SI: Encapsulation and Other Random Java Things

**Encapsulation/Design Principles:**

Congrats you got hired as a software engineer at a startup that makes online card games. They are redesigning their code base and you are in charge of writing the class that represents a deck of cards. With your infinite wisdom, you came up with a clever trick to encode both the suit and rank of the card in one. The class you wrote is shown below.

public class Deck {

/\* Each value in the array can range from 1 to 52. 1 is an ace of diamonds, 2 is a two of diamonds etc.

This continues for all suits in the order of D -> H -> S -> C

For example, a 14 is an ace of hearts and a 27 is an ace of spades.

The top of the deck is the BACK of the array.

\*/

private int[] deck;

// return this.deck

public int[] getDeck();

}

Now that you did that, you go one to your next task which is to write a method that takes the top five cards, prints the value to the screen, and removes them from the deck (you can just print the encoded value). Note this method is in a different class.

public class someClass {

//assume this get initialized

Deck deck;

public void printCards() {

**}**

While you were doing that, your colleague, Anthony, had just learned about ArrayLists and as a result changed the class to the following

public class Deck {

private ArrayList<Integer> deck;

// return this.deck

public ArrayList<Integer> getDeck();

}

Oh no, now your method doesn't work. Fix it below with the new Deck implementation. Remember the methods get(), remove(), and size().

public void printCards() {

While you were doing that, your manager, Jeff, decided both ideas were bad. He changed the Deck class to the following and added a Card class.

public class Card {

private String suit;

// 1 = Ace, 2 = 2, ... , 11 = Jack, 12 = Queen, 13 = King

private int rank;

}

public class Deck {

private ArrayList<Card> deck;

// return this.deck

public ArrayList<Card> getDeck();

}

Great, printCards doesn't work again. You notice a pattern with these design flaws, so you change the deck class to the following.

public class Deck {

private ArrayList<Card> deck;

// removes and returns the card on the top of the deck.

public Card drawCard();

}

So now you can fix the printCards method hopefully for the last time.

public void printCards() {

Your colleague, Anthony, finally learns about linked lists and decides to change the deck class to the following.

public class Deck {

private LinkedList<Card> deck;

// removes and returns the card on the top of the deck.

public Card drawCard();

}

He also changes the drawCard() method to accommodate these changes

Do you have to change your printCards method?

If you change the implementation of how the deck is stored, in how many other classes do you have to change code?

How would you describe what you did to fix the issue?

**Keywords:**

For each of the following Java keywords, say what they mean or how one might use them.

instanceof extends

super this

class static (this has multiple meanings depending on what it is used with)

**Equals Method:**

Write an equals method for the following car class.

public class Car {

private String color;

private int year;

private boolean isFourDoor;

private String make;

...

// Two cars are equal if all of their private instance variables are equal

// This class has NO getters or setters

public boolean equals (Object o) {