Name:
Date:

17.06 Elevens Lab Worksheet

Directions: Make note of your responses to the following questions as you work through the activities and exercise in the lesson.

Activity 1 Exercise Results

1. From step a, paste your `Card` class constructor below.

```
public Card(String cardRank, String cardSuit, int cardPointValue)
 {
   this.rank = cardRank;
   this.suit = cardSuit;
   this.pointValue = cardPointValue;
 }
```

From step c, paste your `matches` method below.

```
public boolean matches(Card otherCard)
 {
   if(otherCard.rank().equals(this.rank) &&
     otherCard.suit().equals(this.suit) &&
     otherCard.pointValue == this.pointValue)
   {
    return true;
   }
   return false;
 }
```

Paste the results of running the `CardTester.java` class below.

```
**** Tests Card 1: ace of hearts ****
rank: ace
suit: hearts
pointValue: 1
toString: ace of hearts (point value = 1)

**** Test matches ****
Matching: true
Not Matching: false
```

Activity 2 Exercise Results

1. From step a, paste your `Deck` class constructor below.

```
public Deck(String[] ranks, String[] suits, int[] values)
  {

    /* *** TO BE IMPLEMENTED IN ACTIVITY 2 *** */
    // initialize cards as a new ArrayList
    cards = new ArrayList<Card>();
    // loop through the parameter arrays to add new cards
    for(String rank : ranks)
    {
      for(String suit : suits)
      {
        for(int value : values)
        {
          cards.add(new Card(rank, suit, value));
```

```
        }
       }
     }
     // assign size of cards to the size instance variable
     size = cards.size();
     // shuffle cards
     shuffle();
   }
```

From step b, paste your `isEmpty` method below.

```
public boolean isEmpty()
  {
     if(cards.size() == 0){return true;}
     return false;
  }
```

From step d, paste your `deal` method below.

```
 public Card deal()
  {
     /* *** TO BE IMPLEMENTED IN ACTIVITY 2 *** */
     if(size == 0){return null;}
     else
     {
```

```
      size--;
      return cards.get(size);
    }
  }
```

Paste the results of running the `DeckTester.java` class below.

**** Original Deck Methods ****

toString:

size = 12

Undealt cards:

king of spade (point value = 10), king of spade (point value = 11),

king of spade (point value = 12), king of heart (point value = 10),

king of heart (point value = 11), king of heart (point value = 12),

queen of spade (point value = 10), queen of spade (point value = 11),

queen of spade (point value = 12), queen of heart (point value = 10),

queen of heart (point value = 11), queen of heart (point value = 12)

Dealt cards:

isEmpty: false

size: 12

**** Deal a Card ****

deal: king of spade (point value = 10)

**** Deck Methods After 1 Card Dealt ****

toString:

size = 11

Undealt cards:

king of spade (point value = 11), king of spade (point value = 12),

king of heart (point value = 10), king of heart (point value = 11),

king of heart (point value = 12), queen of spade (point value = 10),

queen of spade (point value = 11), queen of spade (point value = 12),

queen of heart (point value = 10), queen of heart (point value = 11),

queen of heart (point value = 12)

Dealt cards:

king of spade (point value = 10)

isEmpty: false

size: 11

**** Deal Remaining 5 Cards ****

deal: king of spade (point value = 11)

deal: king of spade (point value = 12)

deal: king of heart (point value = 10)

deal: king of heart (point value = 11)

deal: king of heart (point value = 12)

**** Deck Methods After All Cards Dealt ****

toString:

size = 6

Undealt cards:

queen of spade (point value = 10), queen of spade (point value = 11),

queen of spade (point value = 12), queen of heart (point value = 10),

queen of heart (point value = 11), queen of heart (point value = 12)

Dealt cards:

king of spade (point value = 10), king of spade (point value = 11),

king of spade (point value = 12), king of heart (point value = 10),

king of heart (point value = 11), king of heart (point value = 12)

isEmpty: false

size: 6

**** Deal a Card From Empty Deck ****

deal: queen of spade (point value = 10)

deal: queen of spade (point value = 11)

deal: queen of spade (point value = 12)

deal: queen of heart (point value = 10)

deal: queen of heart (point value = 11)

deal: queen of heart (point value = 12)

deal empty one: null

Activity 2 Questions:

1. Explain in your own words the relationship between a `deck` and a `card`.
   A deck is a class that creates a shuffled ArrayList of cards and manipulated the cards according to different methods
   A card is an individual card object described by its suit, value, and rank.

2. Consider the deck initialized with the statements below. How many cards does the deck contain? 18

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

3. Many card games are played with a deck of 52 cards. It is common for ranks to run from ace (highest) down to 2 (lowest). Suits are spades, hearts, diamonds, and clubs. A face card has point value 10; an ace has point value 11; point values for 2, …, 10 are 2, …, 10, respectively. Write the statements to declare and initialize the contents of the ranks, suits, and pointValues arrays so that the following statement initializes a deck as described.

```
String[] ranks = {“2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”,
“jack”, “queen”, “king”, “ace”};

String[] suits = {“spades”, “hearts”, “diamonds”, “clubs”};

int[] pointValues = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

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

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

Yes, as a rank must correspond with its point value