CardDeck.docx

Introduction

Well, you created the Card class, guess what's next? You guessed it, the Deck class.

Hint: Start by importing the CardDeck.zip project. The Card class is given to you by your instructor. No need to modify it.

Deck Class

What does a real-world deck look like? Something like a stack of cards. Let's make things simple by creating a deck of 52 cards with no jokers. But we are getting a bit ahead of ourselves. Let's talk about class attributes and methods.

Attributes

We will need to represent 52 cards and treat each card uniformly. Hey, that sounds like an array.

```
private Card [] cards; // must allocate space for 52 cards and fill each
                       //element with a Card object
```

The class will need another data member to keep track of which card to deal next, let's call it the `topCard` and it's just an index into the cards array.

```
private int topCard;   // range: 0 .. 51
```

Constructor

The Deck class will need a constructor. The constructor should not take any parameters. if it did, what parameters could you pass? The 52 individual cards? No, let's make the constructor create the card objects by individual suit (SPADE, HEART, DIAMOND, CLUB). Think of how the cards in a brand-new box of playing cards.

Hint: You will need a loop to create the cards of a suit. Perhaps the loop should be within a helper method such as:

```
void createSuit(Suit s)
```

could help. Be careful not to "overlay" the cards of one suit with those of another suit!

Methods

Well what should our Deck class do? How about shuffling its cards and dealing out some cards?

Let's think about the `deal()` method. It is local to the class so it can directly reference the `cards` data member. What does it need to know? Answer, how many cards to deal which we will call count. And what should it return to the caller? An array of count cards. Here is the signature of `deal()`.

```
public Card[] deal(int count)
```

Now deal must avoid dealing the same cards as it did to previous callers (players). It can do this by using the topCard variable to keep track of the "top card" index into the cards array. Here is pseudo-code for deal() method.

```
-- deal count cards
if topCard + count > 51 then exception
allocate space for local array of count cards called deal
for i from 0 to count-1 do
     deal[i] ← cards[topCard + i]
topCard ← topCard + count
return deal
```

Hint: You must allocate a local Card array of count cards at the beginning of the deal() method. The card references are then transferred from the cards array to the deal array which is the return value.

Hint: Stop and uncomment the first batch of lines in the MainClass to test if the constructor and the deal() method are working properly. If the suits of the deck are not clearly printed, there is probably an error within the constructor.

Below is the signature of the shuffle method.

```
public void shuffle()
```

The shuffle method takes no parameters and returns no value (void). It only has the side-effect of randomizing the elements in the cards array.

Here is pseudo code for a shuffle algorithm.

```
-- To shuffle an array a of n elements (indices 0..n-1):
for i from 0 to n−1 do
     j ← random integer such that 0 ≤ j < n
     swap a[i] and a[j]
```

You will want to use this technique to generate a random integer in the proper range.

```
j = (int) (Math.random() * 52);
```

Hint: Remember that to swap two elements in an array (e.g. `a[i]` and `a[j]` you will need a temporary variable to hold the value of the first element that is assigned to.

Hint: Stop and uncomment the second batch of lines in the MainClass to test if the shuffle() method is working properly. If you are a real nerd programmer, you will check that you haven't "lost" a card in the shuffle process.

For this class there is no need for getters and setters. It doesn't make sense to compare two decks (how would you do it?). It makes some sense to have a `toString()` method but that method would have to return a very long string indeed. So, let's not do that.

Hand Class

Then Hand class is a container for a subset of the cards in the deck. It requires a constructor and a couple of methods. Its instance variable is simply a reference to the cards that have been dealt for the hand:

```
private Card[] cards;
```

Constructor

The constructor Is simple. It is passed an array parameter that is a reference to the cards that have been dealt for the hand. All it has to do is save that reference.

Hint: This is an assignment from the parameter to the instance variable. Be sure to reference `this` if the names are the same.

Methods

The Hand class needs three simple methods:

- `int getCount()`
- `Card getCard()`
- `void sort()`

See the comments within the starter code.

Hint: When you have finished implementing these methods you should uncomment the third batch of lines in the MainClass. Look closely at what these lines do.

MainClass

Examine the code in these methods: `main()`, `show()`, and `compareHands()`. If you understand this code, then you understand object-oriented programming.

When the program is complete the output may look like this:

```
A deck:
2:SPADE 3:SPADE 4:SPADE 5:SPADE 6:SPADE 7:SPADE 8:SPADE 9:SPADE
10:SPADE J:SPADE Q:SPADE K:SPADE A:SPADE
2:HEART 3:HEART 4:HEART 5:HEART 6:HEART 7:HEART 8:HEART 9:HEART
10:HEART J:HEART Q:HEART K:HEART A:HEART
2:DIAMOND 3:DIAMOND 4:DIAMOND 5:DIAMOND 6:DIAMOND 7:DIAMOND 8:DIAMOND
9:DIAMOND 10:DIAMOND J:DIAMOND Q:DIAMOND K:DIAMOND A:DIAMOND
```

```
2:CLUB 3:CLUB 4:CLUB 5:CLUB 6:CLUB 7:CLUB 8:CLUB 9:CLUB 10:CLUB J:CLUB
Q:CLUB K:CLUB A:CLUB
A shuffled deck:
Q:CLUB J:SPADE 8:HEART A:SPADE 4:CLUB 8:CLUB 7:CLUB 7:DIAMOND
A:DIAMOND 4:HEART Q:HEART 6:DIAMOND 2:HEART
8:DIAMOND 3:SPADE 6:HEART 2:DIAMOND 5:SPADE A:CLUB K:CLUB 2:CLUB
K:SPADE 9:DIAMOND 2:SPADE 6:SPADE 10:HEART
J:DIAMOND J:CLUB Q:DIAMOND 4:SPADE K:HEART 3:DIAMOND 9:HEART 8:SPADE
Q:SPADE 10:DIAMOND 3:HEART 7:SPADE 3:CLUB
10:SPADE 10:CLUB 5:DIAMOND A:HEART K:DIAMOND 5:CLUB 4:DIAMOND 9:CLUB
6:CLUB 5:HEART 9:SPADE J:HEART 7:HEART
Comparing hands:
5:HEART = 5:SPADE
9:HEART > 5:CLUB
10:HEART > 6:SPADE
10:DIAMOND > 8:CLUB
K:DIAMOND > J:CLUB
```

Assignment

Get your Deck and Hand classes working so that the test code in `main()` produces the desired results. Submit both your Deck.java and your Hand.java files. In a comment write your reflections on "what I learned" when doing this assignment.