```
1 import java.util.ArrayList;
2 import java.util.Random;
3
4 /**
5 * Simulates a deck of 52 playing cards.
6 *
7 * @author Chris Hegang Kim
8 * @note I affirm that I have carried out the attached
   academic endeavors with full academic honesty,
9 * in accordance with the Union College Honor Code and the
   course syllabus.
10 */
11
12 public class Deck {
13
       private static final int NUMBER_OF_CARDS=52;
14
       private static final int NUMBER_OF_SUITS=4;
15
       private static final int CARDS_IN_SUIT=13;
16
       private final int START = 0;
17
       private final int FIRST_RANK = 2;
18
       private final int FIRST_CARD = 0;
19
       private final int EMPTY = 0;
20
21
       private ArrayList<Card> theCards;
22
       private boolean shuffled;
23
24
      /**
25
       * Constructs a new ordered deck of playing cards
26
       */
27
       public Deck()
28
       {
29
           theCards = new ArrayList<Card>(NUMBER_OF_CARDS);
30
31
           addAllCards();
32
33
           shuffled=false;
34
       }
35
36
      /**
37
38
       * Deals out next card in deck; returns null if no cards
   left
```

```
39
40
        * @return next card in deck or null if deck empty
41
        */
42
       public Card deal() {
           if (! isEmpty()) {
43
44
               if (shuffled) {
45
                   return dealRandom();
               }
46
47
48
               else {
49
                   return theCards.remove(FIRST_CARD);
50
               }
           }
51
52
53
           else {
54
               return null;
55
           }
       }
56
57
58
       /**
        * Deals out random card in deck
59
60
61
        * @return random card in deck
62
        */
63
       private Card dealRandom() {
64
           Random random = new Random();
           int randomNumber = random.nextInt(size());
65
66
           return theCards.remove(randomNumber);
67
       }
68
69
70
       /** determines if deck has any cards left in it
71
72
        * @return true if Deck empty; else false
73
        */
       public boolean isEmpty(){
74
           if (size() = EMPTY) {
75
76
               return true;
77
           }
78
79
           else {
```

```
80
                return false;
 81
            }
 82
        }
 83
        /**
 84
 85
         * Shuffles the cards
 86
         */
        public void shuffle() {shuffled = true;}
 87
 88
 89
        /** Returns number of undealt cards left in the deck
 90
         *
 91
         * @return number of undealt cards in the deck
 92
        public int size() {return theCards.size();}
 93
 94
 95
 96
         * Reset the deck by gathering up all dealt cards.
         * Postcondition: Deck contains all cards and is NOT
 97
    shuffled
 98
         */
 99
        public void gather() {
            theCards.clear();
100
            addAllCards();
101
102
103
            shuffled=false;
        }
104
105
106
107
         * Adds all cards into the deck
108
         */
        private void addAllCards() {
109
            for (int suit = START; suit < NUMBER_OF_SUITS; suit</pre>
110
    ++) {
111
                for (int rank = FIRST_RANK; rank < CARDS_IN_SUIT</pre>
     + 2; rank++) {
112
                    theCards.add(new Card(rank, suit));
113
                }
            }
114
115
        }
116
117
        /**
```

```
* DEBUGGING METHOD: prints out stats of the given list
118
    of cards, that is, what was dealt.
119
         * Prints the remaining number of cards of each suit and
     of each rank.
120
         *
121
         * @param cardList list of cards that are (were) in the
   deck
122
        * @hidden
123
         */
        public void printStats(ArrayList<Card> cardList)
124
125
        {
126
            int Hcount=0;
127
            int Dcount=0;
128
            int Scount=0;
129
            int Ccount=0;
130
            int[] ranks = new int[CARDS_IN_SUIT];
131
            int size=cardList.size();
132
            for (int i=0; i<size; i++)</pre>
133
            {
134
                int val = cardList.get(i).getRank();
                String suit = cardList.get(i).getSuit();
135
136
                if (suit.equals("clubs"))
137
                    Ccount++;
138
                else if (suit.equals("diamonds"))
139
                    Dcount++;
                else if (suit.equals("spades"))
140
141
                    Scount++:
142
                else if (suit.equals("hearts"))
143
                    Hcount++;
144
                ranks[val-2]++; // deck RANKS run from 2-14 so
   need to subtract 2
145
            }
146
            System.out.println("***PRINTING DECK STATS***");
            System.out.println("# clubs: " + Ccount);
147
            System.out.println("# diamonds: " + Dcount);
148
            System.out.println("# hearts: " + Hcount);
149
150
            System.out.println("# spades: " + Scount);
151
152
            System.out.print("Card:\t");
153
            for (int j = 2; j < Card.RANKS.length; j++) {</pre>
154
                System.out.print(Card.RANKS[j]+"\t");
```

```
File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim_Lab8/src/Deck.java
155
156
             System.out.println();
             System.out.print("Qty:\t");
157
             for (int j=0; j<ranks.length; j++) {</pre>
158
                  System.out.print(ranks[j] + "\t");
159
                  if (j>8) { // indices 9-12 are Jack thru Ace
160
                      System.out.print("\t");
161
                  }
162
163
             }
             System.out.println("\n");
164
         }
165
166
167
         /**
168
          * DEBUGGING METHOD: prints out stats of this Deck
     object
          * Prints the remaining number of cards of each suit and
169
      of each rank.
170
          *
171
          * @hidden
172
          */
173
         public void printStats() {
             printStats(theCards);
174
175
         }
176
177
178 }
179
```