Dealing a Card From a Deck

(designed to accompany the AP® Computer Science A Elevens Lab)

- Consider the following questions first:

1. Where are the actual Card objects stored when we build a deck of cards?
2. Which object actually deals a card: the card or the deck? Which class will need the code for dealing cards?
3. What type of code do we need to implement in order to perform a "deal card" action?
4. What would a method header look like for a deal() method if it were to select a card and return a card?
5. What code would need to be in the deal() method? Be sure to consider the following:
    - One strategy for dealing a card is to randomly select a card from the ArrayList, delete it and then return it. While good logic, it is not how the code will be handled in this lab. This logic would delete cards from the deck completely, meaning a completely new deck object would need to be created to play again.
    - A second strategy (one which should be used in this lab) involves NOT deleting/removing the card from the ArrayList. Instead, the first time that the deal() method is called, the last card in the ArrayList will be dealt using a reference to the size of the ArrayList. The size variable will then be decremented so that the code is "tricked" into thinking that the size of the deck just shrunk. Each time the deal() method is called this process repeats, continually dealing what the code believes to be the "last card".
    - Be sure to add code that will check to see if the deck is "empty". If so, the deal method will need to return "null" (empty Card) no cards remain to deal! This can be done two ways (by using the size() method and by using the isEmpty() method). Be sure to use the isEmpty() method to perform this task.

- There are no instance variables to be declared in this lab as you will be adding a method to an already-created class:

- Below is the framework for the basic deal() method. How would you complete it?

```
public Card deal(){
        //code to change the size of the ArrayList (shrink the deck … after all, you are dealing a card
        //code to check to see if all the cards are gone (using the isEmpty method)
        //code to return the "last card" of the ArrayList (using the size variable to do so)
 }
```

- Use the following code to begin your DeckTester Class (starts with only 4 cards to simplify a deck of cards)

```
String[] types = {"A","B","C","D"};          // The 4 types of cards for this card game
String[] suits = {"DOG","PIG","COW","BAT"};  // The 4 pictures shown on the cards for this card game
int[] values = {6,7,8,9};                    // The 4 point-values for cards in this card game

Deck myDeck = new Deck(types,suits,values);  // Builds a deck of 4 cards for this card game
```

- Add code to the rest of the DeckTester Class that does the following…

1. Deals the 4 cards and stores them into new Card objects (call these objects card1, card2, card3, and card4)
2. Code four sets of print statements that prints the information for card1, card2, card3, and card4 (like in lab 1)

- The beginning of the output for the DeckTester Class should look like the following (with more after)…

```
Here is the first card ...
D
BAT
9
D of BAT (point value = 9)

Here is the second card ...
C
```