APCS-A Java Card & Deck Coding Assignment

Before starting to write code, you should carefully and thoroughly read all pages of these instructions.

You are to create two Java files to represent a **single playing card** and a **deck of playing cards** (as described below) that could be used as part of card games and other card-related programs. All of your work should be completed within your CodeHS.com account, and you must always do your own work. That is, all of the code that you submit for this assignment must be written entirely and only by you and not obtained from, or modified by, other sources.

A standard deck of 52 playing cards is divided into four suits (clubs, diamonds, hearts, spades), each consisting of 13 ranked cards. The diamonds and hearts suits are red, while the clubs and spades are black. Each suit contains three "face" cards (a jack, queen, and king), along with ten "pip" cards that have ranks numbered from one to ten (with the "one" card being renamed the "ace"). For purposes of this assignment, the ace card has a rank of 1, the jack has a rank of 11, the queen's rank is 12, and the king's rank is 13.

Write a non-static single-file class named "Card", which represents one playing card. A playing card consists of a rank and a suit:

`public Card()` ← Required Constructor #1
This constructor selects a random rank and suit to represent a single playing card that can appear in a standard 52-card deck.

`public Card(int rank, String suit)` ← Required Constructor #2
This constructor takes exactly two arguments: an integer and a string (in that order). The integer should be in the range of 1 to 13, representing the rank of the playing card to be constructed, and the string should be "Clubs", "Diamonds", "Hearts", or "Spades" (with no restrictions on how the letters are cased), representing the suit. If the rank is out of range, or if the suit is invalid, then the "bad" value(s) should be chosen randomly, but the "good" value (if there is one) should still be used.

In addition to the two constructors, for the Card class you should create the following public instance (non-static) methods:

`public String getCard()` ← Required Method #1
This method returns the rank and suit of the constructed Card object in the format "[Rank] of [Suit]" (e.g., "Two of Diamonds" or "Queen of Hearts" or "Ace of Spades" or "Ten of Clubs"). Each returned string consists of exactly three words and two spaces, and it does not contain any digits. All of the characters in the string are lowercase letters, except for the first character of the rank and the first character of the suit, which are always uppercased.

`public int getRank()` ← Required Method #2
This method returns a number from 1 through 13 representing the rank of the constructed Card object.

`public String getSuit()` ← Required Method #3
This method returns the suit of the Card object ("Clubs", "Diamonds", "Hearts", or "Spades", with only the first letter capitalized).

`public String getColor()` ← Required Method #4
This method returns the color of the constructed Card object ("Black" or "Red", with only the first letter capitalized).

`public String toString()` ← Required Method #5
This method overrides java.lang.Object.toString to return a string containing the suit and rank of the constructed playing card. This is what is shown whenever a Card object is displayed (via a 'System.out.print' or 'System.out.println' statement). See the sample program output for exactly how this string should be formatted (what this method should return, including spacing).

Your Card class should contain exactly two constructors and five public methods. Since an object of type "Card" represents just one playing card, for each instance of the Card class, multiple calls to any of the public instance methods of the class should always return the same value. To represent different playing cards, multiple instances of the Card class must be constructed.

Along with the Card class, you should write another non-static single-file class named "Deck", which represents one standard deck of (up to) 52 unique playing cards (Card objects). This class should contain exactly one constructor, which takes no arguments. Whenever a Deck object is created, it must initially contain exactly 52 unique Card objects. How much code (if any) the constructor contains will depend on how you choose to represent the deck of cards and/or the drawn card list (read on). In addition to the lone constructor, your Deck class should contain the following public instance (non-static) methods:

`public Card drawCard()` ← Required Method #1
This method draws and returns a single Card object with a random rank and suit matching the rank and suit of a playing card that is currently in the deck. If no cards are in the deck, then 'null' is returned.

`public Card[] drawCards(int numCardsToDraw, boolean shouldSort)` ← Required Method #2
This method returns an array (not an ArrayList) of 'numCardsToDraw' Card objects, each with a random rank and suit that matches the rank and suit of a playing card that is currently in the deck. The length of the array equals the number of drawn cards. The order of the Card objects in the array is random, unless 'shouldSort' is 'true', in which case the Card objects are sorted in ascending alphabetical order by suit, with each suit sorted in ascending numerical order by rank. If there are not at least 'numCardsToDraw' cards in the deck, or if 'numCardsToDraw' is zero or negative, then a zero-length array is returned.

`public Card[] drawCards(int numCardsToDraw, String cardInfo, boolean shouldSort)` ← Required Method #3
This method functions in the same manner as the above 'drawCards' method, with the exception that it accepts an additional argument, a string that should be "Clubs", "Diamonds", "Hearts", "Spades", "Black", or "Red" (with no restrictions on how the letters are cased). The returned array contains only Card objects with randomly-selected suits/colors that match 'cardInfo'. If not enough 'cardInfo' cards are in the deck, or if 'cardInfo' is not a valid suit or color, then a zero-length array is returned.

`public boolean returnCardToDeck(Card cardToReturn)` ← Required Method #4
This method attempts to place the playing card matching the rank and suit of the supplied Card object back into the deck. If successful, then 'true' is returned. If the card is already in the deck, then 'false' is returned.

`public boolean isCardInDeck(Card cardToCheck)` ← Required Method #5
This method returns 'true' if the playing card matching the rank and suit of the supplied Card object is currently in the deck, or 'false' if it is not.

`public int getNumCardsInDeck()` ← Required Method #6
This method returns the number of playing cards currently in the deck.

`public void resetDeck()` ← Required Method #7
This method "resets" the deck so that it contains all 52 unique playing cards.

`public String toString()` ← Required Method #8
This method overrides java.lang.Object.toString to return a string containing a list (sorted by suit, then rank) of every playing card currently in the deck. This is what is shown whenever a Deck object is displayed (via a 'System.out.print' or 'System.out.println' statement). See the sample program output for exactly how this string should be formatted (what this method should return).

An object of type "Deck" represents one standard deck of 52 playing cards. So, in the same way that additional instances of the Card class must be created in order to represent more than one playing card, new instances of the Deck class must be made in order to represent multiple decks of cards. However, unlike a Card object, which knows nothing about other playing cards, a Deck object must always be able to determine which cards are currently in the deck and which cards are not in the deck.

For the three public methods of the Deck class used to draw cards, make sure every drawn card is currently in the deck. That is, the rank/suit combination of every newly-drawn card must match a card currently in the deck. After a card is drawn, the card is no longer in the deck and cannot be drawn again (unless the card has been returned to the deck). Within the Deck class you may represent the deck of cards and the drawn card list in any manner of your choosing (including not even having two separate lists).

When creating new Card objects from within the Deck class, it is up to you which of the two Card class constructors to use. You may use either one, or both. How you decide to store cards in the deck and/or the drawn card list, as well as your approach to drawing cards from the deck, may influence your decision regarding which constructor to use and when to construct new cards.

The signatures of your public methods in both the Card Class and the Deck class must exactly match the method signatures shown above, including the parameter names and return types. The keyword 'static' and a 'main' method should not appear anywhere in either of the two classes. None of the constructors or the regular methods in either class should display anything.

For both the Card class and the Deck class, in addition to the public instance methods described above, you are welcome to write other "helper" methods. If you do, make sure they are all private and non-static. You may also create private instance variables to use throughout the two classes. And, even though the "drawCards" methods of the Deck class must each return an array, you may use private ArrayLists internally within the Deck class (as well as within the Card class) if you wish.

You may use global variables in both of your classes. Be sure to comment your code sufficiently, indent and space your code properly, and use meaningful identifier names. The design and layout of your code will be factored into your assignment score.

You will almost certainly want to create a calling class to test your work as you write the two required classes. However, the only files that will be examined and graded are "Card.java" and "Deck.java".

Sample Calling Class (the red letters are provided to help you quickly match the code with the output)

```
public static void main(String[] args)
{
A →    System.out.println("Create two decks of playing cards.");
       Deck deck1 = new Deck(), deck2 = new Deck();
       System.out.println("Number of cards in Deck #1:  " + deck1.getNumCardsInDeck());

B →    System.out.println("\nCreate a random Card object.");
       Card card1 = new Card();
       System.out.println("Can the new card be added to Deck #1?  " + deck1.returnCardToDeck(card1));

C →    System.out.println("\nDraw a random card from Deck #1 using the 'drawCard' method.");
       Card card2 = deck1.drawCard();
       System.out.println("Show the random card using the 'toString' method:  " + card2);
       System.out.println("Show the random card using the 'getCard' method:  " + card2.getCard());

D →    System.out.println("\nDeck #1: Draw 9 random cards. Receive in RANDOM order. Show using 'getCard':");
       Card[] hand1 = deck1.drawCards(9, false);
       for (int i = 0; i < hand1.length; i++)
           System.out.println("   " + hand1[i].getCard());

E →    System.out.println("\nDeck #1: Draw 9 random cards. Receive in SORTED order. Show using 'getCard':");
       Card[] hand2 = deck1.drawCards(9, true);
       for (int i = 0; i < hand2.length; i++)
           System.out.println("   " + hand2[i].getCard());

F →    System.out.println("\nShow the same 9 cards using 'toString':");
       for (int i = 0; i < hand2.length; i++)
           System.out.println("   " + hand2[i]);

G →    System.out.println("\nDeck #1: Draw 8 random SPADES cards. Receive in SORTED order. Show using 'getCard':");
       Card[] hand3 = deck1.drawCards(8, "spades", true);
       for (int i = 0; i < hand3.length; i++)
           System.out.println("   " + hand3[i].getCard());

H →    System.out.println("\nDeck #1: Draw 8 random RED cards. Receive in SORTED order. Show using 'getCard':");
       Card[] hand4 = deck1.drawCards(8, "rEd", true);
       for (int i = 0; i < hand4.length; i++)
           System.out.println("   " + hand4[i].getCard());

I →    System.out.println("\nIs the Six of Hearts in Deck #1?  " + deck1.isCardInDeck(new Card(6, "hearts")));
       System.out.println("Number of cards in Deck #1:  " + deck1.getNumCardsInDeck());

J →    System.out.println("\nDeck #1: Draw 14 random cards. Receive in RANDOM order. Show using 'toString':");
       Card[] hand5 = deck1.drawCards(14, false);
       for (int i = 0; i < hand5.length; i++)
           System.out.println("   " + hand5[i]);

K →    System.out.println("\nNumber of cards in Deck #1:  " + deck1.getNumCardsInDeck());
       System.out.println("Attempt to draw 4 random cards from Deck #1.");
       Card[] hand6 = deck1.drawCards(4, false);
       System.out.println("Length of returned array of Card objects:  " + hand6.length);

L →    System.out.println("\nShow all of the cards in Deck #1 (uses 'toString'):  " + deck1);
```

```
M →     System.out.println("\nDraw a random card from Deck #1 using the 'drawCard' method.");
        Card card3 = deck1.drawCard();
        System.out.println("Show the random card using the 'getCard' method:  " + card3.getCard());
        System.out.println("Show all of the cards in Deck #1 (uses 'toString'):  " + deck1);

N →     System.out.println("\nCan the drawn card be returned to Deck #1?  " + deck1.returnCardToDeck(card3));
        System.out.println("Show all of the cards in Deck #1 (uses 'toString'):  " + deck1);

O →     System.out.println("\nRank of first card in above array of 14 Card objects:  " + hand5[0].getRank());
        System.out.println("Suit of first card in above array of 14 Card objects:  " + hand5[0].getSuit());
        System.out.println("Color of first card in above array of 14 Card objects:  " + hand5[0].getColor());

P →     System.out.println("\nReset Deck #1.");
        deck1.resetDeck();
        System.out.println("Number of cards in Deck #1:  " + deck1.getNumCardsInDeck());

Q →     System.out.println("\nAttempt to draw -5 random purple cards from Deck #1.");
        Card[] hand7 = deck1.drawCards(-5, "PuRpLe", false);
        System.out.println("Length of returned array of Card objects:  " + hand7.length);

R →     System.out.println("\nNumber of cards in Deck #2:  " + deck2.getNumCardsInDeck());

S →     System.out.println("\nDeck #2: Draw 26 random BLACK cards. Receive in SORTED order. Show with 'getCard':");
        Card[] hand8 = deck2.drawCards(26, "BLACK", true);
        for (int i = 0; i < hand8.length; i++)
            System.out.println("    " + hand8[i].getCard());

T →     System.out.println("\nDeck #2: Draw 26 random RED cards. Receive in SORTED order. Show with 'toString':");
        Card[] hand9 = deck2.drawCards(26, "red", true);
        for (int i = 0; i < hand9.length; i++)
            System.out.println("    " + hand9[i]);

U →     System.out.println("\nNumber of cards in Deck #2:  " + deck2.getNumCardsInDeck());
        System.out.println("Show all of the cards in Deck #2 (uses 'toString'):  " + deck2);

V →     System.out.println("\nUse 'drawCard' to draw and show a random card from Deck #2:  " + deck2.drawCard());
    }
```

Sample Output from the Sample Calling Class

```
A → Create two decks of playing cards.
    Number of cards in Deck #1:  52

B → Create a random Card object.
    Can the new card be added to Deck #1?  false                          ← This fails because
                                                                            the deck is full.
C → Draw a random card from Deck #1 using the 'drawCard' method.
    Show the random card using the 'toString' method:  Suit = Spades     Rank = Seven
    Show the random card using the 'getCard' method:  Seven of Spades

D → Deck #1: Draw 9 random cards. Receive in RANDOM order. Show using 'getCard':
       Eight of Diamonds
       Six of Clubs
       Queen of Spades
       Eight of Clubs
       Four of Diamonds
       Nine of Clubs
       Jack of Hearts
       Queen of Clubs
       Seven of Clubs

E → Deck #1: Draw 9 random cards. Receive in SORTED order. Show using 'getCard':
       Ace of Clubs
       Six of Diamonds
       Jack of Diamonds
       Four of Hearts
       Eight of Hearts
       Queen of Hearts
       King of Hearts
       Four of Spades
       King of Spades
```

```
F → Show the same 9 cards using 'toString':
       Suit = Clubs      Rank = Ace
       Suit = Diamonds   Rank = Six
       Suit = Diamonds   Rank = Jack
       Suit = Hearts     Rank = Four
       Suit = Hearts     Rank = Eight
       Suit = Hearts     Rank = Queen
       Suit = Hearts     Rank = King
       Suit = Spades     Rank = Four
       Suit = Spades     Rank = King

G → Deck #1: Draw 8 random SPADES cards. Receive in SORTED order. Show using 'getCard':  ← Based on 'C' and 'D'
       Ace of Spades                                                                       and 'E' above, this
       Three of Spades                                                                     draw attempt may not
       Five of Spades                                                                      always be successful.
       Six of Spades
       Eight of Spades
       Nine of Spades
       Ten of Spades
       Jack of Spades

H → Deck #1: Draw 8 random RED cards. Receive in SORTED order. Show using 'getCard':
       Ace of Diamonds
       Two of Diamonds
       Three of Diamonds
       Five of Diamonds
       Seven of Diamonds
       Ace of Hearts
       Three of Hearts
       Seven of Hearts

                                                                                           Due to randomness,
I → Is the Six of Hearts in Deck #1?  true                                              ← this can just as
    Number of cards in Deck #1:  17                                                        easily be false.

J → Deck #1: Draw 14 random cards. Receive in RANDOM order. Show using 'toString':
       Suit = Clubs      Rank = King                                                   ← Notice the spacing
       Suit = Clubs      Rank = Two                                                       (how "Rank" is lined
       Suit = Hearts     Rank = Two                                                       up vertically with
       Suit = Hearts     Rank = Six                                                       each row of output).
       Suit = Clubs      Rank = Jack
       Suit = Hearts     Rank = Nine
       Suit = Clubs      Rank = Five                                                      There are 3 spaces
       Suit = Diamonds   Rank = Ten                                                    ← between "Diamonds"
       Suit = Hearts     Rank = Ten                                                       and "Rank".
       Suit = Clubs      Rank = Three
       Suit = Hearts     Rank = Five
       Suit = Spades     Rank = Two                                                       There are 6 spaces
       Suit = Clubs      Rank = Four                                                   ← between "Clubs"
       Suit = Diamonds   Rank = Queen                                                     and "Rank".

K → Number of cards in Deck #1:  3
    Attempt to draw 4 random cards from Deck #1.
    Length of returned array of Card objects:  0

L → Show all of the cards in Deck #1 (uses 'toString'):  {Ten of Clubs | Nine of Diamonds | King of Diamonds}

M → Draw a random card from Deck #1 using the 'drawCard' method.                        Notice the braces and
    Show the random card using the 'getCard' method:  Nine of Diamonds                  pipe symbols in 'L'
    Show all of the cards in Deck #1 (uses 'toString'):  {Ten of Clubs | King of Diamonds}  and 'M' and 'N'.

N → Can the drawn card be returned to Deck #1?  true
    Show all of the cards in Deck #1 (uses 'toString'):  {Ten of Clubs | Nine of Diamonds | King of Diamonds}

O → Rank of first card in above array of 14 Card objects:  13                           Above: Sorted first
    Suit of first card in above array of 14 Card objects:  Clubs                        by suit, and then
    Color of first card in above array of 14 Card objects:  Black                       numerically by rank.

P → Reset Deck #1.
    Number of cards in Deck #1:  52

                                                                                           Number and type are
Q → Attempt to draw -5 random purple cards from Deck #1.                                ← both invalid; either
    Length of returned array of Card objects:  0                                           one causes failure.
```

R → Number of cards in Deck #2: 52

S → Deck #2: Draw 26 random BLACK cards. Receive in SORTED order. Show with 'getCard':
Ace of Clubs
Two of Clubs
Three of Clubs
Four of Clubs
Five of Clubs
Six of Clubs
Seven of Clubs
Eight of Clubs
Nine of Clubs
Ten of Clubs
Jack of Clubs
Queen of Clubs
King of Clubs
Ace of Spades
Two of Spades
Three of Spades
Four of Spades
Five of Spades
Six of Spades
Seven of Spades
Eight of Spades
Nine of Spades
Ten of Spades
Jack of Spades
Queen of Spades
King of Spades

T → Deck #2: Draw 26 random RED cards. Receive in SORTED order. Show with 'toString':
Suit = Diamonds   Rank = Ace
Suit = Diamonds   Rank = Two
Suit = Diamonds   Rank = Three
Suit = Diamonds   Rank = Four
Suit = Diamonds   Rank = Five
Suit = Diamonds   Rank = Six
Suit = Diamonds   Rank = Seven
Suit = Diamonds   Rank = Eight
Suit = Diamonds   Rank = Nine
Suit = Diamonds   Rank = Ten
Suit = Diamonds   Rank = Jack
Suit = Diamonds   Rank = Queen
Suit = Diamonds   Rank = King
Suit = Hearts   Rank = Ace
Suit = Hearts   Rank = Two
Suit = Hearts   Rank = Three
Suit = Hearts   Rank = Four
Suit = Hearts   Rank = Five
Suit = Hearts   Rank = Six
Suit = Hearts   Rank = Seven
Suit = Hearts   Rank = Eight
Suit = Hearts   Rank = Nine
Suit = Hearts   Rank = Ten
Suit = Hearts   Rank = Jack
Suit = Hearts   Rank = Queen
Suit = Hearts   Rank = King

U → Number of cards in Deck #2: 0
Show all of the cards in Deck #2 (uses 'toString'): {}

V → Use 'drawCard' to draw and show a random card from Deck #2: null