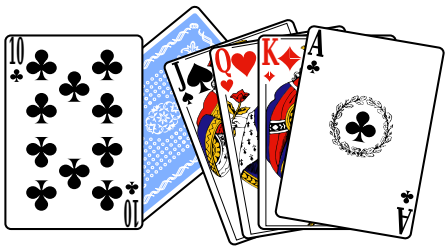
English name: _____

PBL: `class PlayingCard`

©2024 Chris Nielsen – *www.nielsenedu.com*

This assignment is intended to solidify your understanding of your understanding of classes and objects (instances of classes) that you learned in Unit 5.

A standard deck of French-suited playing cars consists of 52 cards. This is divided into four suits – spades, diamonds, clubs, hearts – represented, respectively, by the symbols: ♠ ♦ ♣ ♥.



Each suit consists of 13 ranks. The ranks are:
Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King.

Thus, if we pair the rank and suit, and represent the named cards by their first letter, the table below shows all the cards that should be in a standard deck of 52 cards:

A♠	2♠	3♠	4♠	5♠	6♠	7♠	8♠	9♠	10♠	J♠	Q♠	K♠
A♦	2♦	3♦	4♦	5♦	6♦	7♦	8♦	9♦	10♦	J♦	Q♦	K♦
A♣	2♣	3♣	4♣	5♣	6♣	7♣	8♣	9♣	10♣	J♣	Q♣	K♣
A♥	2♥	3♥	4♥	5♥	6♥	7♥	8♥	9♥	10♥	J♥	Q♥	K♥

This project starts with two classes, a `PlayingCard` class that represents individual playing cards, and a `Deck` class that represents a group of cards. A partial declaration of the `PlayingCard` class is given in the box below.

```
1  public class PlayingCard {
2
3     // Characters for the different suits
4     public static final String spade   = "♠";
5     public static final String diamond = "♦";
6     public static final String club    = "♣";
7     public static final String heart   = "♥";
8
9     // Array containing all the suits
10    public static final String[] suits =
11       { spade, diamond, club, heart };
12
13    // Array containing all the ranks
14    public static final String[] ranks = {
15       "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"
16    };
17
18    // Instance Fields
19    private final String suit;
20    private final String rank;
21
22    public PlayingCard(String suit, String rank) {
23       this.suit = suit;
24       this.rank = rank;
25    }
26
27    public String toString() {
28       return this.rank + this.suit;
29    }
30
31    // There may be other methods that are not shown
32 }
```

The `PlayingCard` class, above, defines some class fields (`static` fields). As these fields are modified with the keyword "`static`", they can be accessed through the class name, for example `PlayingCard.diamond` is equal to the string `"♦"`. You will need to use both the static field `suits` and the static field `ranks` to complete the constructor and receive full marks.

PBL: `class PlayingCard`

©2024 Chris Nielsen – *www.nielsenedu.com*

A partial declaration of the `Deck` class is given in the box below.

```
public class Deck {

   ArrayList<PlayingCard> cards;

   public Deck() {
      // You will implement this constructor in part (a)
   }

   @Override
   public String toString() {
      // You will implement this method in part (b)
   }

   public PlayingCard draw() {
      // You will implement this method in part (c)
   }

   public PlayingCard drawRandom() {
      // You will implement this method in part (d)
   }

   public void shuffle() {
      // You will implement this method in part (e)
   }

   // There may be other methods that are not shown

}
```

1. Complete the implementation of the `Deck` class according to the following specifications. It is highly recommended that you write test code to test each individual method you write, and **thoroughly test each method** before you go on to write the next method. Example test code and the expected for part (1) is given at the end of part (1).

   a) **Implement the zero-parameter constructor for class `Deck` according to the following specifications.**

   The constructor must use the field named `ranks` and the field named `suits` from the `PlayingCard` class – as lists of all the possible ranks and suits – to create a standard deck of cards that includes one card of each of all the 52 combinations of rank and suit that make up a standard set of playing cards. (Those combinations that are in the table on page 1).

   b) **Implement the method `toString` according to the following specifications.**

   For each `PlayingCard` in the `Deck`, call the card's `toString` method to generate a combined string containing all the cards from the `Deck`, in order, with a space between each card, and a newline character after every 13 cards, and a newline at the end of the `Deck`.

   c) **Implement the method `draw` according to the following specifications.**

   The method `draw` shall return the last `PlayingCard` from the cards in the `Deck`, and remove that card from the `Deck`.

   Consider why this method specifies to remove the last card from the Deck rather than the first. *Hint*: it has to do with efficiency when the underlying implementation uses an array (or `ArrayList`) to store the values.

   If this method is repeatedly called, the number of cards in the `Deck` will eventually reach zero. If there is an attempt to remove a card from an empty `Deck`, the method should return `null`.

d) **Implement the method `drawRandom` according to the following specifications.**

The method `drawRandom` shall return a random `PlayingCard` from the cards in the `Deck`, and remove that card from the `Deck`.

For example, if the `Deck` is a newly created deck with 52 cards, and the random card turns out to be the *Queen of Hearts* ("Q♥"), then after the method has completed, the `Deck` should contain 51 cards – all the cards of the standard deck except the *Queen of Hearts*.

If this method is repeatedly called, the number of cards in the `Deck` will eventually reach zero. If there is an attempt to remove a card from an empty `Deck`, the method should return `null`.

e) **Implement the method `shuffle` according to the following specifications.**

The `shuffle` method rearranges the cards in the `Deck` such that their order is randomized. This method must call the `drawRandom` method from part (b) to receive full marks. You may assume that the `drawRandom` method works correctly as specified above, even if the code you wrote may not work to specifications.

Hint: One way to implement this is as follows:

- Create a new `ArrayList` of `PlayingCard`.
- For each card you take from the `Deck` using `drawRandom`, you add it to the new `ArrayList`.
- Once the `Deck` is empty, have the `Deck` use the new `ArrayList` as its list of cards.

Write the completed code in the box immediately below.

Example Test Code for Class `Deck` and Class `PlayingCard`

```
public class TestDeck {
    public static void main(String[] args) {
        Deck deck = new Deck();
        System.out.println("New Deck:\n" + deck);
        System.out.println("Drawing random card: "
                           + deck.drawRandom());
        System.out.println("Remaining Deck:\n" + deck);
        System.out.print("\nDrawing more random cards:");
        for(int i = 0; i < 5; i++) {
            System.out.print(" " + deck.drawRandom());
        }
        System.out.println("\nRemaining Deck:\n" + deck);
        deck.shuffle();
        System.out.println("\nShuffled Deck:\n" + deck);
        System.out.println("\nDrawing remainder of deck:");
        PlayingCard p = deck.draw();
        while(p != null) {
            System.out.print(" " + p);
            p = deck.draw();
        }
        System.out.println();
        System.out.println("\nRemaining Deck:\n" + deck);
        System.out.println("Attempt to draw more cards: " +
                           deck.draw());
    }
}
```

©2024 Chris Nielsen – *www.nielsenedu.com*

Output of Example Test Code for Class `Deck` and Class `PlayingCard`

```
New Deck:
A♠ 2♠ 3♠ 4♠ 5♠ 6♠ 7♠ 8♠ 9♠ 10♠ J♠ Q♠ K♠
A♦ 2♦ 3♦ 4♦ 5♦ 6♦ 7♦ 8♦ 9♦ 10♦ J♦ Q♦ K♦
A♣ 2♣ 3♣ 4♣ 5♣ 6♣ 7♣ 8♣ 9♣ 10♣ J♣ Q♣ K♣
A♥ 2♥ 3♥ 4♥ 5♥ 6♥ 7♥ 8♥ 9♥ 10♥ J♥ Q♥ K♥

Drawing random card: 6♦
Remaining Deck:
A♠ 2♠ 3♠ 4♠ 5♠ 6♠ 7♠ 8♠ 9♠ 10♠ J♠ Q♠ K♠
A♦ 2♦ 3♦ 4♦ 5♦ 7♦ 8♦ 9♦ 10♦ J♦ Q♦ K♦ A♣
2♣ 3♣ 4♣ 5♣ 6♣ 7♣ 8♣ 9♣ 10♣ J♣ Q♣ K♣ A♥
2♥ 3♥ 4♥ 5♥ 6♥ 7♥ 8♥ 9♥ 10♥ J♥ Q♥ K♥

Drawing more random cards: 5♣ K♦ 4♦ 8♣ K♠
Remaining Deck:
A♠ 2♠ 3♠ 4♠ 5♠ 6♠ 7♠ 8♠ 9♠ 10♠ J♠ Q♠ A♦
2♦ 3♦ 5♦ 7♦ 8♦ 9♦ 10♦ J♦ Q♦ A♣ 2♣ 3♣ 4♣
6♣ 7♣ 9♣ 10♣ J♣ Q♣ K♣ A♥ 2♥ 3♥ 4♥ 5♥ 6♥
7♥ 8♥ 9♥ 10♥ J♥ Q♥ K♥

Shuffled Deck:
J♣ 10♦ J♦ 8♥ 7♠ Q♠ 4♣ A♣ 3♣ 10♣ 5♦ 5♠ K♣
6♣ 2♦ 4♠ A♠ A♦ J♠ 2♠ 9♠ 2♣ 3♠ 7♥ 3♥ 2♥
9♣ 8♦ 10♠ 9♦ 3♦ 5♥ 9♥ Q♥ Q♣ 6♥ 6♠ A♥ 4♥
7♣ K♥ 10♥ Q♦ J♥ 7♦ 8♠

Drawing remainder of deck:
 8♠ 7♦ J♥ Q♦ 10♥ K♥ 7♣ 4♥ A♥ 6♠ 6♥ Q♣ Q♥ 9♥ 5♥ 3♦ 9♦ 10♠ 8♦ 9♣ 2♥ 3♥ 7♥
3♠ 2♣ 9♠ 2♠ J♠ A♦ A♠ 4♠ 2♦ 6♣ K♣ 5♠ 5♦ 10♣ 3♣ A♣ 4♣ Q♠ 7♠ 8♥ J♦ 10♦ J♣

Remaining Deck:

Attempt to draw more cards: null
```