AP Computer Science A
Winter 2018 Class Notes 4

<u>Today's Agenda:</u>
Classes and Objects
Inheritance and Polymorphism
Cohesion and Coupling
Designing Software Systems from an OOP Perspective
Project: Card and Deck (tie in with Elevens Lab) - Game of 21

--

What is a class?
The **design** for an object.

What is an object?
An **instance** of a class.

What is an instance?
An *object* instance is a single occurrence of a *class*. One *instantiates* a <u>new</u> object from the class design.

What is inheritance?
A property of OOP where a subclass (child class) takes on all of the state and behavior of its superclass (parent class). This is generally described in a movement from <u>general</u> to <u>specific</u> in the context of a *class hierarchy*.

What is polymorphism?
poly = many
morph = shapes
A property of OOP where a superclass can become one of its subclass type, in a direct movement from general to specific.

What is cohesion?
A class that is cohesive effectively describes a single concept or idea. The "Swiss army knife" approach to creating classes is a poor one.

What is coupling?

Coupling describes the number of interdependencies that a software system has. The greater the number, the more fragile the entire system is, as a single change to one class could cause many other classes to malfunction.

The goal of software design is to create software the demonstrates **high cohesion** and **low coupling**.

--

Connect to the new WiFi router:
`STEAMFORALL`

Password:
`Steam123!`

--

Game of 21

This game is also known as "BlackJack" and is a card game for between 2-6 players using a standard 52-card playing card deck.

If you are unfamiliar with this game, please read this article:
https://en.wikipedia.org/wiki/Blackjack

The simple goal of the game is to get the cards that you have in your hand to total 21. Other players in the game have the same goal. The player who has the highest total that is not greater than 21 is the winner.

During a turn, two cards are dealt to you. You add up the value of the cards and determine whether you want the dealer to give you another card ("hit") or you want to stop ("stand"). If you "hit" and get another card, you recalculate your total. If it is not over 21, you can decide to continue the process, or stand. If it goes over 21, you automatically lose ("bust").

In a two-player version of this game on a computer, the computer player serves as the dealer that distributes the card. A round of the game will look like this:

1. Dealer deals two cards to the player, and two cards to itself, in alternating deals from the top of the deck.

2. Player looks at their cards, chooses to hit or stand.

3. Computer looks at their cards, chooses to hit or stand. Common logic here is to hit on 16 and stand on 17, and this is actually the dealer "rule" in many casinos.

4. Compare the totals of each hand of cards, highest wins.

The values of cards in the game look like this:

2 = 2
3 = 3
4 = 4
5 = 5
6 = 6
7 = 7
8 = 8
9 = 9
10 = 10
J = 10
Q = 10
K = 10
A = 1 or 11

The Ace is a modal card, meaning that its value depends on the state of the hand. If the value of the Ace at 11 would cause the hand to bust, it changes in value to 1.

An initial opening hand with an Ace and either a 10, J, Q, or K is equal to 21 and is called a BlackJack. This hand automatically wins unless another player also has a BlackJack. This hand also wins even if another player in the game reaches 21 by hitting.

--

Create a **GameOf21** or **BlackJack** class that plays the above described game.

You should create a **Hand** class to deal with the individual BlackJack hands held by the dealer and the player. *Hint: a hand is like a small deck that only contains a few cards.*

```
public class BlackJack
{
    public static void main( String[] args )
    {
        // main game code goes here
        // and the steps of the algorithm are...
        // create a deck and shuffle it
    }
}

import java.util.ArrayList;
public class Hand
{
    // variables
    private ArrayList<Card> cards;
    private int handValue;

    // constructor
    // methods
}
```

Write out the algorithm for one game first before writing any code!

Dealing with the Ace Problem

There are two ways we can deal with handling an ace being 1 or 11:

1. We can directly change the value of the card object itself from 11 to 1 in the corresponding situation when we **add** it to the hand.

2. We can calculate the **value** of the hand separately without changing the value of the card.

Either way you solve it will have direct design consequences on your BlackJack main() function.

Common Issues

1. If you're trying to access an object and you get a "Null Pointer Exception" this means that you haven't actually instantiated that object yet, and it literally does not exist. The concept of *null* is one that you can actually reference in Java, as the keyboard **null** refers to a data object that does not exist.

2. Some of you are trying to compare the rank using the ==. This isn't going to work because Strings are not primitive data types! You must use the **equals()** method of the String to make a proper lexical comparison. If you use == by mistake, all your program will tell you is whether you have two strings or not.

3. Consider carefully what kind of behavior you write methods for with your hand. Each class should have clearly defined behavior that is **cohesive**. There are some things that each class should not need to know despite being used for a greater purpose ("the game"). The hand class has no idea what the rules of the game are, it just holds cards and reports statistics. The Card and Deck classes should not know what game they are being used for...that is dealt with when we instantiate the deck for the game in the main() function.

Testing

There are a number of things we have to address in testing this program:

1. Consider the interface to the program. What is your end-user expecting in terms of a game? What should they be able to see? What should they NOT be able to see? Recall that in computer games, you must explicitly define everything that happens...it's not like real life! Announcing the different steps (Player hits, player stands, dealer hits, dealer stands) is important in outlining the game's behavior.

2. A good way to test the BlackJack condition (where someone is dealt an Ace and one card worth 10, like 10, J, Q, or K) is to create a new deck but don't shuffle it! The deal will hand the player the A and Q of Spades, and the dealer the K and J of Spades, giving them hand totals of 21 (a BlackJack) and 20.

3. You should at some point create a test deck with fewer cards that is "stacked" (or customized) in such a way so you can test difficult situations, like a player being dealt four aces in a row! This is an example of *unit testing*, where we have to ensure that one small part of the program (the Hand class) works as intended.

4. You don't have to create any more Card objects in the main function beyond the first four which are used to create the Hand objects. You can deal() directly from your deck and give the Card directly to the add() function of your Hand objects. This is one of the "dealing with data flow" considerations of this project.

--

Compound boolean expressions in Java are done using the following symbols:

AND is **&&** also known as "double ampersand"
OR is **||** also known as "double pipe"

--

Homework

1. Go and watch an actual BlackJack game so you can see how the actual game works, and how some of the special exceptions take place. It's one thing to read an article about how the game works, and one to actually see it all the way through and then code a functioning example of it.

2. Once you have your BlackJack game tested and working, go and check out the first canonical AP lab, which is called **Elevens**. Elevens is very similar to our project in that it asks for the creation of Card and Deck classes to play a game. The Elevens lab comes with a pre-made GUI (graphical user interface) so that the game can be accomplished with point-and-click gestures. The concept of the labs is that the course has a 20-hour coding requirement that involves the application of the CS principles you have learned in a practical fashion involving the creation of three programming projects. Besides Elevens, there are also two other labs, one which involves String processing for a ChatBot, and one that looks at 2D arrays and how they are used for image processing.