```java
package project;

import java.awt.CardLayout;

/* This is the class responsible for handling or processing all the
 * user interactions with the User Interface(View Component)
 * This contains references to the View Component,Repository,
 * Authentication service*/

public class UIController implements ActionListener,ListSelectionListener,TableModelListener,FormSaveListener{
    private Map<String,GenericForm> forms;
    private JTable table;
    private TableModel tableModel;
    private Repository repository;              //Repository where all the entities are stored
    private JFrame mainFrame;                   // Main Frame of the User Interface
    private AuthFrame authFrame;                // Authentication frame or window
    private View view;                          // View Component represent the Main Frame
    private AuthService authService;            // Authentication Service handles logon
    private User loggedUser;                    // The Logged User
    private SearchQuery sQuery;                 // Serach text entered by user in the Search form

    public UIController(View view, Repository repository,AuthService authService){
        this.view=view;
        this.authService=authService;
        this.repository=repository;
        forms=view.getForms();
        /* Set the Repository for all the forms so that it knows where to store and retrieve enitities
         * to and from*/
        setFormRepository(repository);
        /* Get the MainFrame from the View Component */
        mainFrame=view.getMainFrame();
        /* Get the Authentication  from the View Component */
        authFrame=(AuthFrame)view.authFrame;
        /* Add the Handlers to the AuthenticationForm to handle the Logon event*/
        addAuthFormListeners();
    }

    /*This will add the UIController as the handler for the Logon form
     * This will delegate processing to the UIController */
    public void addAuthFormListeners(){
        authFrame.addFormSaveListener(this);
    }

    /*This will add the UIController as the handler for any user interactions with the components
     * such as Search Form,Naviagtion Pane,Footer Pane, Table,
     * This will delegate processing to the UIController */
```

```
66      public void addViewListeners(){
67          view.getSearchPane().addFormSaveListner(this);
68          view.getNavPane().addActionListener(this);
69          view.getFooterPane().addActionListner(this);
70          table.getSelectionModel().addListSelectionListener(this);
71          view.getContentPane().addActionListener(this);
72      }
73
74      /*As the user navigate through the tabular list(Table), Any selection of
    record trigger List
75       * selection event and it is being handled by this method, This method update
    the forms with the
76       * correct entities by calling the "bind" method of the form. */
77      @Override
78      public void valueChanged(ListSelectionEvent e) {
79          int row=table.getSelectedRow();
80          if (row!=-1){
81              int id=(Integer) table.getModel().getValueAt(row, 0);
82              Participant p=repository.getParticipant(id);
83              Relationships r=p.getRelationship();
84              Interests i=p.getInterest();
85              forms.get("Participant").bind(p);
86              forms.get("Relationship").bind(r);
87              forms.get("Interests").bind(i);
88              Questionaire q=new Questionaire(p);
89              forms.get("Questionaire").bind(q);
90          }
91      }
92
93
94      /* This is the method handle most of the interactions of the User with the
    User Interface
95       * Buttons in the Navigation Pane, Buttons in the Footer Pane and Buttons in
    the Tool Pane are all
96       * processed by this method */
97      @Override
98      public void actionPerformed(ActionEvent e) {
99          /* Command contains the information of the component which are being
    interacted */
100         String command=e.getActionCommand();
101         /* Get the Content Pane of the View component*/
102         ContentPane contentPane=view.getContentPane();
103         /* Get the Layout of the content Pane*/
104         CardLayout cLayout=(CardLayout)contentPane.getLayout();
105         if (command.contains("Show")){
106             /* This updates the Content Pane with the correct view of the
    participants
107              * which is being triggered through the buttons in the Navigation
    pane.
108              * Parsing command will select the coorect view */
109             displayTable( contentPane,command);
110         }
111         else if (command.equals("Home")){
112             /* This will display the Home Page in the Content Pane*/
113             cLayout.show(contentPane, "home");
```

```
114         }
115         else if (command.equals("Next") || command.equals("Previous")){
116             /* This will process the navigation of the tabular list using the
    "Next" and "Previous" buttons
117              * in the Footer Pane*/
118             int row=table.getSelectedRow();
119             row=(row==-1)?0:row;
120             int rowCount=table.getRowCount();
121             if (command.equals("Next")){
122                 row=(row==rowCount-1)?row:++row;
123             }
124             else{
125                 row=(row==0)?row:--row;
126             }
127             table.setRowSelectionInterval(row, row);
128         }
129         else if (command.equals("Refresh")) {
130             /*This will be processed when refresh button is clicked on the Footer
    Pane for updating the forms
131              * for any changes in the external file "config.txt" */
132             view.refreshForms(loggedUser.getAccessLevel());
133         }
134         if (command.equals("Logoff")) {
135             /* This will close the application completely */
136             mainFrame.dispose();
137         }
138         else
139         if (command.equals("Add New Participant")){
140             /* This will get processed when user click the "Add Participant"
    Button in Navigation Pane*/
141             ((MyModel) tableModel).setTableData(repository.getParticipants());
142             TableModelEvent event=new TableModelEvent(tableModel, -1);
143             ((MyModel) tableModel).fireTableChanged(event);
144             cLayout.show(view.getContentPane(), "FormTable");
145             /*Select the Questionnaire form in the Tabbed Pane to dispaly to the
    user */
146             view.getContentPane().getTabbedPane().setSelectedIndex(3);
147             /* As there are no data for a new Paticipant, bind the forms with the
    null object so that
148              * it display a blank form*/
149             for (String formName : forms.keySet()){
150                 forms.get(formName).bind(null);
151             }
152         }
153         if (command.equals("Remove Participant")){
154             /* This will get processed when user click the "RemoveParticipant"
    Button in Navigation Pane*/
155             ((MyModel) tableModel).setTableData(repository.getParticipants());
156             /* Get the selected row for removal*/
157             int selectedRow=table.getSelectedRow();
158             /* Find the id of the record by querying the Table model , id is
    displyed at the 0th column*/
159             int id=(Integer) table.getModel().getValueAt(selectedRow, 0);
160             /* Retrieve the correct Participant for removal*/
161             Participant participant=repository.getParticipant(id);
```

```
162             TableModel model=(MyModel)table.getModel();
163             /*Remove it from the Model */
164             ((MyModel) model).removeRecord(participant,selectedRow);
165             /* Remove it from the Repository */
166             repository.remove(participant);
167             for (String formName : forms.keySet()){
168                 forms.get(formName).bind(null);
169             }
170 
171         }
172 
173         if (command.equals("FormSave")){
174             /* This will be processed when user click the save button on the Tool
    bar of the content pane*/
175             JTabbedPane tabbedPane =view.getContentPane().getTabbedPane();
176             String
    selectedForm=tabbedPane.getTitleAt(tabbedPane.getSelectedIndex());
177             if (selectedForm.equals("Questionaire")){
178                 GenericForm form=forms.get("Questionaire");
179                 /*As FormConfiguration is registered as a handler for the
    FormSaveEvent of the GenericForm
180                  *Firing a "FormSaveEvent will fire the "FormSaveOccurred" Method
    in the FormConfiguration
181                  *class which will eventually process the form save  */
182                 form.fireFormSaveEvent(new FormSaveEvent(form, ""));
183                 /* Get the Saved data from the FormConfigurator object of the
    Questionnaire form */
184                 Questionaire q=
    (Questionaire)form.getFormConfigurator().getFormData();
185                 /* Use the Questionnaire object to get the other three objects and
    bind with the
186                  * respective forms*/
187                 forms.get("Participant").bind(q.getParticipant());
188                 forms.get("Relationship").bind(q.getRelatioship());
189                 forms.get("Interests").bind(q.getInterest());
190             }
191             if (table !=null) {
192                 /*Update the tabular list to reflect any changes which are made to
    the Participant object */
193                 int row=table.getSelectedRow();
194                 for (int i=0 ;i<8;i++){
195                     ((AbstractTableModel)table.getModel()).fireTableCellUpdated(ro
    w,i);
196                 }
197             }
198         }
199         JPanel contentPanel=contentPane.getContentPanel();
200         if (command.equals("Form Expand")){
201             /* This will be processed when Form Region to be expanded to take the
    full length of the
202              * content Pane*/
203             contentPanel.removeAll();
204             contentPanel.add(contentPane.getToolPanel());
205             contentPanel.add(contentPane.getTabbedPane());
206             contentPanel.revalidate();
```

```
207         }
208         if (command.equals("Table Expand")){
209             /* This will be processed when Tabular List Region to be expanded to take the full length of the
210              * content Pane*/
211             contentPanel.removeAll();
212             contentPanel.add(contentPane.getToolPanel());
213             contentPanel.add(new JScrollPane(contentPane.getTable()));
214             contentPanel.revalidate();
215 
216         }
217         if (command.equals("Reset")){
218             /* This will reset the content Pane to the default to display both Form window
219              * and Tabular List*/
220             contentPanel.removeAll();
221             contentPanel.add(contentPane.getToolPanel());
222             contentPanel.add(contentPane.getTabbedPane());
223             contentPanel.add(new JScrollPane(contentPane.getTable()));
224             contentPanel.revalidate();
225         }
226     }
227 
228     /*This will be processed when user click "Search button" or "Save button" or "Logon Button" on the
229      * forms */
230     @Override
231     public void FormSaveOccured(FormSaveEvent event) {
232         GenericForm form=(GenericForm)event.getSource();
233 
234         /* This handles the Authenticaton Process */
235         if (form.getFormConfigurator().getFormName().equals("Logon")){
236             List<JComponent> messageLabels=form.getMessageList();
237             JLabel errorMessage=(JLabel)messageLabels.get(1);
238             /* Get the User form the Logon form */
239             loggedUser=(User)form.getFormConfigurator().getFormData();
240             /* Get the ACl of the user by calling the Authenicate method in the Auth Service */
241             Optional<String> acl=authService.Authenticate(loggedUser);
242             /* If the User is get Authenticated */
243             if (acl.isPresent()){
244                 String access=acl.get();
245                 loggedUser.setAccessLevel(access);
246                 /*Set the Authentication window invisible */
247                 authFrame.setVisible(false);
248                 /* Render the Main Frame by using the user's acl */
249                 view.renderMainFrame(loggedUser.getAccessLevel());
250                 /* Render the Table */
251                 table=view.getContentPane().getTable();
252                 tableModel=new MyModel();
253                 /* For Author/Guest do not display any data axcept their data*/
254                 if (access.equals("Author")){
255                     repository.clear();
256                 }
257                 /* For others  display all the data*/
```

```
258                 ((MyModel)
    tableModel).setTableHeaders(repository.getParticipantsAttributes());
259                 ((MyModel) tableModel).setTableData(repository.getParticipants());
260                 table.setModel(tableModel);
261                 addViewListeners();
262                 /* Set the Main User interface to be visible*/
263                 mainFrame.setVisible(true);
264                 mainFrame.setEnabled(true);
265                 mainFrame.pack();
266             }
267             else {
268                 /*If authentication fails display the error message in the
    Authentication window */
269                 errorMessage.setVisible(true);
270             }
271         }
272         else if (form.getFormConfigurator().getFormName().equals("Search")){
273             /*This will be processed for Serach form"*/
274             ContentPane contentPane=view.getContentPane();
275             /* Get the search text from the form*/
276             sQuery=(SearchQuery)form.getFormConfigurator().getFormData();
277             /* Update the content pane with the serach results*/
278             displayTable( contentPane,sQuery.getQuery());
279 
280         }
281     }
282 
283     /*This will be processed whenever content pane need to be updated by user
    interactions through buttons */
284     private void displayTable(ContentPane contentPane,String command){
285         List<Participant>
    participantList=RelatioshipRatingService.getNParticipants(repository,parseString(c
    ommand), parseNumber(command));
286         CardLayout cLayout=(CardLayout)contentPane.getLayout();
287         ((MyModel) tableModel).setTableData(participantList);
288         TableModelEvent event=new TableModelEvent(tableModel, -1);
289         ((MyModel) tableModel).fireTableChanged(event);
290         cLayout.show(contentPane, "FormTable");
291         if (tableModel.getRowCount()>0){
292             table.setRowSelectionInterval(0, 0);
293         }
294 
295     }
296 
297     /* This will set the correct Repository for all the forms*/
298     private void setFormRepository(Repository repository){
299         for (String key : forms.keySet()){
300             forms.get(key).getFormConfigurator().setRepository(repository);
301         }
302     }
303 
304     /* The following two utility methods are for parsing Strings and numbers*/
305     private int parseNumber(String s){
306         String str=s.replaceAll("[\\D]", "");
307         if (str.matches("[\\d]+[A-Za-z]?")){
```

```
308             return Integer.parseInt(str);
309         }
310         else return 0;
311     }
312 
313     private String parseString(String s){
314         s=s.toLowerCase();
315         if (s.contains("top")){
316             return "top";
317         }
318         else if (s.contains("bottom")){
319             return "bottom";
320         }
321         return "";
322     }
323 
324     @Override
325     public void tableChanged(TableModelEvent e) {
326     }
327 
328 }
329 
```