```java
//  Kevin Andrew Hance
//  March 8th, 2018
//  CPSC 224: Object Oriented Programming (Zhang)
//  HW Assignment #5: Yahtzee
//
//  This subclass models the upper and lower parts of the Yahtzee scoreboard. Allows
//  for scores to be saved and bonuses to be applied to the scoreboard. Also has implementation
//  to print out possible categories to score in, and the final scoreboard with bonuses applied.

public class Scoreboard {

    private int[] upperSb;
    private int[] lowerSb;
    Hand h;


    public Scoreboard(int maxDieVal, Hand newHand) {
        // aces-sixes (or more), plus one element for 35 pt bonus
        upperSb = new int[maxDieVal + 1];
        // 3kind, 4kind, FH, SmStr, LgStr, Yhtz, Chance
        lowerSb = new int[7];
        // ensure all score values are initially set to zero
        for (int j = 0; j < upperSb.length; j++) {
            upperSb[j] = 0;
        }
        for (int i = 0; i < lowerSb.length; i++) {
            lowerSb[i] = 0;
        }
        h = newHand;
    }

    // add together and return all scores on upper scoreboard
    public int sumUpperScore() {
        int sum = 0;
        for(int i  : upperSb) {
            sum += i;
        }
        return sum;
    }

    // add together and return all scores on lower scoreboard
    public int sumLowerScore() {
        int sum = 0;
        for(int i  : lowerSb) {
            sum += i;
        }
        return sum;
    }

    // returns boolean representing whether or not the user has already scored in a specific category
    // in upper scoreboard
    public boolean isScoreEmptyUpper(int index) {
        return upperSb[index] == 0;
    }
```

```
55
56      // returns boolean representing whether or not the user has already scored in a specific
    category
57      // in lower scoreboard
58      public boolean isScoreEmptyLower(int index) {
59          return lowerSb[index] == 0;
60      }
61
62      //apply score to ones row
63      public void acesScore() {
64          int currentCount = 0;
65          for (int diePosition = 0; diePosition < 5; diePosition++)
66          {
67              if (h.getDieVal(diePosition) == 1)
68                  currentCount++;
69          }
70          upperSb[0] = currentCount;
71      }
72
73      //apply score to twos row
74      public void twosScore() {
75          int currentCount = 0;
76          for (int diePosition = 0; diePosition < 5; diePosition++)
77          {
78              if (h.getDieVal(diePosition) == 2)
79                  currentCount++;
80          }
81          upperSb[1] = currentCount;
82      }
83
84      //apply score to threes row
85      public void threesScore() {
86          int currentCount = 0;
87          for (int diePosition = 0; diePosition < 5; diePosition++)
88          {
89              if (h.getDieVal(diePosition) == 3)
90                  currentCount++;
91          }
92          upperSb[2] = currentCount;
93      }
94
95      //apply score to fours row
96      public void foursScore() {
97          int currentCount = 0;
98          for (int diePosition = 0; diePosition < 5; diePosition++)
99          {
100             if (h.getDieVal(diePosition) == 4)
101                 currentCount++;
102         }
103         upperSb[3] = currentCount;
104     }
105
106     //apply score to fives row
107     public void fivesScore() {
108         int currentCount = 0;
109         for (int diePosition = 0; diePosition < 5; diePosition++)
110         {
```

```
111             if (h.getDieVal(diePosition) == 5)
112                 currentCount++;
113         }
114         upperSb[4] = currentCount;
115     }
116 
117     //apply score to sixes row
118     public void sixesScore() {
119         int currentCount = 0;
120         for (int diePosition = 0; diePosition < 5; diePosition++)
121         {
122             if (h.getDieVal(diePosition) == 6)
123                 currentCount++;
124         }
125         upperSb[5] = currentCount;
126     }
127     public void bigDieScore(int dieVal) {
128         int currentCount = 0;
129         for (int diePosition = 0; diePosition < 5; diePosition++)
130         {
131             if (h.getDieVal(diePosition) == dieVal)
132                 currentCount++;
133         }
134         upperSb[dieVal-1] = currentCount;
135     }
136 
137     //apply 35pt bonus for having over 63 points in upper scoreboard
138     public void applyUpperBonus() {
139         int sum = 0;
140         for (int i : upperSb)
141             sum += i;
142         if(sum >= 63)
143             upperSb[upperSb.length-1] = 35;
144     }
145 
146     //apply score to ones row
147     public void threeOfaKindScore() {
148         if (maxOfAKindFound(h) >= 3)
149             lowerSb[0] = totalAllDice(h);
150     }
151 
152     //apply score to ones row
153     public void fourOfaKindScore() {
154         if (maxOfAKindFound(h) >= 3)
155             lowerSb[1] = totalAllDice(h);
156     }
157 
158     //apply score to ones row
159     public void fullHouseScore() {
160         if (fullHouseFound(h))
161             lowerSb[2] = 25;
162     }
163 
164     //apply score to ones row
165     public void smStraightScore() {
166         if (maxStraightFound(h) >= 4)
167             lowerSb[3] = 30;
```

```
168     }
169
170     //apply score to large straight row
171     public void lgStrightScore() {
172         if (maxStraightFound(h) >= 5)
173             lowerSb[4] = 40;
174     }
175
176     //apply score to yahtzee row
177     public void yahtzeeScore() {
178         if (maxOfAKindFound(h) >= 5)
179             lowerSb[5] += 50;
180     }
181
182     //apply score to chance row
183     public void chanceScore() {
184         lowerSb[6] = totalAllDice(h);
185     }
186
187     //apply 50pt bonus for each second, third, fourth, etc. yahtzees
188     public void applyYahtzeeBonus() {
189         int bonus = (((lowerSb[5] / 50) - 1) * 50);
190         lowerSb[7] = bonus;
191     }
192
193     public void printScoreboard(int maxDieValue, Hand h)
194     {
195         //upper scorecard
196         //all die
197         for (int dieValue = 1; dieValue <= maxDieValue; dieValue++)
198         {
199             int currentCount = 0;
200             for (int diePosition = 0; diePosition < 5; diePosition++)
201             {
202                 if (h.getDieVal(diePosition) == dieValue)
203                     currentCount++;
204             }
205             System.out.print("Enter " + dieValue + " to score " + dieValue * currentCount + "
    on the ");
206             System.out.println(dieValue + " line");
207         }
208
209
210         //lower scorecard
211         //3 of a kind
212         if (maxOfAKindFound(h) >= 3)
213         {
214             System.out.print("Enter A to score " + totalAllDice(h) + " on the ");
215             System.out.print("3 of a Kind line\n");
216         }
217         else System.out.print("Enter A to score 0 on the 3 of a Kind line\n");
218
219         //4 of a kind
220         if (maxOfAKindFound(h) >= 4)
221         {
222             System.out.print("Enter B to score " + totalAllDice(h) + " on the ");
223             System.out.print("4 of a Kind line\n");
```

```
224         }
225         else System.out.print("Enter B to score 0 on the 4 of a Kind line\n");
226 
227         //full house
228         if (fullHouseFound(h))
229             System.out.print("Enter C to score 25 on the Full House line\n");
230         else
231             System.out.print("Enter C to score 0 on the Full House line\n");
232 
233         //small straight
234         if (maxStraightFound(h) >= 4)
235             System.out.print("Enter D to score 30 on the Small Straight line\n");
236         else
237             System.out.print("Enter D to score 0 on the Small Straight line\n");
238 
239         //large straight
240         if (maxStraightFound(h) >= 5)
241             System.out.print("Enter E to score 40 on the Large Straight line\n");
242         else
243             System.out.print("Enter E to score 0 on the Large Straight line\n");
244 
245         //yahtze
246         if (maxOfAKindFound(h) >= 5)
247             System.out.print("Enter F to score 50 on the Yahtzee line\n");
248         else
249             System.out.print("Enter F to score 0 on the Yahtzee line\n");
250 
251         //chance
252         System.out.print("Enter G to score " + totalAllDice(h) + " on the ");
253         System.out.print("Chance line\n");
254     }
255 
256     // prints out all scores, bonuses, subtotals and grand total
257     public void printFinalScore() {
258         System.out.println("Aces:\t\t" + upperSb[0]);
259         System.out.println("Twos:\t\t" + upperSb[1]);
260         System.out.println("Threes:\t\t" + upperSb[2]);
261         System.out.println("Fours:\t\t" + upperSb[3]);
262         System.out.println("Fives:\t\t" + upperSb[4]);
263         System.out.println("Sixes:\t\t" + upperSb[5]);
264         System.out.println("Upper Bonus:\t\t" + upperSb[h.getMaxDieValue()]);
265         System.out.println("\nUpper Total:\t\t" + sumUpperScore());
266 
267         System.out.println("3 of a kind:\t" + lowerSb[0]);
268         System.out.println("4 of a kind:\t" + lowerSb[1]);
269         System.out.println("Full House:\t" + lowerSb[2]);
270         System.out.println("Small Straight:\t" + lowerSb[3]);
271         System.out.println("Large Straight:\t" + lowerSb[4]);
272         System.out.println("Yahtzee:\t\t" + lowerSb[5]);
273         System.out.println("Chance:\t\t" + lowerSb[6]);
274         System.out.println("Yahtzee Bonus:\t" + lowerSb[7]);
275         System.out.println("\nLower Total:\t" + sumLowerScore());
276         System.out.println("\nGrand Total:\t" + (sumUpperScore() + sumLowerScore()));
277     }
278 
279     public static int maxOfAKindFound(Hand hand)
280     //this function returns the count of the die value occurring most in the hand
```

```
281     //but not the value itself
282     {
283         int maxCount = 0;
284         int currentCount ;
285         for (int dieValue = 1; dieValue <=6; dieValue++)
286         {
287             currentCount = 0;
288             for (int diePosition = 0; diePosition < 5; diePosition++)
289             {
290                 if (hand.getDieVal(diePosition) == dieValue)
291                     currentCount++;
292             }
293             if (currentCount > maxCount)
294                 maxCount = currentCount;
295         }
296         return maxCount;
297     }
298     public static int totalAllDice(Hand hand)
299     //this function returns the total value of all dice in a hand
300     {
301         int total = 0;
302         for (int diePosition = 0; diePosition < 5; diePosition++)
303         {
304             total += hand.getDieVal(diePosition);
305         }
306         return total;
307     }
308
309     public static int maxStraightFound(Hand hand)
310     //this function returns the length of the longest
311     //straight found in a hand
312     {
313         int maxLength = 1;
314         int curLength = 1;
315         for(int counter = 0; counter < 4; counter++)
316         {
317             if (hand.getDieVal(counter) + 1 == hand.getDieVal(counter + 1) ) //jump of 1
318                 curLength++;
319             else if (hand.getDieVal(counter) + 1 < hand.getDieVal(counter + 1)) //jump of >= 2
320                 curLength = 1;
321             if (curLength > maxLength)
322                 maxLength = curLength;
323         }
324         return maxLength;
325     }
326     public static boolean fullHouseFound(Hand hand)
327     //this function returns true if the hand is a full house
328     //or false if it does not
329     {
330         boolean foundFH = false;
331         boolean found3K = false;
332         boolean found2K = false;
333         int currentCount ;
334         for (int dieValue = 1; dieValue <=6; dieValue++)
335         {
336             currentCount = 0;
337             for (int diePosition = 0; diePosition < 5; diePosition++)
```

```
338             {
339                 if (hand.getDieVal(diePosition) == dieValue)
340                     currentCount++;
341             }
342             if (currentCount == 2)
343                 found2K = true;
344             if (currentCount == 3)
345                 found3K = true;
346         }
347         if (found2K && found3K)
348             foundFH = true;
349         return foundFH;
350     }
351 
352 }
353 
```