AP Computer Science A
Winter Quarter 2018-2019

Fall Quarter covered **Structured** and **Procedural** Programming.
Winter Quarter covers Object Oriented Programming.
Spring Quarter covers Algorithms, Problem-Solving, Software Design, and test preparation for the AP Exam in May.

Structured

- main()
- libraries and packages; importing
- function calls
- variables
- operators
- decisions
- loops

Procedural

- methods
- functional decomposition--the process where a single large complex task is broken down into smaller, easier to understand tasks or *subtasks.*

Object Oriented Programming

In the context of a video game like Diablo III, all of the characters, interactions, and gameplay are contained in **classes.**

Classes are the design for objects, and in the game, each class represents the different object types that interact with one another.

Our hero or main character has a class that describes what he does.
The monsters that he/she fights have a class that describe their behavior.
The pet that follows our main character has unique code the drives their behavior.

"AAA" Video Game

Class Hero
Class Follower
Class Pet
Class World
Class Monster
Class Weapon
Class Armor
Class Item
Class Money
Class Controller
Class UI
Class Physics
Class 3DEngine
etc...

Class Design

Name of the Class

> Variables - STATE
> Constructor - when you use the *new* keyword, it is calling this!
> Methods - BEHAVIOR

--

Card Project

Playing cards
52 unique cards (we don't use the Jokers)
Suit - Clubs, Diamonds, Hearts, Spades
Rank - 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

Step 1
Create a class to represent an individual playing card.
Create variables to represent the rank, suit, and value*
* *each different game played using cards will assign a value to them; otherwise cards don't have any specific values other than the symbols and numbers printed on them.*
Remember, the constructor must have the same name as the class!
Create a toString method that returns the card's suit, rank, and value in the following format:

```
A of Spades (value = 11)
Q of Hearts (value = 10)
5 of Clubs (value = 5)
```

Create a CardTest class to instantiate a few cards and test them by sending the card objects to System.out.println().

Step 2
Take your Card Class, and incorporate it into a Deck Class.
Before you write any code, write out the design for this class. What variables must it have? What will the constructor do to instantiate a deck? What variables must we pass to the constructor to create a standard playing card deck? What kinds of methods does a deck have?

CLASS DESIGN: Class Deck

- Variables
  - ArrayList of type Card
- Constructor with parameters
  - Constructor accepts the following parameters:
    1. array of Ranks
    2. array of Suits
    3. array of Values
  - Creates cards based on those ranks, suits, and values and stores them in the ArrayList
- Methods
  - toString()
  - deal()
  - shuffle()

Implement the basic Deck design with variables, constructor, and a single method, deal() at this time.

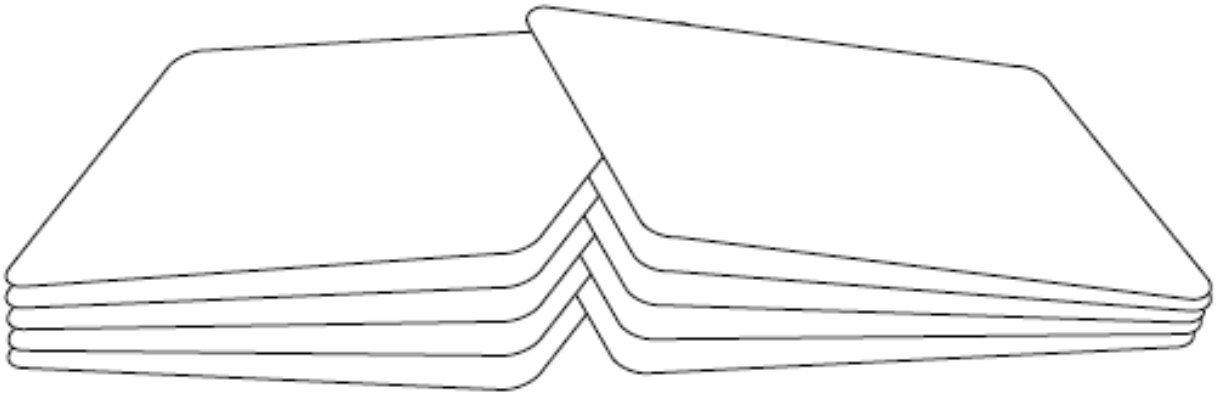
Write a TestDeck class that instantiates a new deck based on the ranks and suits and values you want to use, and tests it by display the deck contents to screen, and calling the deal() method a few times to retrieve a card out of the deck.

Part 3: Shuffling

There is a built-in shuffling function in the Collections package of Java. We are not going to use it here, because then you wouldn't learn how to come up with your own shuffling algorithm, or design an algorithm yourself. While library functions are very useful, sometimes they get overused in beginning programming classes to the point that young "programmers" aren't really doing any programming, they're just calling functions all the time.

Part 3a: Perfect Shuffle

A perfect shuffle divides a deck in two and interleaves the cards one after another. It is called a "perfect" shuffle because after eight iterations, the deck returns to its initial state. It is not ideal as a shuffling algorithm which is usually intended to randomize the state of the deck. However, it is useful for us to implement from an algorithm design standpoint.



Implement a method called **perfectShuffle()** which interleaves the deck as illustrated above. Test it by calling it eight times and determining if the deck has returned to its original instantiated (sorted) state. *Do not start coding*

yet...write out the steps you are going to perform to accomplish the shuffle first before writing any code! ***Show me your algorithm first!***

Part 3b: Randomization in Shuffling

PerfectShuffle clearly isn't "perfect." In fact, it's a terrible algorithm to use for deck randomization. What we need to do is to come up with an algorithm that can randomize the contents of the deck. Your homework is to come up with your own way to perform this randomization.

You might want to get a physical deck of cards and consider all the ways you can transpose the position of cards in the deck to achieve this randomization.

Write out your algorithm for a randomization shuffle and share it with the class at our next meeting. Some of you will come up with similar solutions, and we will vote on one to implement for our Deck class.