File - D:\Users\gjr8050\262refactoring\test\Main\Frame\FrameTest.java

```
1  package Main.Frame;
2
3  import org.junit.Assert;
4  import org.junit.Before;
5  import org.junit.Test;
6
7  import java.util.ArrayList;
8
9  import static org.junit.Assert.*;
10
11 /**
12  * Created by gjr8050 on 5/1/2016.
13  */
14 public class FrameTest {
15
16     Frame frame;
17
18     private ArrayList<Frame> frames;
19
20     @Before
21     public void setUp() throws Exception {
22         frame = new Frame();
23         frames = new ArrayList<>();
24         for(int i = 0; i < 3; i++){
25             Frame frame = new Frame();
26             frames.add(frame);
27             if(i > 0){
28                 frames.get(i - 1).setNext(frame);
29                 frame.setPrev(frames.get(i - 1));
30             }
31         }
32     }
33
34     @Test
35     public void setBallPinCount() throws Exception{
36         int pinCount1 = 5;
37         frame.setPinCount(Ball.ONE, pinCount1);
38         Assert.assertEquals("Expected pin count for Ball.ONE", pinCount1, frame.getPinCount(
   Ball.ONE));
39     }
40
41     @Test
42     public void setPinCountActive() throws Exception {
43         int pinCount1 = 5;
44         frame.setPinCount(pinCount1);
45         Assert.assertEquals("Expected pin count for Ball.ONE", pinCount1, frame.getPinCount(
   Ball.ONE));
46
47         int pinCount2 = 3;
48         frame.setPinCount(pinCount2);
49         Assert.assertEquals("Expected pin count for Ball.TWO", pinCount2, frame.getPinCount(
   Ball.TWO));
50     }
51
52     @Test
53     public void setPinCountChain() throws Exception {
54         Frame frame = new Frame();
55         Frame frame2 = new Frame();
56         frame.setNext(frame2);
57         frame2.setPrev(frame);
```

```
58
59          int pinCount = 3;
60          int pinCount2 = 2;
61
62          frame.setPinCount(5);
63          frame.setPinCount(4);
64          frame.setPinCount(pinCount);
65          frame.setPinCount(pinCount2);
66
67          Assert.assertEquals("Expected pin count for frame 2 Ball.ONE", pinCount, frame2.
    getPinCount(Ball.ONE));
68          Assert.assertEquals("Expected pin count for frame 2 Ball.TWO", pinCount2, frame2.
    getPinCount(Ball.TWO));
69      }
70
71      @Test(expected = UnsupportedOperationException.class)
72      public void setPinCountLast() {
73          frame.setPinCount(1);
74          frame.setPinCount(1);
75          //We should not be allowed to set a third ball on the last frame w/o strike or
    spare
76          frame.setPinCount(1);
77      }
78
79      @Test
80      public void isBowled() throws Exception {
81          assert !frame.isBowled() : "Expected frame to not be bowled";
82          frame.setPinCount(1);
83          frame.setPinCount(1);
84
85          assert frame.isBowled() : "Expected frame to be bowled";
86          assert !frame.isStrike() : "Expected frame to not be strike";
87          assert !frame.isSpare() : "Expected frame to not be a spare";
88      }
89
90      @Test
91      public void isStrike() throws Exception {
92          frame.setPinCount(10);
93
94          assert frame.isBowled() : "Expected frame to be bowled";
95          assert frame.isStrike() : "Expected frame to be strike";
96          assert !frame.isSpare() : "Expected frame to not be a spare";
97      }
98
99      @Test
100     public void isSpare() throws Exception {
101         frame.setPinCount(5);
102         frame.setPinCount(5);
103
104         assert frame.isBowled() : "Expected frame to be bowled";
105         assert !frame.isStrike() : "Expected frame to not be strike";
106         assert frame.isSpare() : "Expected frame to be a spare";
107     }
108
109     @Test
110     public void setPinCountLastSpare() throws Exception {
111         frame.setPinCount(5);
112         frame.setPinCount(5);
113
114         int pinCount3 = 4;
```

```
115         frame.setPinCount(pinCount3);
116
117         assert frame.getPinCount(Ball.THREE) == pinCount3 : "Expected Ball.THREE to have
    pin count of " + pinCount3;
118         assert frame.isSpare() : "Expected frame to be a spare";
119     }
120
121     @Test
122     public void setPinCountLastStrike() throws Exception {
123         frame.setPinCount(10);
124
125         int pinCount2 = 5;
126         frame.setPinCount(pinCount2);
127
128         int pinCount3 = 4;
129         frame.setPinCount(pinCount3);
130
131         assert frame.getPinCount(Ball.TWO) == pinCount2 : "Expected Ball.TWO to have pin
    count of " + pinCount2;
132         assert frame.getPinCount(Ball.THREE) == pinCount3 : "Expected Ball.THREE to have
    pin count of " + pinCount3;
133         assert frame.isStrike() : "Expected frame to be strike";
134     }
135
136     @Test
137     public void getBaseScore() throws Exception {
138         int pinCount = 5;
139         frame.setPinCount(pinCount);
140         int pinCount2 = 4;
141         frame.setPinCount(pinCount2);
142
143         Assert.assertEquals("Frame did not return expected base score", pinCount +
    pinCount2, frame.getBaseScore());
144     }
145
146     @Test
147     public void getBaseScoreLast() throws Exception {
148         int pinCount = 10;
149         frame.setPinCount(pinCount);
150         int pinCount2 = 5;
151         frame.setPinCount(pinCount2);
152         int pinCount3 = 4;
153         frame.setPinCount(pinCount3);
154
155         Assert.assertEquals("Frame did not return expected base score", pinCount +
    pinCount2 + pinCount3, frame.getBaseScore());
156     }
157
158     @Test
159     public void isBowled1() throws Exception {
160         frame.setPinCount(1);
161
162         assert frame.isBowled(Ball.ONE) : "Expected Ball.ONE to be bowled";
163         assert !frame.isBowled(Ball.TWO) : "Expected Ball.TWO to not be bowled";
164         assert !frame.isBowled(Ball.THREE) : "Expected Ball.THREE to not be bowled";
165     }
166
167     @Test
168     public void isLast() throws Exception {
169         Frame frame1 = new Frame();
```

```
170         Frame frame2 = new Frame();
171
172         frame1.setNext(frame2);
173         frame2.setPrev(frame1);
174
175         assert !frame1.isLast() : "Expected frame 1 to not be last";
176         assert frame2.isLast() : "Expected frame 2 to be last";
177     }
178
179     @Test
180     public void getChainScore() throws Exception {
181
182     }
183
184     @Test
185     public void setFrameState() throws Exception {
186         frame.setFrameState(FrameState.BOWLED);
187         assert frame.isBowled() : "Expected frame to have state of Bowled";
188     }
189
190     @Test
191     public void setPinCount1() throws Exception {
192
193     }
194
195     @Test
196     public void calculateFrameScore() throws Exception {
197         Assert.assertEquals("Expected frame score to be Unset", Frame.Unset, frame.
    calculateFrameScore());
198     }
199
200     @Test
201     public void getTotalScore() throws Exception {
202         Assert.assertEquals("Expected total score to be Unset", Frame.Unset, frame.
    getTotalScore());
203     }
204
205     @Test
206     public void calculateTotalScoreActive() throws Exception {
207         frame.calculateTotalScore();
208         Assert.assertEquals("Expected total score to be Unset", Frame.Unset, frame.
    getTotalScore());
209     }
210
211     @Test
212     public void calculateTotalScore() throws Exception {
213         Frame frame1 = new Frame();
214         Frame frame2 = new Frame();
215
216         frame1.setNext(frame2);
217         frame2.setPrev(frame1);
218
219         frame1.setPinCount(1);
220         frame1.setPinCount(1);
221         frame1.setPinCount(1);
222         frame1.setPinCount(1);
223
224         frame1.calculateTotalScore();
225         Assert.assertEquals("Unexpected total score for frame 1", 2, frame1.getTotalScore()
    );
```

```
226
227         frame2.calculateTotalScore();
228         Assert.assertEquals("Unexpected total score for frame 2", 4, frame2.getTotalScore()
    );
229     }
230
231     @Test
232     public void getTotalScore2() throws Exception {
233         Frame frame = new Frame();
234         frames.add(frame);
235         frames.get(2).setNext(frame);
236         frame.setPrev(frames.get(2));
237
238
239         Frame root = frames.get(0);
240         root.reset();
241
242         //[X, -], [ , ], [ , ], [ , , ]
243         root.setPinCount(10);
244         Assert.assertEquals("Unexpected Total Score result 1b1", Frame.Unset, frames.get(0)
    .getTotalScore());
245         Assert.assertEquals("Unexpected Total Score result 2b1", Frame.Unset, frames.get(1)
    .getTotalScore());
246         Assert.assertEquals("Unexpected Total Score result 3b1", Frame.Unset, frames.get(2)
    .getTotalScore());
247         Assert.assertEquals("Unexpected Total Score result 4b1", Frame.Unset, frames.get(3)
    .getTotalScore());
248
249         //[X,-], [X,-], [ , ], [ , , ]
250         root.setPinCount(10);
251         Assert.assertEquals("Unexpected Total Score result 1b2", Frame.Unset, frames.get(0)
    .getTotalScore());
252         Assert.assertEquals("Unexpected Total Score result 2b2", Frame.Unset, frames.get(1)
    .getTotalScore());
253         Assert.assertEquals("Unexpected Total Score result 3b2", Frame.Unset, frames.get(2)
    .getTotalScore());
254         Assert.assertEquals("Unexpected Total Score result 4b2", Frame.Unset, frames.get(3)
    .getTotalScore());
255
256         //[X,-], [X,-], [5, ], [ , , ]
257         root.setPinCount(5);
258         Assert.assertEquals("Unexpected Total Score result 1b3", 25, frames.get(0).
    getTotalScore());
259         Assert.assertEquals("Unexpected Total Score result 2b3", Frame.Unset, frames.get(1)
    .getTotalScore());
260         Assert.assertEquals("Unexpected Total Score result 3b3", Frame.Unset, frames.get(2)
    .getTotalScore());
261         Assert.assertEquals("Unexpected Total Score result 4b3", Frame.Unset, frames.get(3)
    .getTotalScore());
262
263         //[X,-], [X,-], [5,5], [ , , ]
264         root.setPinCount(5);
265         Assert.assertEquals("Unexpected Total Score result 1b4", 25, frames.get(0).
    getTotalScore());
266         Assert.assertEquals("Unexpected Total Score result 2b4", 45, frames.get(1).
    getTotalScore());
267         Assert.assertEquals("Unexpected Total Score result 3b4", Frame.Unset, frames.get(2)
    .getTotalScore());
268         Assert.assertEquals("Unexpected Total Score result 4b4", Frame.Unset, frames.get(3)
    .getTotalScore());
```

File - D:\Users\gjr8050\262refactoring\test\Main\Frame\FrameTest.java

```
269
270         //[X,-], [X,-], [5,5], [X, , ]
271         // 25, 20, 20
272         root.setPinCount(10);
273         Assert.assertEquals("Unexpected Total Score result 1b5", 25, frames.get(0).
    getTotalScore());
274         Assert.assertEquals("Unexpected Total Score result 2b5", 45, frames.get(1).
    getTotalScore());
275         Assert.assertEquals("Unexpected Total Score result 3b5", 65, frames.get(2).
    getTotalScore());
276         Assert.assertEquals("Unexpected Total Score result 4b5", Frame.Unset, frames.get(3)
    .getTotalScore());
277
278         //[X,-], [X,-], [5,5], [X,X, ]
279         root.setPinCount(10);
280         Assert.assertEquals("Unexpected Total Score result 1b6", 25, frames.get(0).
    getTotalScore());
281         Assert.assertEquals("Unexpected Total Score result 2b6", 45, frames.get(1).
    getTotalScore());
282         Assert.assertEquals("Unexpected Total Score result 3b6", 65, frames.get(2).
    getTotalScore());
283         Assert.assertEquals("Unexpected Total Score result 4b6", Frame.Unset, frames.get(3)
    .getTotalScore());
284
285         //[X,-], [X,-], [5,5], [X,X,5]
286         root.setPinCount(5);
287         Assert.assertEquals("Unexpected Total Score result 1b7", 25, frames.get(0).
    getTotalScore());
288         Assert.assertEquals("Unexpected Total Score result 2b7", 45, frames.get(1).
    getTotalScore());
289         Assert.assertEquals("Unexpected Total Score result 3b7", 65, frames.get(2).
    getTotalScore());
290         Assert.assertEquals("Unexpected Total Score result 4b7", 90, frames.get(3).
    getTotalScore());
291     }
292 }
```

File - D:\Users\gjr8050\262refactoring\test\Main\Frame\SpareFrameTest.java

```java
package Main.Frame;

import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;

import java.util.ArrayList;

import static org.junit.Assert.*;

/**
 * Created by gjr8050 on 5/3/2016.
 */
public class SpareFrameTest {

    private ArrayList<Frame> frames;

    @Before
    public void setUp() throws Exception {
        frames = new ArrayList<>();
        for(int i = 0; i < 3; i++){
            Frame frame = new Frame();
            frames.add(frame);
            if(i > 0){
                frames.get(i - 1).setNext(frame);
                frame.setPrev(frames.get(i - 1));
            }
        }
    }

    private void setScores(){
        Frame root = frames.get(0);
        frames.get(0).reset();
        root.setPinCount(1);
        root.setPinCount(1);

        root.setPinCount(5);
        root.setPinCount(5);

        root.setPinCount(10);
        root.setPinCount(5);
        root.setPinCount(5);
    }

    private void setScores2(){
        Frame root = frames.get(0);
        frames.get(0).reset();
        root.setPinCount(10);

        root.setPinCount(5);
        root.setPinCount(5);

        root.setPinCount(10);
        root.setPinCount(10);
        root.setPinCount(5);
    }

    @Test
    public void setPinCount() throws Exception {
        Frame frame1 = new Frame();
```

```
61          Frame frame2 = new Frame();
62
63          frame1.setNext(frame2);
64          frame2.setPrev(frame1);
65
66          FrameState.SPARE.setPinCount(frame1, 5);
67          assert !frame1.isBowled(Ball.ONE) : "Expected frame 1 BALL.ONE to not be bowled";
68          Assert.assertEquals("Expected frame 2 to have pin count", 5, frame2.getPinCount(
    Ball.ONE));
69
70          FrameState.SPARE.setPinCount(frame2, 6);
71          Assert.assertEquals("Expected frame 2 to have pin count on BALL.THREE", 6, frame2.
    getPinCount(Ball.THREE));
72      }
73
74      @Test
75      public void calculateFrameScore() throws Exception {
76          setScores();
77          //1 + 1 + 5 = 7
78          Assert.assertEquals("Unexpected score calculation 1", 7, FrameState.SPARE.
    calculateFrameScore(frames.get(0)));
79          //5 + 5 + (10) = 20
80          Assert.assertEquals("Unexpected score calculation 2", 20, FrameState.SPARE.
    calculateFrameScore(frames.get(1)));
81          //10 + 5 + 5 = 20
82          Assert.assertEquals("Unexpected score calculation 3", 20, FrameState.SPARE.
    calculateFrameScore(frames.get(2)));
83      }
84
85
86
87      @Test
88      public void getChainScore() throws Exception {
89          setScores();
90          Assert.assertEquals("Unexpected Chain Score result 1", 10, FrameState.SPARE.
    getChainScore(frames.get(1),FrameState.STRIKE));
91          Assert.assertEquals("Unexpected Chain Score result 2", 5, FrameState.SPARE.
    getChainScore(frames.get(1),FrameState.SPARE));
92          Assert.assertEquals("Unexpected Chain Score result 3", 0, FrameState.SPARE.
    getChainScore(frames.get(1),FrameState.BOWLED));
93      }
94  }
```

File - D:\Users\gjr8050\262refactoring\test\Main\Frame\BowledFrameTest.java

```java
package Main.Frame;

import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;

import java.util.ArrayList;

import static org.junit.Assert.*;

/**
 * Testing functionality of BowledFrame state handler
 * Created by gjr8050 on 5/3/2016.
 */
public class BowledFrameTest {

    private ArrayList<Frame> frames;

    @Before
    public void setUp() throws Exception {
        frames = new ArrayList<>();
        for(int i = 0; i < 3; i++){
            Frame frame = new Frame();
            frames.add(frame);
            if(i > 0){
                frames.get(i - 1).setNext(frame);
                frame.setPrev(frames.get(i - 1));
            }
        }
    }

    private void setScores(){
        Frame root = frames.get(0);
        //[1,1],[ , ],[ , , ]
        root.setPinCount(1);
        root.setPinCount(1);
        //[1,1],[10,-],[ , , ]
        root.setPinCount(10);
        //[1,1],[10,-],[5,5,5]
        root.setPinCount(5);
        root.setPinCount(5);
        root.setPinCount(5);
    }

    @Test
    public void setPinCount() throws Exception {
        Frame frame1 = new Frame();
        Frame frame2 = new Frame();

        frame1.setNext(frame2);
        frame2.setPrev(frame1);

        FrameState.BOWLED.setPinCount(frame1, 5);
        assert !frame1.isBowled(Ball.ONE) : "Expected frame 1 BALL.ONE to not be bowled";
        Assert.assertEquals("Expected frame 2 to have pin count", 5, frame2.getPinCount(Ball
.ONE));
    }

    @Test
    public void calculateFrameScore() throws Exception {
```

File - D:\Users\gjr8050\262refactoring\test\Main\Frame\BowledFrameTest.java

```
60          setScores();
61          Assert.assertEquals("Unexpected score calculation 1", 2, FrameState.BOWLED.
    calculateFrameScore(frames.get(0)));
62          Assert.assertEquals("Unexpected score calculation 2", 10, FrameState.BOWLED.
    calculateFrameScore(frames.get(1)));
63          Assert.assertEquals("Unexpected score calculation 3", 15, FrameState.BOWLED.
    calculateFrameScore(frames.get(2)));
64      }
65
66      @Test
67      public void getChainScore() throws Exception {
68          setScores();
69          Assert.assertEquals("Unexpected Chain Score result 1", Frame.Unset, FrameState.
    BOWLED.getChainScore(frames.get(1),FrameState.STRIKE));
70          Assert.assertEquals("Unexpected Chain Score result 2", 10, FrameState.BOWLED.
    getChainScore(frames.get(1),FrameState.SPARE));
71          Assert.assertEquals("Unexpected Chain Score result 3", 0, FrameState.BOWLED.
    getChainScore(frames.get(1),FrameState.BOWLED));
72      }
73  }
```

File - D:\Users\gjr8050\262refactoring\test\Main\Frame\StrikeFrameTest.java

```
1  package Main.Frame;
2
3  import org.junit.Assert;
4  import org.junit.Before;
5  import org.junit.Test;
6
7  import java.util.ArrayList;
8
9  import static org.junit.Assert.*;
10
11 /**
12  * Created by gjr8050 on 5/3/2016.
13  */
14 public class StrikeFrameTest {
15
16     private ArrayList<Frame> frames;
17
18     @Before
19     public void setUp() throws Exception {
20         frames = new ArrayList<>();
21         for(int i = 0; i < 3; i++){
22             Frame frame = new Frame();
23             frames.add(frame);
24             if(i > 0){
25                 frames.get(i - 1).setNext(frame);
26                 frame.setPrev(frames.get(i - 1));
27             }
28         }
29     }
30
31     private void setScores(){
32         Frame root = frames.get(0);
33         root.setPinCount(10);
34
35         root.setPinCount(10);
36
37         root.setPinCount(10);
38         root.setPinCount(10);
39         root.setPinCount(10);
40     }
41
42     @Test
43     public void setPinCount() throws Exception {
44         Frame frame1 = new Frame();
45         Frame frame2 = new Frame();
46
47         frame1.setNext(frame2);
48         frame2.setPrev(frame1);
49
50         FrameState.STRIKE.setPinCount(frame1, 5);
51         assert !frame1.isBowled(Ball.ONE) : "Expected frame 1 BALL.ONE to not be bowled";
52         Assert.assertEquals("Expected frame 2 to have pin count", 5, frame2.getPinCount(Ball
   .ONE));
53
54         FrameState.STRIKE.setPinCount(frame2, 6);
55         Assert.assertEquals("Expected frame 2 to have pin count on BALL.TWO", 6, frame2.
   getPinCount(Ball.TWO));
56
57         FrameState.STRIKE.setPinCount(frame2, 4);
58         Assert.assertEquals("Expected frame 2 to have pin count on BALL.THREE", 4, frame2.
```

```
58 getPinCount(Ball.THREE));
59     }
60 
61     @Test
62     public void calculateFrameScore() throws Exception {
63         setScores();
64         //10 + 10 + 10 = 30
65         Assert.assertEquals("Unexpected score calculation 1", 30, FrameState.STRIKE.
   calculateFrameScore(frames.get(0)));
66         //10 + 10 + (10) = 30
67         Assert.assertEquals("Unexpected score calculation 2", 30, FrameState.STRIKE.
   calculateFrameScore(frames.get(1)));
68         //10 + 10 + (10) = 30
69         Assert.assertEquals("Unexpected score calculation 3", 30, FrameState.STRIKE.
   calculateFrameScore(frames.get(2)));
70     }
71 
72 
73 
74     @Test
75     public void getChainScore() throws Exception {
76         setScores();
77     }
78 }
```