```java
package project;

import java.awt.Color;

/* This class contains necessary data to construct a Form base Interface.
 * The data describe the Input Fields(JTextField,JRadioButton,JComboBox..etc), Labels for the Fields,
 * Buttons, Prompt Messages & Error Messages to the User, Input Fields Validators, Input Fields sizes,
 * Directions of the Radio Button group or Check boxes (possible directions are Vertical or Horizontal),
 * Locations of the Buttons position (Whether at the bottom of the form or Side),
 * Options values(display and actual values) for RadioButtons or ComboBoxes etc, Input Filed's Class Type,
 * Color of the Form, Actual data of the entity that it represent (Data of the Inputs) */
public class FormConfiguration implements FormSaveListener {

    private String[] labelNames;        //Contains all the Labels of the Underlying Form Interface
    private String[] fieldNames;        //Contains all the Input Fields of the Underlying Form Interface
    private Class<?>[] fieldClass;      //Contains all Class Types(Input Fields) of the Underlying Form Interface
    private int[] fieldSize;            //Contains all the Input Filed Size of the Underlying Form Interface
    private int[] verifiers;            //Contains all the Inputs Validators of the Underlying Form Interface
    private int[] directions;
    private String[] buttons;           //Contains all the Buttons of the Underlying Form Interface
    private String[] buttonsLocation;
    private String[] message;           //Contains all the Prompt Messages of the Underlying Form Interface
    private String[] errorMessage;      //Contains all the Error Messages of the Underlying Form Interface
    private String[] messageLabel;

    private Hashtable<Integer,List<String>> fieldOptionsValue;
    private Hashtable<Integer,List<String>> fieldOptionsDisplay;
    /* The object(Entity) where the Form Interface store the data*/
    private Object formDataObject;
    /* The class of the underlying Entity which the Form Interface store the data*/
    private Class<?> formClass;
    private String formName;
    private Color formColor;
    /* No of the Columns in the underlying form*/
    private int gridSize;
    /* Whether the form is editable or not, Some forms are made Readonly while others are editable
     * This is set by the Business rules of the user*/
    private boolean isEdit=true;
    /* Whether mapping is used when the Labels ,Field Data are being read from the external file*/
```

```
53      private boolean mapping=false;
54
55      /*This is an In Memory Persistance Storage for the Entity(Collection Of Input
  Fields) that this class
56       *  represent through the Form Interface (GenericForm Class)*  */
57      private Repository repository;
58      private int newID;
59      /* Action Listener which will be used to Handle Action commands in the
  underlying form*/
60      private ActionListener actionListener;
61
62      public FormConfiguration(){}
63
64
65      // *************** All the Getters and Setters *************************//
66      public String[] getLabelNames() {
69
70      public void setLabelNames(String[] labelNames){
73
74      public String[] getFieldNames() {
77
78      public void setFieldNames(String[] fieldNames){
81
82      public Class<?>[] getFieldClass() {
85
86      public void setFieldClass(Class<?>[] fieldClass){
89
90      public int[] getFieldSize() {
93
94      public void setFieldSize(int[] fieldSize){
97
98      public int[] getVerifiers() {
101
102     public void setVerifiers(int[] verifiers) {
105
106     public int[] getDirections() {
109
110     public void setDirections(int[] directions) {
113
114     public String[] getButtons() {
117
118     public void setButtons(String[] buttons) {
121
122     public String[] getButtonsLocation() {
125
126     public void setButtonsLocation(String[] buttonsLocation) {
129
130     public Class<?> getFormClass(){
133
134     public void setFormClass(Class<?> formClass){
137
138     public Color getFormColor() {
141
142     public void setFormColor(Color formColor) {
145
```

```
146     public String getFormName() {
149
150     public void setFormName(String formName) {
153
154     public String[] getMessageLabel() {
157
158     public void setMessageLabel(String[] messageLabel) {
161
162     public String[] getMessage() {
165
166     public void setMessage(String[] message) {
169
170     public String[] getErrorMessage() {
173
174     public void setErrorMessage(String[] errorMessage) {
177
178     public boolean isMapping() {
181
182     public void setMapping(boolean mapping) {
185
186     public boolean isEdit() {
189
190     public void setEdit(boolean isEdit) {
193
194     public Object getFormData(){
197
198     public void setFormData(Object formDataObject){
201
202     public Hashtable<Integer, List<String>> getFieldOptionsValue() {
205
206     public void setFieldOptionsValue(Hashtable<Integer,List<String>>
    fieldOptionsValue){
209
210     public Hashtable<Integer, List<String>> getFieldOptionsDisplay() {
213
214     public void setFieldOptionsDisplay(Hashtable<Integer,List<String>>
    fieldOptionsDisplay){
217
218     public int getGridSize(){
221
222     public void setGridSize(int gridSize){
225
226     public ActionListener getActionListener() {
229
230     public void setActionListener(ActionListener actionListener) {
233
234     public Repository getRepository(){
237
238     public void setRepository(Repository repository){
241
242     // *************** End Of All the Getters and Setters
    *************************//
243
244     /* Get the TotalInput Fields in the form */
245     public int getTotalFields(){
```

```
248
249     /* Get the size of the Message Labels in the Form */
250     public int getMessagesSize() {
254
255
256     /* This  Method is called whenever GenericForm class need to add the Label
    Component into the Form Layout
257      * Basically it create the JLabel component and return it to the Generic Form
    Object*/
258     public JComponent getLabel(int index){
263
264     /* This is the Method basically creates the Input Field Component and set all
    the necessary
265      * properties such as Field Size,Field Verifier etc. Also it populate all the
    options literals for the
266      * Radio Button,Combo Box etc */
267     public JComponent getFieldComponent(int index){
307
308     /* This  Method is called whenever GenericForm class need to add the Input
    Field Component
309      * into the Form Layout*/
310     public JComponent getInputField(int index){
323
324     /* This method is called Whenever Genericform class need to add the
    Messages(Prompt Message or Error Message)
325      * into the Form layout*/
326     public JComponent getMessageLabel(int index){
342
343     /* This  Method is called whenever GenericForm class need to add the Buttons
    into the Form Layout*/
344     public JComponent getButton(int index, ActionListener listener){
353
354     /*This method return the value of the Component , If it is TextComponent , it
    return
355      * the contents of the TextComponent , If it is Radio Buttons, it return the
    the selected value.*/
356     public String getFieldValue(JComponent component, int index){
370
371     /* This method initialise all the input Fields on the Form */
372     private void formIntialise(GenericForm form ){
390
391     /*This Method bind the FormDataObject (Which is the Entity which store the
    data of the Form) with the
392      * Input Fields of the Form. Basically it synchronize the FormDataObject with
    the Input Fields of the Form */
393     public void bindForm(GenericForm form){
434
435     /* This method call the above method to bind the object with the form*/
436     public void bind(Object formDataObject,GenericForm form){
440
441     /* This method is called everytime form is saved using the save button*/
442     private void formSave(GenericForm form){
467
468
469     /* This is the Event Handler for the FormSaveEvent. This Event is fired when
```

```
   buttons are clicked on the
470      * Form.  */
472     public void FormSaveOccured(FormSaveEvent event) {
490
491     public int fromOptionValueToIndex(String value,int fieldIndex){
496
497     /* The following two methods are used to get the parameters and their values
   for the
498      * Object's Method type using Reflection, These are being used in the method
   "formSave" */
499     public  Class<?>[] getMethodParameters(String methodName){
510     public Object[] getMethodParametersValue(Class<?>[] parameters,String value){
529 }
530
```