```java
package project;

import java.awt.GridBagConstraints;

/* This is the class represent a Generic Form without any Components pre populated.
 *  rather components will be added using a FormConfigration Objects which
 *  hold the data to define the form. That is , it hold the data of the JTextField,JLabel,
 *  JButton (JComponent) etc */
public class GenericForm extends JPanel implements ActionListener {

    private static final long serialVersionUID = 1L;
    private int totalFields;
    /*This variable hold the reference to List of JComponents where the component is editable  in the
     *  visual form  */
    private List<JComponent> fieldList;
    /*This variable hold the reference to List of JComponents where the component is used as Labels
     * in the visual form  */
    private List<JComponent> messageList;
    private List<JComponent> labelList;
    /* This variable hold the reference to all the form related data */
    private FormConfiguration formConfigurator;
    /*Store all the Event Listener related to the FormSaveEvent */
    private EventListenerList listenersList=new EventListenerList();

    public GenericForm(){
        fieldList=new ArrayList<>();
        messageList=new ArrayList<>();
        labelList=new ArrayList<>();
    }

    public GenericForm(FormConfiguration formConfigurator){
        fieldList=new ArrayList<>();
        messageList=new ArrayList<>();
        labelList=new ArrayList<>();
        renderForm( formConfigurator);
    }

    /*This will use the data from the FormConfiguraton object and construct the form by adding the
     * appropriate components from the data */
    public void renderForm(FormConfiguration formConfigurator){
        this.formConfigurator=formConfigurator;
        /* Add FormConfiguration class to handle the FormSaveEvent */
        this.addFormSaveListener(formConfigurator);
        /* Get the total number of the Input Fields of the Form*/
        totalFields=formConfigurator.getTotalFields();
        /* Get the number of the columns of the Form*/
        int gridSize=formConfigurator.getGridSize();
        setBackground(formConfigurator.getFormColor());

        /* Setting the GridBagLayout as Layout Manager for the Panel */
```

```
63          setLayout(new GridBagLayout());
64          GridBagConstraints gc=new GridBagConstraints();
65
66
67          //Laying out all the Label fields
68          // define the GridBagContraints for the first column which is column=1
    row=<?>(all rows)
69          //The aim is to create n * m (row*columns)  Grid with the flexibility of
    varying width of the columns
70
71          gc.anchor=GridBagConstraints.FIRST_LINE_END;  //Right Align
72          gc.insets=new Insets(5,5,5,5);          //Margin 5 pixels from all the
    four sides(Top,Left,Bottom,Right)
73          gc.weightx=0.5;                         // Weightx determine the relative
    width from the other columns
74          gc.weighty=0.5;                         // Weighty determine the relative
    width from the other rows
75
76          //Laying the 1st column Label Fields
77          int rowCount=totalFields/gridSize;   //index start from 0 (0 to
    rowCount-1)
78          int colCount=gridSize;               //index start from 0 (0 to
    colCount-1)
79          int position;
80          for(int col=0;col<colCount;col++){
81              for (int row=0;row < rowCount;row++){
82                  gc.gridx=colCount*col ;
83                  gc.gridy=row ;
84
85                  position=row+(totalFields/colCount)*col;
86                  /* use position to locate the label from the formConfigurator
    object*/
87                  JComponent label=formConfigurator.getLabel(position);
88                  /* Add label component to the Panel(Form) */
89                  add(label,gc);
90                  labelList.add(position, label);
91
92                  if ((col==colCount-1) && (row==rowCount-1)  && (totalFields %
    gridSize>0)){
93                      gc.gridx=0;
94                      gc.gridy=rowCount;
95                      position+=1;
96                      label=formConfigurator.getLabel(position);
97                      add(label,gc);
98                      labelList.add(position, label);
99                  }
100
101             }
102         }
103
104         //Laying out all the Input Fields
105         // define the GridBagContraints for the first column which is column=1
    row=<?>(all rows)
106         //The aim is to create n * m (row*columns)  Grid with the flexibility of
    varying width of the columns
```

```
107
108         gc.anchor=GridBagConstraints.FIRST_LINE_START;  //Right Align
109         gc.insets=new Insets(5,5,5,5);          //Margin 5 pixels from all the
    four sides(Top,Left,Bottom,Right)
110         gc.weightx=0.5;                         // Weightx determine the relative
    width from the other columns
111         gc.weighty=0.5;                         // Weighty determine the relative
    width from the other rows
112         position=0;
113         for(int col=0;col<colCount;col++){
114             for (int row=0;row < totalFields/colCount;row++){
115                 gc.gridx=colCount*col+1 ;
116                 gc.gridy=row ;
117                 if (col==colCount-1) gc.weightx=5*colCount;
118                 position=row+(totalFields/colCount)*col;
119
120                 JComponent field=formConfigurator.getInputField(position);
121                 if ((field instanceof JTextArea)  ||(field instanceof
    RoundedTextArea)) {
122                     JScrollPane sPane=new
    JScrollPane(field,JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
123                             JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
124                     add(sPane,gc);
125                 }
126                 else add(field,gc);
127                 fieldList.add(position, field);
128
129                 if ((col==colCount-1) && (row==rowCount-1)  && (totalFields %
    gridSize>0)){
130                     gc.gridx=1;
131                     gc.gridy=rowCount;
132                     position+=1;
133                     field=formConfigurator.getInputField(position);
134                     if ((field instanceof JTextArea)  ||(field instanceof
    RoundedTextArea)) {
135                         JScrollPane sPane=new
    JScrollPane(field,JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
136                                 JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
137                         add(sPane,gc);
138                     }
139                     else add(field,gc);
140                     fieldList.add(position, field);
141                 }
142             }
143         }
144
145         /* Laying out Buttons, Location of the Button determined by button
    Location property of the
146          * formConfigurator Object */
147         gc.gridx=1;
148         gc.gridy=totalFields;
149         if (formConfigurator.getButtonsLocation()!=null &&
    formConfigurator.getButtonsLocation()[0].equals("side")){
150             gc.gridx=gridSize+1;
151             gc.gridy=totalFields-1;
```

```
152         }
153         gc.weighty=15;
154         gc.anchor=GridBagConstraints.FIRST_LINE_START;
155         add(formConfigurator.getButton(0, this),gc);
156 
157         /*Laying out any Messages(if avaialable) at the bottom of the Form */
158         for (int i=0;i<formConfigurator.getMessagesSize();i++){
159             gc.gridx=0;
160             gc.gridy=totalFields+1+i;
161             if (i==formConfigurator.getMessagesSize()-1) gc.weighty=200;
162             gc.weightx=colCount*5;
163             gc.gridwidth=2;
164             gc.anchor=GridBagConstraints.FIRST_LINE_START;
165             JComponent label=formConfigurator.getMessageLabel(i);
166             add(label,gc);
167             messageList.add(label);
168         }
169 
170 
171     }
172 
173     /******** Getters and Setters *************************/
174     public List<JComponent> getLabelList() {
175         return labelList;
176     }
177 
178     public void setLabelList(List<JComponent> labelList) {
179         this.labelList = labelList;
180     }
181 
182     public List<JComponent> getMessageList() {
183         return messageList;
184     }
185 
186     public void setMessageList(List<JComponent> messageList) {
187         this.messageList = messageList;
188     }
189 
190     public JComponent getFieldComponent(int index){
191         return fieldList.get(index);
192     }
193 
194     public FormConfiguration getFormConfigurator(){
195         return formConfigurator;
196     }
197 
198     public void setConfigurator(FormConfiguration formConfigurator){
199         this.formConfigurator=formConfigurator;
200     }
201     /******** End Of Getters and Setters *************************/
202 
203     /* This method provide the ability to other Objects(FormSaveEvent aware) to
    add their Listeners with
204      * this object*/
205     public void addFormSaveListener(FormSaveListener listener){
```

```
206         listenersList.add(FormSaveListener.class, listener);
207     }
208 
209     /* This method provide the ability to other Objects(FormSaveEvent aware) to remove their Listeners from
210      * this object*/
211     public void removeFormSaveListener(FormSaveListener listener){
212         listenersList.remove(FormSaveListener.class, listener);
213     }
214 
215     /*This is the method being called when  FormSaveEvent is fired on this object*/
216     public void fireFormSaveEvent(FormSaveEvent event){
217         Object[] listeners=listenersList.getListenerList();
218         for (int i=0;i<listeners.length;i +=2){
219             if (listeners[i]==FormSaveListener.class){
220                 ((FormSaveListener)listeners[i+1]).FormSaveOccured(event);
221             }
222 
223         }
224     }
225 
226     /* This method extract the data from the formDataObject and bind to the respective
227      * input fields in the form . This delegate the bind task to the underlying formConfigurator obect*/
228     public void bind(Object formDataObject){
229         formConfigurator.bind(formDataObject, this);
230     }
231 
232     /*This method is being called when the form layout is changed */
233     public void reRender(FormConfiguration formConfiguration){
234         Object formDataObject=formConfigurator.getFormData();
235         this.removeAll();
236         renderForm(formConfiguration);
237         formConfigurator.bind(formDataObject, this);
238         this.revalidate();
239     }
240 
241     /* This is the method is being called when the button is clicked
242      * The action has been delegated to formconfiguration object to handle it outside the form*/
243     @Override
244     public void actionPerformed(ActionEvent e) {
245         String actionName=e.getActionCommand();
246         if (actionName.equalsIgnoreCase("SAVE")||actionName.equalsIgnoreCase("Login") || actionName.equalsIgnoreCase("Search")){
247             this.fireFormSaveEvent(new FormSaveEvent(this, "formDeatils"));
248         }
249     }
250 }
251 
```