```java
package Test;

import static org.junit.Assert.*;

public class FormFactoryTest {
    List<ConfigAttribute> configAttributes;

    @Before
    public void setUp() throws Exception {

    }

    @Test
    public void CSVParserTest() {
        /* This will Test the csvParser method whether it produce the correct configuration
         * attributes by reading an external text file using the ConfigAttribute Class
         * Basically csvParser Method is a staic method in FormFactory */
        /* The content of the file is given below
         * ********  Participant Form Configuration
            action=formParsing
            formName=Participant
            labelNames=ID,How old are you?,What is your Gender?,What is your ethnicity?,What is your Occupation?,Country Of Residence,Country Of Origin,What is the highest level of education you have completed?,RelatioshipValue
            fieldNames=Id,Age,Gender,Ethnicity,Occupation,Country,Origin,Education,RelationshipFitnessValue
            fieldClass=1,1,5,1,1,1,1,2,1
            fieldSize=5,5,20,20,20,20,20,25,5
            verifiers=0,1,0,2,2,2,2,2,2
            directions=0,0,2,2,0,0,0,0,0
            formClass=project.Participant
            gridSize=1
            options1=Male,Female,Trans Male,Trans Female,Other
            optionsV=1,2
            optionsD=1,2
            buttons=Save,Save
            color=186,219,241 */

        /*Test the number of the attributes read from the File and all the Attributes must have
         * a formID=1*/
        FormFactory.unID=0;
        FormFactory formFactory=new FormFactory("configParsing.txt");
        configAttributes=formFactory.csvParser();

        assertEquals(32, configAttributes.size());
        /* loop through the attributes randomly and Test it against the original values */
        ConfigAttribute configAttribute=configAttributes.get(0);
        String[] attributeValues=configAttribute.getAttributeValue();
        assertEquals("action",configAttribute.getAttributeName());
        assertEquals(1,attributeValues.length);
        assertEquals("formParsing",attributeValues[0]);
```

```
64          assertEquals(1,configAttribute.getFormID());
65
66          configAttribute=configAttributes.get(2);
67          attributeValues=configAttribute.getAttributeValue();
68          assertEquals("labelNames",configAttribute.getAttributeName());
69          assertEquals(9,attributeValues.length);
70          assertEquals("ID",attributeValues[0]);
71          assertEquals("What is your ethnicity?",attributeValues[3]);
72          assertEquals(1,configAttribute.getFormID());
73
74          configAttribute=configAttributes.get(14);
75          attributeValues=configAttribute.getAttributeValue();
76          assertEquals("color",configAttribute.getAttributeName());
77          assertEquals(3,attributeValues.length);
78          assertEquals("186",attributeValues[0]);
79          assertEquals("241",attributeValues[2]);
80          assertEquals(1,configAttribute.getFormID());
81      }
82
83      @Test
84      public void CSVParserTestForMultipleForms() {
85          /* This will test for correct reading of Multiple forms as Collection of
    Config Attributes
86           * But Respective Config attributes have seperate Unique ID for seperate
    forms*/
87          /* This must have 15 Config attributes for Participant Form and 17 Config
    Attributes
88           * for Relatioship Form
89           * The FormID should be for Participant equals 1 and Relationship equals
    2*/
90
91          FormFactory.unID=0;
92          FormFactory formFactory=new FormFactory("configParsing.txt");
93          configAttributes=formFactory.csvParser();
94
95          assertEquals(32, configAttributes.size());
96
97          /* This is for the Participant form from index=0 to 14*/
98          ConfigAttribute configAttribute=configAttributes.get(1);
99          String[] attributeValues=configAttribute.getAttributeValue();
100         assertEquals("Participant",attributeValues[0]);
101         assertEquals(1,configAttribute.getFormID());
102
103
104         /* This is for the Participant form from index=15 to 31*/
105         configAttribute=configAttributes.get(16);
106         attributeValues=configAttribute.getAttributeValue();
107         assertEquals("Relationship",attributeValues[0]);
108         assertEquals(2,configAttribute.getFormID());
109
110     }
111
112     @Test
113     public void MapToConfigTest() {
114         /* MapToConfig method should create different formIds for differnt action.
```

```
115         * For an example, if the action is mapping then it should create formID=100 or 200 or 300
116         * depend on how many differnt mappings blocks the CSV file have. in our case , there are
117         * 2 mapping blocks, so it should create attributes with formID=100,200
118         */
119        /*There are 51 attributes in the first mapping block, first one should be
120         * "action=mapping" attribute  with the formID=100*/
121        FormFactory.unID=0;
122        FormFactory formFactory=new FormFactory("configMapping.txt");
123        configAttributes=formFactory.csvParser();
124
125        ConfigAttribute configAttribute;
126        String[] attributeValues;
127        configAttribute=configAttributes.get(0);
128        attributeValues=configAttribute.getAttributeValue();
129        assertEquals("action",configAttribute.getAttributeName());
130        assertEquals("mapping",attributeValues[0]);
131        assertEquals(100,configAttribute.getFormID());
132
133        configAttribute=configAttributes.get(2);
134        attributeValues=configAttribute.getAttributeValue();
135        assertEquals("1.1",configAttribute.getAttributeName());
136        assertEquals("How old are you?",attributeValues[0]);
137        assertEquals(100,configAttribute.getFormID());
138
139        /*Next Mapping block should start width formID=200
140         * t start with index=51 that is 52nd attribute*/
141        configAttribute=configAttributes.get(51);
142        attributeValues=configAttribute.getAttributeValue();
143        assertEquals("action",configAttribute.getAttributeName());
144        assertEquals("mapping",attributeValues[0]);
145        assertEquals(200,configAttribute.getFormID());
146
147        configAttribute=configAttributes.get(53);
148        attributeValues=configAttribute.getAttributeValue();
149        assertEquals("1.1",configAttribute.getAttributeName());
150        assertEquals("Age",attributeValues[0]);
151        assertEquals(200,configAttribute.getFormID());
152
153    }
154
155    /* This is Test validate the getOptionsValues method
156     * this method read the option values set for different fields (Radio buttons,Combobox etc)
157     * in the text file*/
158    @Test
159    public void getOptionsValuesTest(){
160
161        FormFactory.unID=0;
162        FormFactory formFactory=new FormFactory("configMapping.txt");
163        configAttributes=formFactory.csvParser();
164        /*The "optionsArray" is the array simulating the data it read from the file
165         * Below it read as a pair from the array . 1st pair (1,11) tell the
```

```
   method to read contents
166         * of the "Options1" as a List and assign to "Field 11"
167         * The values for the options in the text file as follows.
168         * Options1=Yes,No
169         * Options2=1,0
170         * options3=1,2,3,4,5   */
171        int[] optionsArray=new int[]{1,11,3,12,2,13};
172        Hashtable<Integer,List<String>>
   ht=formFactory.getOptionsValues(optionsArray, configAttributes, 202);
173        /* For the field 11, it should call options1 */
174        List<String> option1=ht.get(11);
175        assertEquals(2, option1.size());
176        assertEquals("Yes", option1.get(0));
177        assertEquals("No", option1.get(1));
178        /* For the field 12, it should call options3 */
179        option1=ht.get(12);
180        assertEquals(5, option1.size());
181        assertEquals("1", option1.get(0));
182        assertEquals("2", option1.get(1));
183        assertEquals("3", option1.get(2));
184        assertEquals("4", option1.get(3));
185        assertEquals("5", option1.get(4));
186        /* For the field 13, it should call options2 */
187        option1=ht.get(13);
188        assertEquals(2, option1.size());
189        assertEquals("1", option1.get(0));
190        assertEquals("0", option1.get(1));
191
192    }
193
194    /*This test validate the createFormConfigurators Method. This method create
   FormConfiguration objects
195     * collection from the Text file. There are three FormConfiguration objects
   will be created
196     * as there are only three forms configuration defined in the text file. The
   form names defined are
197     * Participant,Relationship,Questionnaire*/
198    @Test
199    public void createFormConfiguratorsTest(){
200        FormFactory.unID=0;
201        FormFactory formFactory=new FormFactory("configMapping.txt");
202        List<FormConfiguration>
   formConfigurators=formFactory.createFormConfigurators();
203        FormConfiguration formConfiguration=formConfigurators.get(0);
204        FormConfiguration formConfiguration1=formConfigurators.get(1);
205        FormConfiguration formConfiguration2=formConfigurators.get(2);
206
207        assertEquals("Participant", formConfiguration.getFormName());
208        assertEquals("Relationship", formConfiguration1.getFormName());
209        assertEquals("Questionnaire", formConfiguration2.getFormName());
210
211        assertEquals(Participant.class, formConfiguration.getFormClass());
212        assertEquals(new Color(186,219,241), formConfiguration.getFormColor());
213
214        assertEquals(9, formConfiguration.getLabelNames().length);
```

```
215         assertEquals("ID", formConfiguration.getLabelNames()[0]);
216         assertEquals("How old are you?", formConfiguration.getLabelNames()[1]);
217         assertEquals("What is your Gender?", formConfiguration.getLabelNames()
    [2]);
218         assertEquals("RelatioshipValue", formConfiguration.getLabelNames()[8]);
219 
220         assertEquals(9, formConfiguration.getFieldNames().length);
221         assertEquals("Id", formConfiguration.getFieldNames()[0]);
222         assertEquals("Age", formConfiguration.getFieldNames()[1]);
223         assertEquals("Gender", formConfiguration.getFieldNames()[2]);
224         assertEquals("RelationshipFitnessValue", formConfiguration.getFieldNames()
    [8]);
225 
226         assertEquals(9, formConfiguration.getFieldClass().length);
227         assertEquals(JTextField.class, formConfiguration.getFieldClass()[0]);
228         assertEquals(JTextField.class, formConfiguration.getFieldClass()[1]);
229         assertEquals(RadioButtons.class, formConfiguration.getFieldClass()[2]);
230         assertEquals(JTextField.class, formConfiguration.getFieldClass()[8]);
231 
232         assertEquals(9, formConfiguration.getFieldSize().length);
233         assertEquals(5, formConfiguration.getFieldSize()[0]);
234         assertEquals(5, formConfiguration.getFieldSize()[1]);
235         assertEquals(20, formConfiguration.getFieldSize()[2]);
236         assertEquals(5, formConfiguration.getFieldSize()[8]);
237 
238         assertEquals(9, formConfiguration.getVerifiers().length);
239         assertEquals(0, formConfiguration.getVerifiers()[0]);
240         assertEquals(1, formConfiguration.getVerifiers()[1]);
241         assertEquals(0, formConfiguration.getVerifiers()[2]);
242         assertEquals(2, formConfiguration.getVerifiers()[8]);
243 
244 
245     }
246 
247 
248 }
249 
```