```java
import java.util.List;
import java.util.ArrayList;

/**
 * The Deck class represents a shuffled deck of cards.
 * It provides several operations including
 *      initialize, shuffle, deal, and check if empty.
 */
public class Deck {

    /**
     * cards contains all the cards in the deck.
     */
    private List<Card> cards;

    /**
     * size is the number of not-yet-dealt cards.
     * Cards are dealt from the top (highest index) down.
     * The next card to be dealt is at size - 1.
     */
    private int size;

    /**
     * Creates a new Deck instance.
     * It pairs each element of ranks with each element of suits,
     * and produces one of the corresponding card.
     * @param ranks is an array containing all of the card ranks.
     * @param suits is an array containing all of the card suits.
     * @param values is an array containing all of the card point values.
     */
    public Deck(String[] ranks, String[] suits, int[] values) {
        cards = new ArrayList<Card>();
        /* *** TO BE IMPLEMENTED IN ACTIVITY 2 *** */
        size = 0;
        for(int o = 0; o < suits.length; o++)
            for(int i = 0; i < ranks.length; i++) {
                cards.add( new Card( ranks[i], suits[o], values[i] ) );
                size++;
            }
    }

    /**
     * Determines if this deck is empty (no undealt cards).
     * @return true if this deck is empty, false otherwise.
     */
    public boolean isEmpty() {
        /* *** TO BE IMPLEMENTED IN ACTIVITY 2 *** */
       return size() == 0;
    }

    /**
     * Accesses the number of undealt cards in this deck.
     * @return the number of undealt cards in this deck.
     */
    public int size() {
        /* *** TO BE IMPLEMENTED IN ACTIVITY 2 *** */
        return size;
    }

    /**
     * Randomly mix the given collection of cards
     * and reset the size to represent the entire deck.
     */
    public void shuffle() {
        /* *** TO BE IMPLEMENTED IN ACTIVITY 4 *** */
        for( int i = cards.size() - 1; i > 0; i-- ) {
```

```
67              int r = (int)( Math.random() * (i + 1) );
68              Card temp = cards.get(r);
69              cards.set( r, cards.get(i) );
70              cards.set( i, temp );
71          }
72      }
73
74      /**
75       * Deals a card from this deck.
76       * @return the card just dealt, or null if all the cards have been
77       *         previously dealt.
78       */
79      public Card deal() {
80          /* *** TO BE IMPLEMENTED IN ACTIVITY 2 *** */
81          if(size == 0)
82              return null;
83          size--;
84          return cards.get(size);
85      }
86
87      /**
88       * Generates and returns a string representation of this deck.
89       * @return a string representation of this deck.
90       */
91      @Override
92      public String toString() {
93          String rtn = "size = " + size + "\nUndealt cards: \n";
94
95          for (int k = size - 1; k >= 0; k--) {
96              rtn = rtn + cards.get(k);
97              if (k != 0) {
98                  rtn = rtn + ", ";
99              }
100             if ((size - k) % 2 == 0) {
101                 // Insert carriage returns so entire deck is visible on console.
102                 rtn = rtn + "\n";
103             }
104         }
105
106         rtn = rtn + "\nDealt cards: \n";
107         for (int k = cards.size() - 1; k >= size; k--) {
108             rtn = rtn + cards.get(k);
109             if (k != size) {
110                 rtn = rtn + ", ";
111             }
112             if ((k - cards.size()) % 2 == 0) {
113                 // Insert carriage returns so entire deck is visible on console.
114                 rtn = rtn + "\n";
115             }
116         }
117
118         rtn = rtn + "\n";
119         return rtn;
120     }
121 }
122
```