

Interface.java

```

1 import java.awt.Container;
2 import java.awt.GridBagConstraints;
3 import java.awt.GridBagLayout;
4 import java.awt.Insets;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
7 import java.util.ArrayList;
8 import javax.swing.JButton;
9 import javax.swing.JFrame;
10 import javax.swing.JLabel;
11 import javax.swing.JOptionPane;
12 import javax.swing.JTextField;
13
14 public class Interface implements ActionListener{
15
16     ArrayList personsList;
17     PersonDAO pDAO;
18     JFrame appFrame;
19     JLabel jlbName, jlbAddress, jlbPhone, jlbEmail;
20     JTextField jtfName, jtfAddress, jtfPhone, jtfEmail;
21     JButton jbbSave, jbnDelete, jbnClear, jbnUpdate, jbnSearch, jbnForward,
jbnBack, jbnExit;
22     String name, address, email;
23     int phone;
24     int recordNumber; // used to navigate using >> and << buttons
25     Container cPane;
26
27     public static void main(String args[]){
28         new Interface();
29     }
30
31     public Interface(){
32         name = "";
33         address = "";
34         email = "";
35         phone = -1 ; //Stores 0 to indicate no Phone Number
36         recordNumber = -1;
37         createGUI();
38         personsList = new ArrayList();
39         // creating PersonDAO object
40         pDAO = new PersonDAO();
41     }
42
43     public void createGUI(){
44         /*Create a frame, get its contentpane and set layout*/
45         appFrame = new JFrame("Address Book");
46         cPane = appFrame.getContentPane();
47         cPane.setLayout(new GridBagLayout());
48         /*Arrange components on contentPane and set Action Listeners to each
JButton*/
49         arrangeComponents();
50         appFrame.setSize(240,300);
51         appFrame.setResizable(false);
52         appFrame.setVisible(true);
53         appFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
54     }
55
56     public void arrangeComponents(){
57         jlbName = new JLabel("Name");
58         jlbAddress = new JLabel("Address");
59         jlbPhone = new JLabel("Phone");
60         jlbEmail = new JLabel("Email");

```

Interface.java

```

61     jtfName = new JTextField(20);
62     jtfAddress = new JTextField(20);
63     jtfPhone = new JTextField(20);
64     jtfEmail = new JTextField(20);
65     jbbSave = new JButton("Save");
66     jbnDelete = new JButton("Delete");
67     jbnClear = new JButton("Clear");
68     jbnUpdate = new JButton("Update");
69     jbnSearch = new JButton("Search");
70     jbnForward = new JButton(">>");
71     jbnBack = new JButton("<<");
72     jbnExit = new JButton("Exit");
73     /*add all initialized components to the container*/
74     GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
75     gridBagConstraintsx01.gridx = 0;
76     gridBagConstraintsx01.gridy = 0;
77     gridBagConstraintsx01.insets = new Insets(5,5,5,5);
78     cPane.add(jlbName, gridBagConstraintsx01);
79     GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
80     gridBagConstraintsx02.gridx = 1;
81     gridBagConstraintsx02.insets = new Insets(5,5,5,5);
82     gridBagConstraintsx02.gridy = 0;
83     gridBagConstraintsx02.gridwidth = 2;
84     gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
85     cPane.add(jtfName, gridBagConstraintsx02);
86     GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
87     gridBagConstraintsx03.gridx = 0;
88     gridBagConstraintsx03.insets = new Insets(5,5,5,5);
89     gridBagConstraintsx03.gridy = 1;
90     cPane.add(jlbAddress, gridBagConstraintsx03);
91     GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
92     gridBagConstraintsx04.gridx = 1;
93     gridBagConstraintsx04.insets = new Insets(5,5,5,5);
94     gridBagConstraintsx04.gridy = 1;
95     gridBagConstraintsx04.gridwidth = 2;
96     gridBagConstraintsx04.fill = GridBagConstraints.BOTH;
97     cPane.add(jtfAddress, gridBagConstraintsx04);
98     GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
99     gridBagConstraintsx05.gridx = 0;
100    gridBagConstraintsx05.insets = new Insets(5,5,5,5);
101    gridBagConstraintsx05.gridy = 2;
102    cPane.add(jlbPhone, gridBagConstraintsx05);
103    GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
104    gridBagConstraintsx06.gridx = 1;
105    gridBagConstraintsx06.gridy = 2;
106    gridBagConstraintsx06.insets = new Insets(5,5,5,5);
107    gridBagConstraintsx06.gridwidth = 2;
108    gridBagConstraintsx06.fill = GridBagConstraints.BOTH;
109    cPane.add(jtfPhone, gridBagConstraintsx06);
110    GridBagConstraints gridBagConstraintsx07 = new GridBagConstraints();
111    gridBagConstraintsx07.gridx = 0;
112    gridBagConstraintsx07.insets = new Insets(5,5,5,5);
113    gridBagConstraintsx07.gridy = 3;
114    cPane.add(jlbEmail, gridBagConstraintsx07);
115    GridBagConstraints gridBagConstraintsx08 = new GridBagConstraints();
116    gridBagConstraintsx08.gridx = 1;
117    gridBagConstraintsx08.gridy = 3;
118    gridBagConstraintsx08.gridwidth = 2;
119    gridBagConstraintsx08.insets = new Insets(5,5,5,5);
120    gridBagConstraintsx08.fill = GridBagConstraints.BOTH;
121    cPane.add(jtfEmail, gridBagConstraintsx08);
122    GridBagConstraints gridBagConstraintsx09 = new GridBagConstraints();

```

Interface.java

```

123     gridBagConstraints09.gridx = 0;
124     gridBagConstraints09.gridy = 4;
125     gridBagConstraints09.insets = new Insets(5,5,5,5);
126     cPane.add(jbbSave, gridBagConstraints09);
127     GridBagConstraints gridBagConstraints10 = new GridBagConstraints();
128     gridBagConstraints10.gridx = 1;
129     gridBagConstraints10.gridy = 4;
130     gridBagConstraints10.insets = new Insets(5,5,5,5);
131     cPane.add(jbnDelete, gridBagConstraints10);
132     GridBagConstraints gridBagConstraints11 = new GridBagConstraints();
133     gridBagConstraints11.gridx = 2;
134     gridBagConstraints11.gridy = 4;
135     gridBagConstraints11.insets = new Insets(5,5,5,5);
136     cPane.add(jbnUpdate, gridBagConstraints11);
137     GridBagConstraints gridBagConstraints12 = new GridBagConstraints();
138     gridBagConstraints12.gridx = 0;
139     gridBagConstraints12.gridy = 5;
140     gridBagConstraints12.insets = new Insets(5,5,5,5);
141     cPane.add(jbnBack, gridBagConstraints12);
142     GridBagConstraints gridBagConstraints13 = new GridBagConstraints();
143     gridBagConstraints13.gridx = 1;
144     gridBagConstraints13.gridy = 5;
145     gridBagConstraints13.insets = new Insets(5,5,5,5);
146     cPane.add(jbnSearch, gridBagConstraints13);
147     GridBagConstraints gridBagConstraints14 = new GridBagConstraints();
148     gridBagConstraints14.gridx = 2;
149     gridBagConstraints14.gridy = 5;
150     gridBagConstraints14.insets = new Insets(5,5,5,5);
151     cPane.add(jbnForward, gridBagConstraints14);
152     GridBagConstraints gridBagConstraints15 = new GridBagConstraints();
153     gridBagConstraints15.gridx = 1;
154     gridBagConstraints15.insets = new Insets(5,5,5,5);
155     gridBagConstraints15.gridy = 6;
156     cPane.add(jbnClear, gridBagConstraints15);
157     GridBagConstraints gridBagConstraints16 = new GridBagConstraints();
158     gridBagConstraints16.gridx = 2;
159     gridBagConstraints16.gridy = 6;
160     gridBagConstraints16.insets = new Insets(5,5,5,5);
161     cPane.add(jbnExit, gridBagConstraints16);
162     jbbSave.addActionListener(this);
163     jbnDelete.addActionListener(this);
164     jbnClear.addActionListener(this);
165     jbnUpdate.addActionListener(this);
166     jbnSearch.addActionListener(this);
167     jbnForward.addActionListener(this);
168     jbnBack.addActionListener(this);
169     jbnExit.addActionListener(this);
170 }
171
172 public void actionPerformed (ActionEvent e){
173     if (e.getSource () == jbbSave){
174         savePerson();
175         clear();
176     }
177     else if (e.getSource() == jbnDelete){
178         deletePerson();
179         clear();
180     }
181     else if (e.getSource() == jbnUpdate){
182         updatePerson();
183         clear();
184     }

```

Interface.java

```

185     else if (e.getSource() == jbnSearch){
186         searchPerson();
187     }
188     else if (e.getSource() == jbnForward){
189         displayNextRecord();
190     }
191     else if (e.getSource() == jbnBack){
192         displayPreviousRecord();
193     }
194     else if (e.getSource() == jbnClear){
195         clear();
196     }
197     else if (e.getSource() == jbnExit){
198         System.exit(0);
199     }
200 }
201
202 // Save the Person into the Address Book
203 public void savePerson(){
204     name = jtfName.getText();
205     name = name.toUpperCase(); //Save all names in Uppercase
206     address = jtfAddress.getText();
207     try{
208         phone = Integer.parseInt(""+jtfPhone.getText());
209     }catch(Exception e){
210     }
211     email = jtfEmail.getText();
212     if(name.equals("")){
213         JOptionPane.showMessageDialog(null, "Please enter person name.");
214     }else{
215         //create a PersonInfo object and pass it to PersonDAO to save it
216         PersonInfo person = new PersonInfo(name, address, phone, email);
217         pDAO.savePerson(person);
218         JOptionPane.showMessageDialog(null, "Person Saved");
219     }
220 }
221
222 public void deletePerson(){
223
224     name = jtfName.getText();
225     name = name.toUpperCase();
226     if(name.equals("")){
227         JOptionPane.showMessageDialog(null, "Please enter person name to
delete.");
228     }
229     else{
230         //remove Person of the given name from the Address Book database
231         int numberOfDeleted = pDAO.removePerson(name);
232         JOptionPane.showMessageDialog(null, numberOfDeleted + " Record(s)
deleted.");
233     }
234 }
235
236 public void updatePerson(){
237     if (recordNumber >= 0 && recordNumber < personsList.size()){
238         PersonInfo person = (PersonInfo)personsList.get(recordNumber);
239         int id = person.getId();
240         /*get values from text fields*/
241         name = jtfName.getText();
242         address = jtfAddress.getText();
243         phone = Integer.parseInt(jtfPhone.getText());
244         email = jtfEmail.getText();

```

Interface.java

```

245         /*update data of the given person name*/
246         person = new PersonInfo(id, name, address, phone, email);
247         pDAO.updatePerson(person);
248         JOptionPane.showMessageDialog(null, "Person info record updated
successfully.");
249     }
250     else{
251         JOptionPane.showMessageDialog(null, "No record to Update");
252     }
253 }
254
255 //Perform a Case-Insensitive Search to find the Person
256 public void searchPerson() {
257     name = jtfName.getText();
258     name = name.toUpperCase();
259     /*clear contents of arraylist if there are any from previous search*/
260     personsList.clear();
261     recordNumber = 0;
262     if(name.equals("")){
263         JOptionPane.showMessageDialog(null, "Please enter person name to
search.");
264     }
265     else{
266         /*get an array list of searched persons using PersonDAO*/
267         personsList = pDAO.searchPerson(name);
268         if(personsList.size() == 0){
269             JOptionPane.showMessageDialog(null, "No records found.");
270             //Perform a clear if no records are found.
271             clear();
272         }
273         else{
274             /*downcast the object from array list to PersonInfo*/
275             PersonInfo person = (PersonInfo) personsList.get(recordNumber);
276             // displaying search record in text fields
277             jtfName.setText(person.getName());
278             jtfAddress.setText(person.getAddress());
279             jtfPhone.setText(" "+person.getPhone());
280             jtfEmail.setText(person.getEmail());
281         }
282     }
283 }
284
285 public void displayNextRecord(){
286     // inc in recordNumber to display next person info, already stored in
287     // personsList during search
288     recordNumber++;
289     if(recordNumber >= personsList.size()){
290         JOptionPane.showMessageDialog(null, "You have reached end of " +
"search results");
291         /*if user has reached the end of results, disable forward button*/
292         jbnForward.setEnabled(false);
293         jbnBack.setEnabled(true);
294         // dec by one to counter last inc
295         recordNumber -- ;
296     }
297     else{
298         jbnBack.setEnabled(true);
299         PersonInfo person = (PersonInfo) personsList.get(recordNumber);
300         // displaying search record in text fields
301         jtfName.setText(person.getName());
302         jtfAddress.setText(person.getAddress());
303         jtfPhone.setText(" "+person.getPhone());
304

```

Interface.java

```

305     jtfEmail.setText(person.getEmail());
306 }
307 }
308
309 public void displayPreviousRecord(){
310     // dec in recordNumber to display previous person info, already
311     // stored in personsList during search
312     recordNumber--;
313     if(recordNumber < 0 ){
314         JOptionPane.showMessageDialog(null, "You have reached begining " +
315             "of search results");
316         /*if user has reached the begining of results, disable back button*/
317         jbnForward.setEnabled(true);
318         jbnBack.setEnabled(false);
319         // inc by one to counter last dec
320         recordNumber++;
321     }else{
322         jbnForward.setEnabled(true);
323         PersonInfo person = (PersonInfo) personsList.get(recordNumber);
324         // displaying search record in text fields
325         jtfName.setText(person.getName());
326         jtfAddress.setText(person.getAddress());
327         jtfPhone.setText(" "+person.getPhone());
328         jtfEmail.setText(person.getEmail());
329     }
330 }
331
332 public void clear(){
333     jtfName.setText("");
334     jtfAddress.setText("");
335     jtfPhone.setText("");
336     jtfEmail.setText("");
337     /*clear contents of arraylist*/
338     recordNumber = -1;
339     personsList.clear();
340     jbnForward.setEnabled(true);
341     jbnBack.setEnabled(true);
342 }
343 }

```