74
75 public void actionPerformed(ActionEvent e) {
76     if(e.getSource() == start)
77     {
78         desktop.startmenu.setVisible(true);
79         desktop.startmenu.setLocation(0, getY() - desktop.startmenu.getHeight());
80         desktop.getDesktopManager().activateFrame(desktop.startmenu);
81         desktop.startmenuToFront();
82         try {
83             desktop.startmenu.setSelected(true);
84         }
85         catch (Exception ex) {
86             System.out.println(ex);
87         }
88         desktop.startmenu.requestFocus();
89     }
90     if(e.getSource().getClass() == TaskButton.class)
91     {
92         TaskButton button = (TaskButton)e.getSource();
93         Module frame = (Module)button.frame;
94         if(frame.isIcon() || !frame.isSelected())
95         {
96             desktop.getDesktopManager().activateFrame(frame);
97             frame.toFront();
98             frame.requestFocus();
99             try {
100                 frame.setIcon(false);
101                 frame.setSelected(true);
102             } catch (Exception ex) {
103
104             }
105         }
106         else
107         {
108             button.setSelected(false);
109             desktop.getDesktopManager().activateFrame(desktop.taskbar);
110             frame.toBack();
111             try {
112                 desktop.taskbar.setSelected(true);
113                 frame.setIcon(true);
114                 frame.setSelected(false);
115             } catch (Exception ex) {
116
117             }
118         }
119     }
120     if(e.getSource().getClass() == JMenuItem.class)
121     {
122         JMenuItem item = (JMenuItem)e.getSource();
```

```

123         JPopupMenu menu = (JPopupMenu)item.getParent();
124         TaskButton task = (TaskButton)menu.getInvoker();
125         try {
126             task.frame.setClosed(true);
127         } catch (Exception ex) {
128             System.out.println(ex);
129         }
130     }
131 }
132
133 public void mouseClicked(MouseEvent e) {
134 }
135
136 public void mouseEntered(MouseEvent e) {
137 }
138
139 public void mouseExited(MouseEvent e) {
140 }
141
142 public void mousePressed(MouseEvent e) {
143 }
144
145 public void mouseReleased(MouseEvent e) {
146     if (e.isPopupTrigger()) {
147         popup.show(e.getComponent(),
148                 e.getX(), e.getY());
149     }
150 }
151
152
153 //public String getUIClassID() {
154 //    return uiClassID;
155 //}
156 }
157
158

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\TaskButton.java

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package project1;
7
8  import javax.swing.*;
9  /**
10   *
11   * @author Brian Cullinan
12   */
13  public class TaskButton extends JToggleButton {
14
15      Module frame;
16      public TaskButton(Module frame)
17      {
18          super(frame.getTitle());
19          this.frame = frame;
20      }
21  }
22
23

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\StartMenu.java

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package project1;
7
8  import javax.swing.*;
9  import java.awt.*;
10 import javax.swing.event.*;
11 import java.awt.event.*;
12 import org.xml.sax.*;
13 import java.util.*;
14 import org.xml.sax.helpers.*;
15 import javax.swing.table.*;

```

```
16 import java.awt.event.*;
17
18 /**
19  *
20  * @author Brian Cullinan
21  */
22 public class StartMenu extends JInternalFrame implements ActionListener, ContentHandler {
23
24     Desktop desktop;
25
26     JButton home, downloads;
27     JPanel quick_folders;
28
29     public StartMenu(Desktop desktop)
30     {
31         super();
32         this.desktop = desktop;
33
34         // this is the panel that holds the special locations and main folders
35         ((javax.swing.plaf.basic.BasicInternalFrameUI)getUI()).setNorthPane(null);
36         setSize(100, 480);
37         setLayout(new BorderLayout());
38
39         // this is the panel for the special locations (Downloads, etc.)
40         JPanel locations = new JPanel();
41         locations.setLayout(new GridLayout(0, 1));
42         home = new JButton("Home");
43         home.addActionListener(this);
44         locations.add(home);
45         add(locations, BorderLayout.NORTH);
46
47         // this is the panel for the main directory "quick access"
48         quick_folders = new JPanel();
49         quick_folders.setLayout(new GridLayout(0, 1));
50         quick_folders.setSize(100, 330);
51         add(quick_folders, BorderLayout.CENTER);
52         // get quick folders
53         try {
54             XMLReader xr = XMLReaderFactory.createXMLReader();
55             xr.setContentHandler(this);
56             xr.parse("http://dev.bjccullinan.com/plugins/select.php?template=extjs&dir=/Shared/&start=0&limit");
57         } catch (Exception ex) {
58             System.out.println(ex);
59         }
60
61         desktop.add(this);
62     }
63
64     public void actionPerformed(ActionEvent e) {
65         if(e.getSource() == home)
66         {
67             new Portal(desktop);
68         }
69         else
70         {
71             if(e.getSource().getClass() == JButton.class)
72                 new Portal(desktop, "/Shared/" + ((JButton)e.getSource()).getText() + "/");
73         }
74     }
75     public void setDocumentLocator (Locator locator)
76     { }
77
78     public void startDocument ()
79         throws SAXException
80     { }
81     public void endDocument()
82         throws SAXException
83     { }
84     public void startPrefixMapping (String prefix, String uri)
85         throws SAXException
86     { }
87     public void endPrefixMapping (String prefix)
88         throws SAXException
89     { }
90
91     String current_el;
92     boolean in_file = false;
93     String filename = "";
94     public void startElement (String uri, String localName,
95                             String qName, Attributes atts)
96         throws SAXException
```

```

97     {
98         current_el = qName;
99         if(current_el.equals("file"))
100         {
101             in_file = true;
102         }
103     }
104     public void endElement (String uri, String localName,
105                             String qName)
106         throws SAXException
107     {
108         if(qName.equals("file"))
109         {
110             in_file = false;
111             JButton new_button = new JButton(filename);
112             new_button.addActionListener(this);
113             quick_folders.add(new_button);
114         }
115         current_el = "";
116     }
117     public void characters (char ch[], int start, int length)
118         throws SAXException
119     {
120         if(in_file && current_el.equals("name"))
121         {
122             filename = "";
123             for(int i = start; i < start+length; i++)
124                 filename += ch[i];
125         }
126     }
127     public void ignorableWhitespace (char ch[], int start, int length)
128         throws SAXException
129     {
130     }
131     public void processingInstruction (String target, String data)
132         throws SAXException
133     {
134     }
135     public void skippedEntity (String name)
136         throws SAXException
137     {
138     }
139 }
140

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Module.java

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package project1;
7
8  import javax.swing.*;
9  /**
10   *
11   * @author Brian Cullinan
12   */
13  public class Module extends JFrame {
14      JToggleButton taskbutton;
15  }
16
17

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Portal.java

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package project1;
7
8  import javax.swing.*;
9  import java.awt.*;
10 import java.awt.event.*;
11 import java.util.*;
12 import java.net.*;
13 import java.io.*;

```

```
14 import org.xml.sax.*;
15 import org.xml.sax.helpers.*;
16 import javax.swing.tree.*;
17 import javax.swing.event.*;
18 /**
19  *
20  * @author Brian Cullinan
21  */
22 public class Portal extends Module implements ActionListener, org.xml.sax.ContentHandler, TreeSelectionListe
23
24     Desktop desktop;
25     FolderView view;
26     Address address;
27     Stack<String> back_stack;
28     Stack<String> forward_stack;
29     JProgressBar progress;
30     JPanel search_pane, folder_tasks;
31     JScrollPane folders_pane;
32     JToggleButton search, folders;
33     JTextField contains;
34     JTree tree;
35
36     // toolbar buttons that need controlling from other classes
37     JButton back, forward, up;
38
39     public Portal(Desktop desktop)
40     {
41         this(desktop, "/");
42     }
43
44     public Portal(Desktop desktop, String location)
45     {
46         super();
47         this.desktop = desktop;
48         back_stack = new Stack<String>();
49         forward_stack = new Stack<String>();
50
51         setTitle("Portal");
52         setResizable(true);
53         setClosable(true);
54         setMaximizable(true);
55         setLayout(new BorderLayout());
56         setSize(540, 480);
57
58         JPanel tool_container = new JPanel();
59         tool_container.setLayout(new GridLayout(0, 1));
60         add(tool_container, BorderLayout.NORTH);
61
62         JToolBar tools = new JToolBar();
63         tools.setFloatable(false);
64         tool_container.add(tools, BorderLayout.NORTH);
65         // these are some buttons for controlling the folder
66         back = new JButton("Back");
67         back.addActionListener(this);
68         back.setEnabled(false);
69         tools.add(back);
70         forward = new JButton("Forward");
71         forward.addActionListener(this);
72         forward.setEnabled(false);
73         tools.add(forward);
74         up = new JButton("Up");
75         up.addActionListener(this);
76         up.setEnabled(false);
77         tools.add(up);
78         JButton refresh = new JButton("Refresh");
79         refresh.addActionListener(this);
80         tools.add(refresh);
81         search = new JToggleButton("Search");
82         search.addActionListener(this);
83         tools.add(search);
84         folders = new JToggleButton("Folders");
85         folders.addActionListener(this);
86         tools.add(folders);
87
88         // the main address of the current folder
89         address = new Address(this, location);
90         tool_container.add(address, BorderLayout.NORTH);
91
92         // create a view for the files
93         view = new FolderView(this, address);
94         add(view, BorderLayout.CENTER);
```



```
95     progress = new JProgressBar();
96     progress.setStringPainted(true);
97     add(progress, BorderLayout.SOUTH);
98
99     // this is the folder tasks panel
100    JPanel tasks = new JPanel();
101    tasks.setLayout(new OverlayLayout(tasks));
102    add(tasks, BorderLayout.WEST);
103
104    folder_tasks = new JPanel();
105    folder_tasks.setLayout(new FlowLayout());
106    tasks.add(folder_tasks);
107
108    JButton send_to_downloads = new JButton("Download Selected");
109    send_to_downloads.addActionListener(this);
110    folder_tasks.add(send_to_downloads);
111
112    search_pane = new JPanel();
113    search_pane.setLayout(new FlowLayout());
114    search_pane.setVisible(false);
115    tasks.add(search_pane);
116
117    contains = new JTextField();
118    contains.setPreferredSize(new Dimension(100, 20));
119    search_pane.add(contains);
120
121    JButton search_btn = new JButton("Search");
122    search_btn.addActionListener(this);
123    search_pane.add(search_btn);
124
125    folders_pane = new JScrollPane();
126    folders_pane.setVisible(false);
127    tasks.add(folders_pane);
128
129    DefaultMutableTreeNode root = new DefaultMutableTreeNode("Shared");
130    tree = new JTree(root);
131    tree.addTreeSelectionListener(this);
132    folders_pane.setViewportView(tree);
133
134    desktop.add(this);
135    setVisible(true);
136 }
137
138 public void actionPerformed(ActionEvent e) {
139     if(e.getSource().getClass() == JButton.class)
140     {
141         if(((JButton)e.getSource()).getText().equals("Back"))
142         {
143             address.setAddress(back_stack.peek(), true);
144             forward_stack.push(back_stack.pop());
145             forward.setEnabled(true);
146             if(back_stack.size() == 0)
147                 back.setEnabled(false);
148         }
149         if(((JButton)e.getSource()).getText().equals("Forward"))
150         {
151             address.setAddress(forward_stack.peek(), true);
152             back_stack.push(forward_stack.pop());
153             back.setEnabled(true);
154             if(forward_stack.size() == 0)
155                 forward.setEnabled(false);
156         }
157         if(((JButton)e.getSource()).getText().equals("Up"))
158         {
159             // remove folder from address
160             String folders[] = address.address.getText().split("/");
161             String new_add = "/";
162             if(folders.length >= 3)
163             {
164                 for(int i = 1; i < folders.length - 1; i++)
165                 {
166                     new_add += folders[i] + "/";
167                 }
168             }
169             if(new_add.equals("/"))
170                 up.setEnabled(false);
171             else
172                 up.setEnabled(true);
173             address.setAddress(new_add);
174         }
175     }
```

```
176         if(((JButton)e.getSource()).getText().equals("Refresh"))
177         {
178             view.refresh(address.address.getText());
179         }
180
181         if(((JButton)e.getSource()).getText().equals("Download Selected"))
182         {
183             if(view.files.getSelectedRowCount() > 0)
184             {
185                 JFileChooser chooser = new JFileChooser();
186                 int returnval = chooser.showSaveDialog(this);
187                 if(returnval == JFileChooser.APPROVE_OPTION)
188                 {
189                     String files = "";
190                     for(int row : view.files.getSelectedRows())
191                     {
192                         if(!view.files.getValueAt(row, 6).equals("FOLDER"))
193                         {
194                             files += view.files.getValueAt(row, 0) + ",";
195                         }
196                     }
197                     files = files.substring(0, files.length()-1);
198                     try {
199                         URL url = new URL("http://dev.bjcullinan.com/plugins/zip/" + files + "/file.zip");
200                         progress.setValue(0);
201                         Download down = new Download(url, chooser.getSelectedFile(), progress);
202                         down.start();
203                     } catch (Exception ex) {
204                         System.out.println(ex);
205                     }
206                 }
207             }
208         }
209         if(((JButton)e.getSource()).getText().equals("Search"))
210         {
211             view.refresh(new String[]{"template", "dir", "start", "limit", "includes"}, new String[]{"extjs"
212         }
213     }
214     if(e.getSource().getClass() == JToggleButton.class)
215     {
216         if(((JToggleButton)e.getSource()).getText().equals("Search"))
217         {
218             JToggleButton button = (JToggleButton)e.getSource();
219             if(button.isSelected())
220             {
221                 search_pane.setVisible(true);
222                 folder_tasks.setVisible(false);
223                 folders_pane.setVisible(false);
224                 folders.setSelected(false);
225             }
226             else
227             {
228                 if(!folders.isSelected())
229                 {
230                     search_pane.setVisible(false);
231                     folder_tasks.setVisible(true);
232                     folders_pane.setVisible(false);
233                 }
234             }
235         }
236         if(((JToggleButton)e.getSource()).getText().equals("Folders"))
237         {
238             JToggleButton button = (JToggleButton)e.getSource();
239             if(button.isSelected())
240             {
241                 search_pane.setVisible(false);
242                 folder_tasks.setVisible(false);
243                 folders_pane.setVisible(true);
244                 search.setSelected(false);
245             }
246             else
247             {
248                 if(!folders.isSelected())
249                 {
250                     search_pane.setVisible(false);
251                     folder_tasks.setVisible(true);
252                     folders_pane.setVisible(false);
253                 }
254             }
255         }
256     }
```

```
257     }
258     class Download extends Thread implements Runnable {
259         URL url;
260         File output;
261         JProgressBar progress;
262         public Download(URL url, File output, JProgressBar progress)
263         {
264             this.url = url;
265             this.output = output;
266             this.progress = progress;
267         }
268         @Override
269         public void run()
270         {
271             try {
272                 URLConnection connection = url.openConnection();
273                 int length = Integer.parseInt(connection.getHeaderField("Content-Length"));
274                 progress.setMaximum(length);
275                 InputStream in = connection.getInputStream();
276                 FileOutputStream out = new FileOutputStream(output);
277                 byte[] buf = new byte[4 * 1024]; // 4K buffer
278                 int bytesRead;
279                 while ((bytesRead = in.read(buf)) != -1) {
280                     out.write(buf, 0, bytesRead);
281                     progress.setValue(progress.getValue() + buf.length);
282                     progress.setString("Downloading " + Math.round(((double)progress.getValue() / (double)pr
283                 }
284                 in.close();
285                 out.close();
286             } catch (Exception ex) {
287                 System.out.println(ex);
288             }
289         }
290     }
291
292     public void setDocumentLocator (Locator locator)
293     { }
294
295     public void startDocument ()
296         throws SAXException
297     { }
298     public void endDocument()
299         throws SAXException
300     { }
301     public void startPrefixMapping (String prefix, String uri)
302         throws SAXException
303     { }
304     public void endPrefixMapping (String prefix)
305         throws SAXException
306     { }
307
308     String current_el;
309     boolean in_file = false;
310     String filename = "";
311     public void startElement (String uri, String localName,
312                             String qName, Attributes atts)
313         throws SAXException
314     {
315         current_el = qName;
316         if(current_el.equals("file"))
317         {
318             in_file = true;
319         }
320     }
321     public void endElement (String uri, String localName,
322                             String qName)
323         throws SAXException
324     {
325         if(qName.equals("file"))
326         {
327             in_file = false;
328             DefaultMutableTreeNode node = new DefaultMutableTreeNode(filename);
329             ((DefaultMutableTreeNode)tree.getLastSelectedPathComponent()).add(node);
330         }
331         current_el = "";
332     }
333
334     public void characters (char ch[], int start, int length)
335         throws SAXException
336     {
337         if(in_file && current_el.equals("name"))
```

```

338     {
339         filename = "";
340         for(int i = start; i < start+length; i++)
341             filename += ch[i];
342     }
343 }
344 public void ignorableWhitespace (char ch[], int start, int length)
345     throws SAXException
346 { }
347 public void processingInstruction (String target, String data)
348     throws SAXException
349 { }
350 public void skippedEntity (String name)
351     throws SAXException
352 { }
353
354 public void valueChanged(TreeSelectionEvent e) {
355     DefaultMutableTreeNode node = (DefaultMutableTreeNode)
356         tree.getLastSelectedPathComponent();
357
358     String path = "/";
359     for(int i = 0; i < node.getPath().length; i++)
360     {
361         path += node.getPath()[i].toString() + "/";
362     }
363
364     if (node == null) return;
365
366     try {
367         XMLReader xr = XMLReaderFactory.createXMLReader();
368         xr.setContentHandler(this);
369         xr.parse("http://dev.bjccullinan.com/plugins/select.php?template=extjs&dir=" + path + "&start=0&]
370     } catch (Exception ex) {
371         System.out.println(ex);
372     }
373
374     address.setAddress(path);
375     tree.expandPath(tree.getSelectionPath());
376 }
377 }
378 }
379
380

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Address.java

```

1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package project1;
7
8  import javax.swing.*;
9  import java.awt.*;
10 import java.awt.event.*;
11 import java.util.*;
12 /**
13  *
14  * @author Brian Cullinan
15  */
16 public class Address extends JToolBar {
17
18     JTextField address;
19     FolderView view;
20     Portal portal;
21
22     public Address(Portal portal, String value)
23     {
24         this.portal = portal;
25         address = new JTextField(value);
26
27         setFloatable(false);
28         add(new JLabel("Address"));
29         add(address);
30     }
31
32     public void setAddress(String value)
33     {
34         setAddress(value, false);
35     }
36

```

```

36 public void setAddress(String value, boolean dont_change_stack)
37 {
38     if(!dont_change_stack)
39     {
40         portal.back.setEnabled(true);
41         portal.back_stack.push(address.getText());
42         if(portal.forward_stack.size() > 0 && !value.equals(portal.forward_stack.peek()))
43         {
44             portal.forward.setEnabled(false);
45             portal.forward_stack = new Stack<String>();
46         }
47     }
48     if(!value.equals("/"))
49         portal.up.setEnabled(true);
50     address.setText(value);
51     // get last folder to set as title
52     String title = "Portal - ";
53     String folders[] = value.split("/");
54     if(folders.length > 1)
55         title = "Portal - " + folders[folders.length-1];
56     else
57         title = "Portal - Home";
58     portal.setTitle(title);
59     if(portal.taskbutton != null)
60         portal.taskbutton.setText(portal.getTitle());
61     view.refresh(value);
62 }
63 }
64
65

```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\FolderView.java

```

1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package project1;
7
8  import javax.swing.*;
9  import java.awt.*;
10 import javax.xml.transform.stream.*;
11 import javax.xml.transform.sax.*;
12 import javax.xml.transform.*;
13 import javax.xml.*;
14 import org.xml.sax.*;
15 import java.io.*;
16 import java.util.*;
17 import org.xml.sax.helpers.*;
18 import javax.swing.table.*;
19 import java.awt.event.*;
20 /**
21  *
22  * @author Brian Cullinan
23  */
24 public class FolderView extends JScrollPane implements ContentHandler
25 {
26
27     JTable files;
28     Address address;
29     Portal portal;
30     DefaultTableModel model;
31     boolean column_empty[];
32
33     public FolderView(Portal portal, Address value)
34     {
35         this.portal = portal;
36         this.address = value;
37         address.view = this;
38         // this is the main folder view
39
40         files = new JTable()
41         {
42             public boolean isCellEditable(int row, int col)
43             {
44                 return false;
45             }
46         };
47         model = new DefaultTableModel();
48         files.addMouseListener(new MouseAdapter(){

```

```
49     public void mouseClicked(MouseEvent e){
50         if (e.getClickCount() == 2){
51             int row = files.rowAtPoint(e.getPoint());
52             address.setAddress((String)files.getModel().getValueAt(row, 6));
53         }
54     }
55 }
56
57 try {
58     // get the columns from the server
59     XMLReader xr = XMLReaderFactory.createXMLReader();
60     xr.setContentHandler(this);
61     xr.parse("http://dev.bjcullinan.com/plugins/display.php?template=extjs");
62 } catch (Exception ex) {
63     System.out.println(ex);
64 }
65
66 // get the files and folders
67 address.setAddress(address.address.getText(), true);
68
69 setViewportView(files);
70 }
71
72 public void refresh(String dir)
73 {
74     refresh(new String[]{"template", "dir", "start", "limit"}, new String[]{"extjs", dir, "0", "300"});
75 }
76
77 public void refresh(String[] keys, String[] values)
78 {
79     // reset empty values
80     for(int i = 0; i < column_empty.length; i++)
81         column_empty[i] = true;
82     // remove existing rows
83     rows = new Vector<Vector>();
84     // get extra fields
85     String field_str = "";
86     for(int i = 0; i < keys.length; i++)
87     {
88         field_str += "&" + keys[i] + "=" + values[i];
89     }
90     // get the rows from the server
91     try {
92         XMLReader xr = XMLReaderFactory.createXMLReader();
93         xr.setContentHandler(this);
94         xr.parse("http://dev.bjcullinan.com/plugins/select.php?" + field_str);
95     } catch (Exception ex) {
96         System.out.println(ex);
97     }
98     model.setDataVector(rows, column_arr);
99     hideBlankColumns();
100 }
101
102 public void hideBlankColumns()
103 {
104     Vector<String> new_columns = new Vector<String>();
105     // hide blank columns
106     for(int i = column_empty.length-1; i >= 0; i--)
107     {
108         if(column_empty[i] == false || i < 6)
109         {
110             new_columns.add(0, column_arr.get(i));
111         }
112         else
113         {
114             for(int j = 0; j < rows.size(); j++)
115             {
116                 rows.get(j).remove(i);
117             }
118         }
119     }
120     model.setColumnIdentifiers(new_columns);
121     files.setModel(model);
122 }
123
124 public void setDocumentLocator (Locator locator)
125 { }
126
127 public void startDocument ()
128     throws SAXException
129 { }
```

```
130     public void endDocument()
131     {
132     }
133     public void startPrefixMapping (String prefix, String uri)
134     {
135     }
136     public void endPrefixMapping (String prefix)
137     {
138     }
139
140     String current_el;
141     String columns = "";
142     Vector<String> column_arr;
143     Vector<String> row;
144     Vector<Vector> rows;
145     boolean in_file = false;
146
147     public void startElement (String uri, String localName,
148                             String qName, Attributes atts)
149     {
150     }
151     {
152         current_el = qName;
153         if(current_el.equals("file"))
154         {
155             in_file = true;
156             row = new Vector<String>(column_arr.size(), column_arr.size());
157             for(int i = 0; i < column_arr.size(); i++)
158                 row.add("");
159         }
160     }
161     public void endElement (String uri, String localName,
162                             String qName)
163     {
164     }
165     {
166         if(qName.equals("columns"))
167         {
168             column_arr = new Vector<String>();
169             String column_tmp[] = columns.split(",");
170             for(String column_str : column_tmp)
171             {
172                 if(!column_str.equals(""))
173                 {
174                     column_arr.add(column_str);
175                     model.addColumn(column_str);
176                 }
177             }
178             column_empty = new boolean[column_arr.size()];
179         }
180         if(qName.equals("file"))
181         {
182             in_file = false;
183             rows.add(row);
184         }
185         current_el = "";
186     }
187     public void characters (char ch[], int start, int length)
188     {
189     }
190     {
191         if(current_el.equals("columns"))
192         {
193             for(int i = start; i < start+length; i++)
194             {
195                 columns += ch[i];
196             }
197         }
198         if(in_file && current_el.length() > 5 && current_el.substring(0, 5).equals("info-"))
199         {
200             int index = column_arr.indexOf(current_el.substring(5, current_el.length()));
201             if(index != -1)
202             {
203                 String value = "";
204                 for(int i = start; i < start+length; i++)
205                 {
206                     value += ch[i];
207                 }
208                 value = value.trim();
209                 if(!value.equals(""))
210                     column_empty[index] = false;
211                 row.set(index, row.get(index) + value);
212             }
213         }
214     }
```

```
211     }
212   }
213 }
214 public void ignorableWhitespace (char ch[], int start, int length)
215     throws SAXException
216 {
217 }
217 public void processingInstruction (String target, String data)
218     throws SAXException
219 {
220 }
220 public void skippedEntity (String name)
221     throws SAXException
222 {
223 }
224 }
225
226
```