Activity 2: Initial Design of a Deck Class

Introduction:

Think about a deck of cards. How would you describe a deck of cards? When you play card games, what kinds of operations do these games require a deck to provide?

Exploration:

Now consider implementing a class to represent a deck of cards. Describe its instance variables and methods, and discuss your design with a classmate.

Read the partial implementation of the Deck class available in the **Activity2 Starter Code** folder. This file contains the instance variables, constructor header, and method headers for a Deck class general enough to be useful for a variety of card games. Discuss the Deck class with your classmates; in particular, make sure you understand the role of each of the parameters to the Deck constructor, and of each of the private instance variables in the Deck class.

Exercises:

- 1. Complete the implementation of the Deck class by coding each of the following:
 - Deck constructor This constructor receives three arrays as parameters. The arrays contain the ranks, suits, and point values for each card in the deck. The constructor creates an ArrayList, and then creates the specified cards and adds them to the list.

```
For example, if ranks = {"A", "B", "C"}, suits = {"Giraffes", "Lions"}, and values = \{2,1,6\}, the constructor would create the following cards:
```

```
["A", "Giraffes", 2], ["B", "Giraffes", 1], ["C", "Giraffes", 6],
["A", "Lions", 2], ["B", "Lions", 1], ["C", "Lions", 6]
```

and would add each of them to cards. The parameter size would then be set to the size of cards, which in this example is 6.

Finally, the constructor should shuffle the deck by calling the shuffle method. Note that you will not be implementing the shuffle method until Activity 4.

- isEmpty This method should return true when the size of the deck is 0; false otherwise.
- size This method returns the number of cards in the deck that are left to be dealt.

• deal — This method "deals" a card by removing a card from the deck and returning it, if there are any cards in the deck left to be dealt. It returns <code>null</code> if the deck is empty. There are several ways of accomplishing this task. Here are two possible algorithms:

Algorithm 1: Because the cards are being held in an ArrayList, it would be easy to simply call the List method that removes an object at a specified index, and return that object. Removing the object from the end of the list would be more efficient than removing it from the beginning of the list. Note that the use of this algorithm also requires a separate "discard" list to keep track of the dealt cards. This is necessary so that the dealt cards can be reshuffled and dealt again.

Algorithm 2: It would be more efficient to leave the cards in the list. Instead of removing the card, simply decrement the size instance variable and then return the card at size. In this algorithm, the size instance variable does double duty; it determines which card to "deal" and it also represents how many cards in the deck are left to be dealt. This is the algorithm that you should implement.

2. Once you have completed the Deck class, find DeckTester.java file in the Activity2 Starter Code folder. Add code in the main method to create three Deck objects and test each method for each Deck object.

Questions:

- 1. Explain in your own words the relationship between a deck and a card.
- 2. Consider the deck initialized with the statements below. How many cards does the deck contain?

```
String[] ranks = {"jack", "queen", "king"};
String[] suits = {"blue", "red"};
int[] pointValues = {11, 12, 13};
Deck d = new Deck(ranks, suits, pointValues);
```

3. The game of Twenty-One is played with a deck of 52 cards. Ranks run from ace (highest) down to 2 (lowest). Suits are spades, hearts, diamonds, and clubs as in many other games. A face card has point value 10; an ace has point value 11; point values for 2, ..., 10 are 2, ..., 10, respectively. Specify the contents of the ranks, suits, and pointValues arrays so that the statement

```
Deck d = new Deck(ranks, suits, pointValues);
```

initializes a deck for a Twenty-One game.

4. Does the order of elements of the ranks, suits, and pointValues arrays matter?