Name: **Trisha Ganesh**

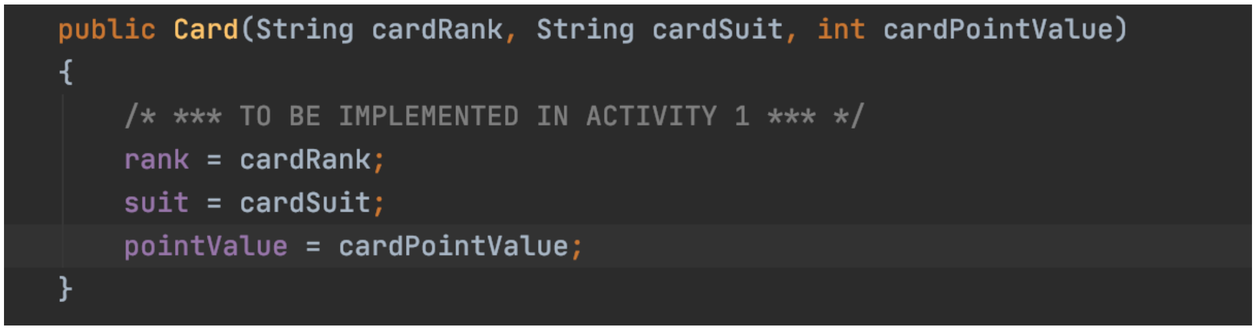
Date: **7/10/2021**

10.08 Elevens Lab Worksheet

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

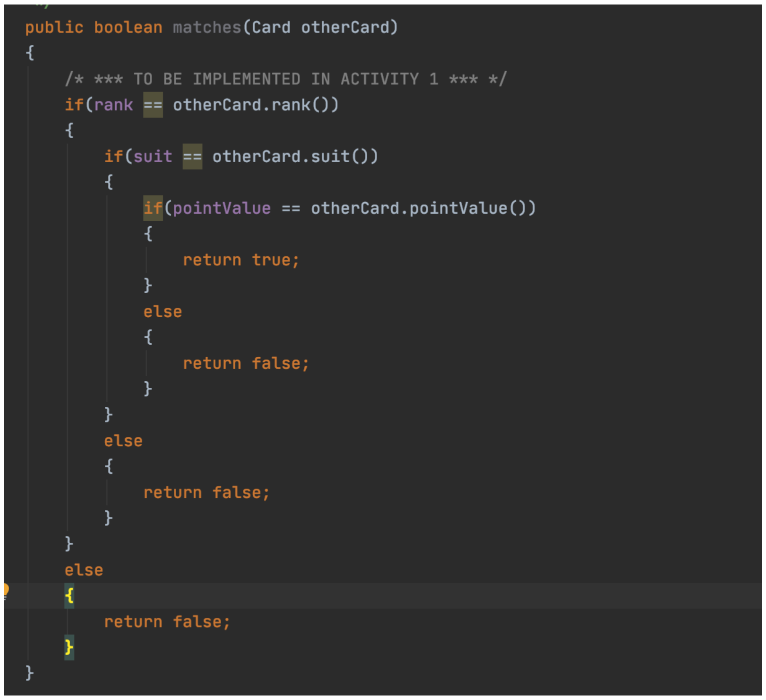
Activity 1 Exercise Results

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



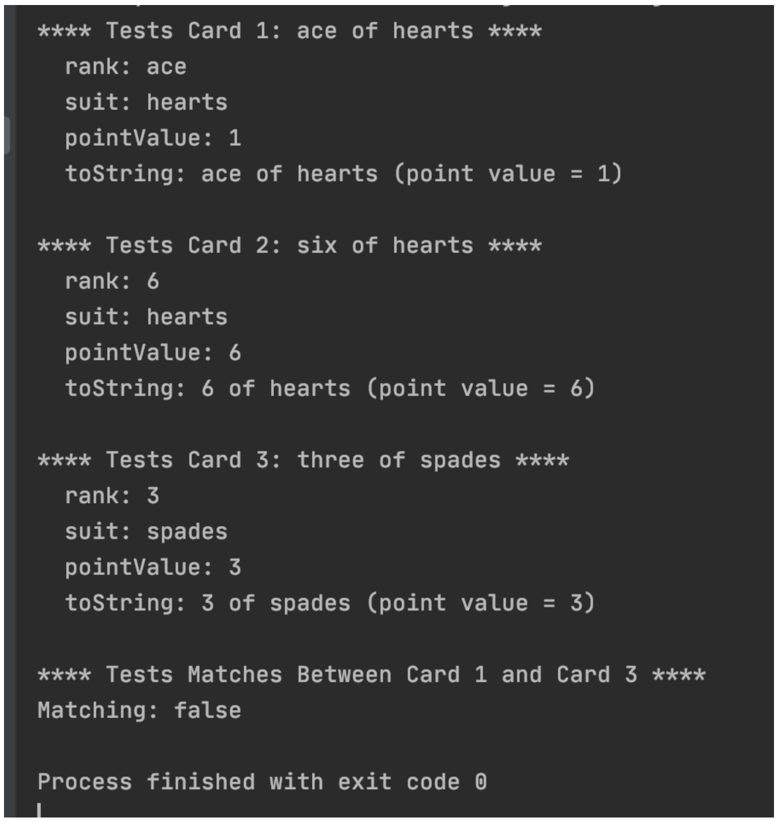
```
public Card(String cardRank, String cardSuit, int cardPointValue)
{
    /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
    rank = cardRank;
    suit = cardSuit;
    pointValue = cardPointValue;
}
```

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



```
public boolean matches(Card otherCard)
{
    /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
    if(rank == otherCard.rank())
    {
        if(suit == otherCard.suit())
        {
            if(pointValue == otherCard.pointValue())
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }
}
```

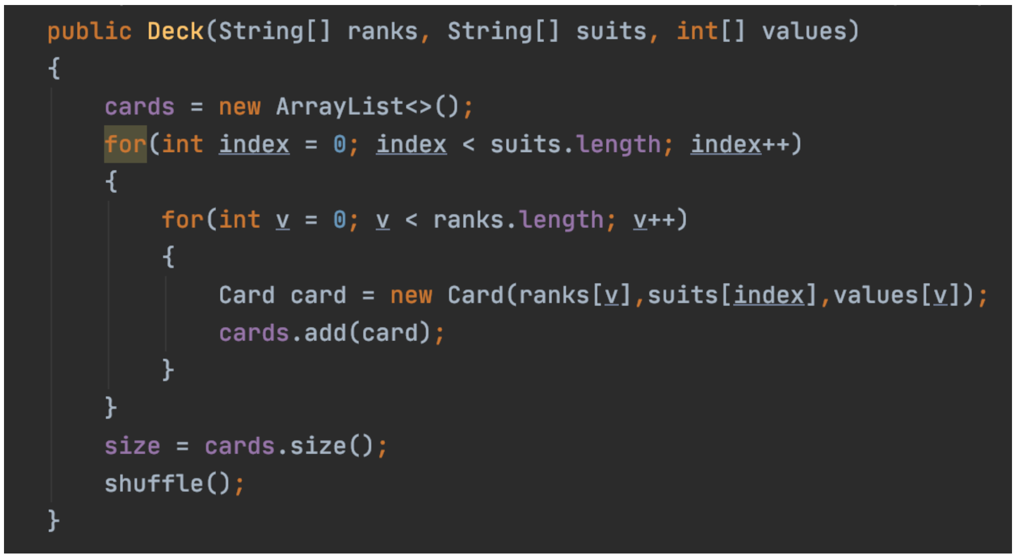
3. Paste the results of running the `CardTester.java` class below.



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

**** Tests Card 2: six of hearts ****
  rank: 6
  suit: hearts
  pointValue: 6
  toString: 6 of hearts (point value = 6)

**** Tests Card 3: three of spades ****
  rank: 3
  suit: spades
  pointValue: 3
  toString: 3 of spades (point value = 3)

**** Tests Matches Between Card 1 and Card 3 ****
Matching: false

Process finished with exit code 0
```

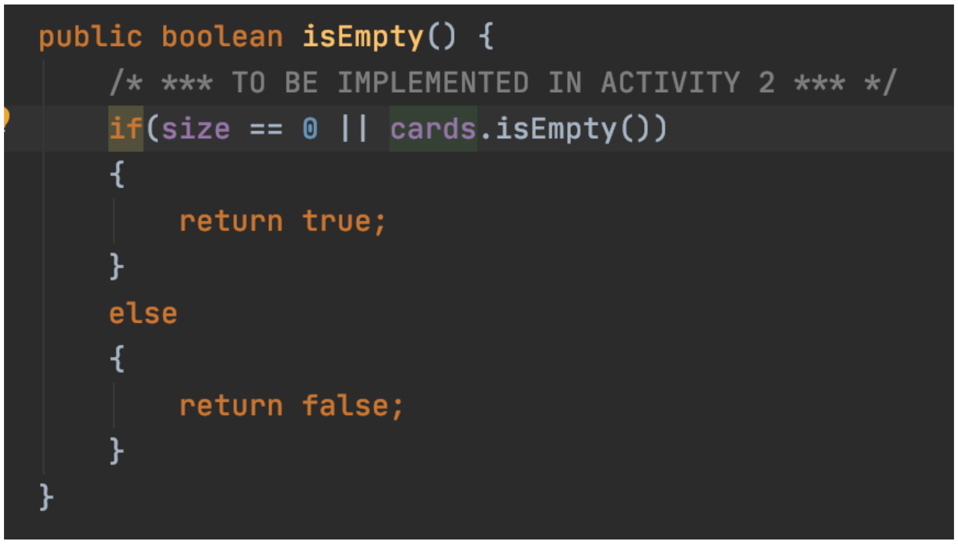
Activity 2 Exercise Results

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



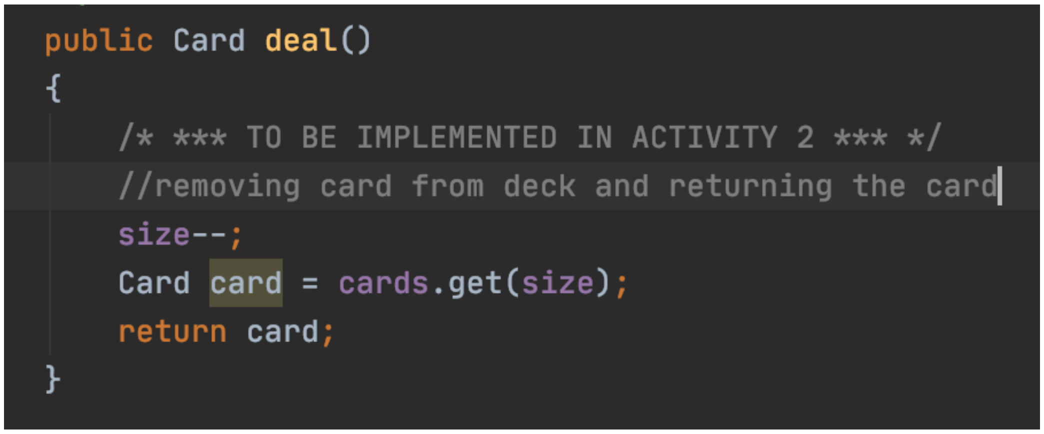
```java
public Deck(String[] ranks, String[] suits, int[] values)
{
    cards = new ArrayList<>();
    for(int index = 0; index < suits.length; index++)
    {
        for(int v = 0; v < ranks.length; v++)
        {
            Card card = new Card(ranks[v],suits[index],values[v]);
            cards.add(card);
        }
    }
    size = cards.size();
    shuffle();
}
```

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



```
public boolean isEmpty() {
    /* *** TO BE IMPLEMENTED IN ACTIVITY 2 *** */
    if(size == 0 || cards.isEmpty())
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

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



```
public Card deal()
{
    /* *** TO BE IMPLEMENTED IN ACTIVITY 2 *** */
    //removing card from deck and returning the card
    size--;
    Card card = cards.get(size);
    return card;
}
```

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

```
**** Original Deck Methods ****
  toString:
size = 22
Undealt cards:
ace of hearts (point value = 1), 2 of hearts (point value = 2),
3 of hearts (point value = 3), 4 of hearts (point value = 4),
5 of hearts (point value = 5), 6 of hearts (point value = 6),
7 of hearts (point value = 7), 8 of hearts (point value = 8),
9 of hearts (point value = 9), 10 of hearts (point value = 10),
J of hearts (point value = 11), ace of spades (point value = 1),
2 of spades (point value = 2), 3 of spades (point value = 3),
4 of spades (point value = 4), 5 of spades (point value = 5),
6 of spades (point value = 6), 7 of spades (point value = 7),
8 of spades (point value = 8), 9 of spades (point value = 9),
10 of spades (point value = 10), J of spades (point value = 11)

Dealt cards:


  isEmpty: false
  size: 22


**** Deal a Card ****
  deal: ace of hearts (point value = 1)
```

Activity 2 Questions:

1. **Explain in your own words the relationship between a `deck` and a `card`.**

   In short terms, a card is part of a deck as a deck is a collection or group of 52 cards.

1. **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);
```

   The deck contains 6 cards as there are all 2 suit values for each rank value. In this case, there are 3 rank values: jack, queen, and king; and there are 2 suit values: blue and red. Therefore, the product of the rank and suit values would be 3 x 2 which is 6 cards in total.

1. **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.**

```
Deck d = new Deck(ranks, suits, pointValues);
String[] ranks = {"ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", “queen”, “king”};
String[] suits = {"hearts", “diamonds”, “clubs”, “spades”};
int[] pointValues = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
```

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

The elements in the suit category can appear in all orders. However, the elements of the rank category can be changed, same for the elements in the pointValues category. As long as both those categories are updated with the same amount of values. For example, if there are 13 elements in the ranks category, there should be 13 in the pointValues category. For the situation above, there are 13 elements in ranks and pointValues and 4 in suits. Thus, stating that 13 x 4 = 52, which is the total number of cards in a deck. Overall, the sequence of elements in rank and pointValues should be in sync, preferably in ascending order.