

```
1 package project3;
2
3 import org.junit.Before;
4 import org.junit.Test;
5
6 import java.text.SimpleDateFormat;
7 import java.util.Date;
8 import java.util.GregorianCalendar;
9
10 import static org.junit.Assert.*;
11
12 public class ListEngineTest {
13
14     SimpleDateFormat df;
15     GregorianCalendar date1;
16     GregorianCalendar date2;
17     ListEngine DList;
18
19     @Before
20     public void before() {
21         df = new SimpleDateFormat("MM/dd/yyyy");
22         date1 = new GregorianCalendar();
23         date2 = new GregorianCalendar();
24         try {
25             Date d1 = df.parse("3/20/2019");
26             date1.setTime(d1);
27             Date d2 = df.parse("9/20/2019");
28             date2.setTime(d2);
29         } catch (Exception e) {
30             //This is needed in order to parse something in case it fails
31         }
32         DList = new ListEngine();
33
34         //sell a car
35         Date d;
36         try{
37             d = df.parse("1/2/2030");
38         } catch(Exception e){
39             return;
40         }
41         GregorianCalendar temp = new GregorianCalendar();
42         temp.setTime(d);
43         Auto auto = DList.get(0);
44         auto.setNameOfBuyer("John");
45         auto.setSoldOn(temp);
46         auto.setSoldPrice(1000);
47
48     }
49
50     @Test
51     public void getColumnBought() {
52         String col = DList.getColumn(5);
53
54         assertEquals("Turbo", col);
55     }
56 }
```

```

57     @Test
58     public void getColumnNameSold() {
59         DList.updateDisplay(DList.soldScreen);
60         String col = DList.getColumnName(5);
61
62         assertEquals("Sold On", col);
63     }
64
65     @Test
66     public void getColumnNameOverdue() {
67         DList.updateDisplay(DList.overdueScreen);
68         String col = DList.getColumnName(3);
69
70         assertEquals("Days Overdue", col);
71     }
72
73     @Test (expected = RuntimeException.class)
74     public void getColumnNameOutOfRange() {
75         String col = DList.getColumnName(10);
76     }
77
78     @Test
79     public void updateDisplay() {
80         DList.updateDisplay(DList.soldScreen);
81
82         assertEquals(DList.soldScreen, DList.displayValue);
83     }
84
85     @Test
86     public void add() {
87         Car Car1 = new Car(date1, "Outback", "Buyer1", "LX", false);
88         Car Car2 = new Car(date2, "Chevy", "Buyer2", "EX", false);
89         DList.add(Car1);
90         DList.add(Car2);
91
92         // 8 total but only 7 in bought screen
93         assertEquals(7, DList.getSize());
94     }
95
96     @Test
97     public void get() {
98         Auto first = DList.get(0);
99
100        assertEquals(first.autoName, "F350");
101    }
102
103    @Test
104    public void getSize() {
105        int size = DList.getSize();
106
107        assertEquals(6, size);
108    }
109
110    @Test
111    public void getRowCount() {
112        int numRows = DList.getRowCount();

```

```
113
114     assertEquals(6, numRows);
115 }
116
117 @Test
118 public void getColumnCountBought() {
119     int numColumns = DList.getColumnCount();
120
121     assertEquals(6, numColumns);
122 }
123
124 @Test
125 public void getColumnCountSold() {
126     DList.updateDisplay(DList.soldScreen);
127     int numColumns = DList.getColumnCount();
128
129     assertEquals(6, numColumns);
130 }
131
132 @Test
133 public void getColumnCountOverdue() {
134     DList.updateDisplay(DList.overdueScreen);
135     int numColumns = DList.getColumnCount();
136
137     assertEquals(4, numColumns);
138 }
139
140
141 // cols 0-2 any screen
142 // col 3 all 3
143 // col 4 & 5 bought sold
144
145 // bought - col 0, 1, 2, 3, 4, 5
146 // sold - 3, 4, 5
147 // overdue - 3
148 // out of range
149 @Test
150 public void getValueAtBought0() {
151     String val = DList.getValueAt(0, 0).toString();
152
153     assertEquals(val, "F350");
154 }
155
156 @Test
157 public void getValueAtBought1() {
158     String val = DList.getValueAt(0, 1).toString();
159
160     assertEquals(val, "0.0");
161 }
162
163 @Test
164 public void getValueAtBought2() {
165     String val = DList.getValueAt(0, 2).toString();
166
167     assertEquals(val, "01/20/2010");
168 }
```

```
169
170     @Test
171     public void getValueAtBought3() {
172         String val = DList.getValueAt(0, 3).toString();
173
174         assertEquals(val, "EX");
175     }
176
177     @Test
178     public void getValueAtBought4Truck() {
179         String val = DList.getValueAt(0, 4).toString();
180
181         assertEquals(val, "true");
182     }
183
184     @Test
185     public void getValueAtBought4Car() {
186         String val = DList.getValueAt(1, 4).toString();
187
188         assertEquals(val, "");
189     }
190
191     @Test
192     public void getValueAtBought5Truck() {
193         String val = DList.getValueAt(0, 5).toString();
194
195         assertEquals(val, "");
196     }
197
198     @Test
199     public void getValueAtBought5Car() {
200         String val = DList.getValueAt(1, 5).toString();
201
202         assertEquals(val, "false");
203     }
204
205     @Test
206     public void getValueAtSold3() {
207         DList.updateDisplay(DList.soldScreen);
208         String val = DList.getValueAt(0, 3).toString();
209
210         assertEquals(val, "John");
211     }
212
213     @Test
214     public void getValueAtSold4() {
215         DList.updateDisplay(DList.soldScreen);
216         String val = DList.getValueAt(0, 4).toString();
217
218         assertEquals(val, "1000.0");
219     }
220
221     @Test
222     public void getValueAtSold5() {
223         DList.updateDisplay(DList.soldScreen);
224         String val = DList.getValueAt(0, 5).toString();
```

```
225
226     assertEquals(val, "01/02/2030");
227 }
228
229 @Test
230 public void getValueAtOverdue3() {
231     DList.updateDisplay(DList.overdueScreen);
232     String val = DList.getValueAt(0, 3).toString();
233
234     assertEquals(val, "316");
235 }
236
237 @Test (expected = RuntimeException.class)
238 public void getValueAtOutOfRange() {
239     String val = DList.getValueAt(0, 10).toString();
240 }
241
242 @Test
243 public void saveDatabase() {
244     DList.saveDatabase("JUnit test");
245 }
246
247 @Test
248 public void loadDatabase() {
249     DList.loadDatabase("JUnit test");
250     String val = DList.get(0).getAutoName();
251
252     assertEquals("F350", val);
253 }
254
255 @Test
256 public void saveAsTextError() {
257     boolean b = DList.saveAsText("");
258
259     assertFalse(b);
260 }
261
262 @Test
263 public void saveAsText() {
264     DList.saveAsText("JUnit text test");
265 }
266
267 @Test
268 public void loadFromText() {
269     DList.loadFromText("JUnit text test");
270     String val = DList.get(0).getAutoName();
271
272     assertEquals("Outback", val);
273 }
274 }
275
```