Unit 3 - Class Notes 2

What is a class?

A class is the **design** for an object.
A class has variables (state) and methods (behavior).
A class is an abstract data type (ADT).

What is an object?

An object is an **instance** of a class.
We create many instances from a single class design.
An object has unique state.

--

The goal of **CardProject** is to develop Card and Deck classes for use in card games like Blackjack and Poker. The simulation of these games allows us to evaluate the statistical outcomes as well as determine the pseudorandom nature of the machine's RNG.

We developed a **Card** class last week, and began working on **Deck**.

Deck is a class that *uses* the **Card** class within it.

--

toString() is a special function that allows us to pass an object to the System.out.println() method, and return a string that represents that object.

Encapsulation, or information hiding, is part of the process of developing classes, and objects, and methods. Our goal is to conceal the implementation of a specific program from the callers of those functions.

--

A perfect shuffle divides the deck evenly in half, then interleaves the contents, alternating from each half. It is "perfect" in the sense that it is a balanced shuffle. It is not a great algorithm for randomization because after eight perfect shuffles, the deck is returned to its initial state.

Perfect Shuffle Algorithm:
1. find the midpoint of the deck
2. divide the deck from 1 to the midpoint (top half) and midpoint+1 to the end (bottom half).
3. make a new array and then alternate taking cards from the two halves until there are no cards left.
4. replace the original array with our shuffled array.

The most memory inefficient way of doing this is to create two new arrays for the split deck.

Faster and more efficient, but more complicated to code, is to only make a single new array.

Perfect Shuffle illustrates how we can develop an algorithm, but it is entirely unsuitable for deck randomization.

Coming up with a better shuffling algorithm:

"Memory-intensive Shuffle"
1. generate a random deck position
2. remove that card and put it at the bottom of a new deck
3. repeat until there are no cards left in the original deck
4. copy the new deck to the actual deck

"Selection Shuffle"
1. for each position in the deck (from 0 to the end)
2. generate a random deck position and swap that card

"Riffle shuffle"
1. Divide the deck into two unequal halves.
2. *Interleave* the cards at a random value of 1-4.

"**Pile Shuffle"**
1. for a specified number of iterations...you choose.
2. generate a random range from 1-8
3. grab that number of cards from their location and put them on the bottom of the deck.

--

Before you implement each one, please write out the algorithm in pseudocode steps so that you can write the code more easily.

Test your shuffle multiple times so that you can verify that the outcome of the deck is mostly random.

Unit Testing - this is where you validate that the individual parts of your method or the entire method is working as designed.

Integration Testing - this is where you validate that the entire software system that you have develoepd is working as designed.

--

In eLearning, there is a survey asking you to select which of the potential days for us to have a class meeting is possible in your schedule. *Complete the survey before Sunday, December 16th--we are not meeting that day.*

December 26-30 Ardent Winter Boot Camp between 9am-3pm
Sunday, January 6, 2019
Sunday, March 4, 2019 ("Ardent Spring Break")

If you want to meet 1-on-1 during the Winter Boot Camp, or alternatively want to send me code to evaluate, contact me at:

gene.wie@gmail.com
(949) 542-6858