# COP 3330 - Assignment 3

## Due: Friday March 4, 2014 at 11:59 p.m.

Assignment 3, Parts 1 and 2 are reasonably challenging. Once you have completed them, you ought to be able to implement most popular cardgames easily.

## Part 1

(Card Shuffling and Dealing). Modify the application of Fig. 7.11 in the Deitel text to deal a five-card poker hand. Then modify class *DeckOfCards* of Fig. 7.10 to include methods that determine whether a hand contains:

a) a pair

b) two pairs

c) three of a kind (e.g., three jacks)

d) four of a kind (e.g., four aces)

e) a flush (i.e., all five cards of the same suit)

f) a straight (i.e., five cards of consecutive face values)

g) a full house (i.e., two cards of one face value and three cards of another face value)

[**Hint**: Add methods *getFace* and *getSuit* to class *Card* of Fig. 7.9.]

## Part 2

(Card Shuffling and Dealing). Use the methods developed in Part 1 to write an application that deals two five-card poker hands, evaluates each hand and determines which is better.

**Please submit this Word document (fill it out) and the .java file from Part 2. No need to submit Part 1 as it's incorporated into Part 2. Thanks.**

**Analysis:**

(Describe the problem including input and output in your own words.)

The program will take no input from the user and deal two poker hands of five cards. The program will print out to the screen the two hands and then evaluate which hand is greater and print that result to the screen.

**Design:**

(Describe the major steps for solving the problem. Create a UML diagram to accompany your major steps).

The project will be consisted of four classes, DeckOfCardsTest, DeckOfCards, Card, and CardHand.

DeckOfCardsTest:

This is the main class of the project and is responsible for all the print calls to the screen.

DeckOfCards :

This class is the template for a DeckOfCards object. It has three methods; the constructor, shuffle, and dealCard. Upon creation, a DeckOfCards object will consist of 52 poker cards. The shuffle function will randomize their order within the deck. The dealCard function will return a card within the deck and remove it from the deck.

|  |
| --- |
| DeckOfCards |
| - deck: Card  - currentCard: int  - NUMBER\_OF\_CARDS: int  - randomNumbers: Random |
| <<constructor>> DeckOfCards()  + shuffle() : void  + dealCard() : Card |

Card:

This class is the template for a Card object. It has five methods; the constructor, toString, getFace, getSuit, and getValue. toString returns a string concatenation of the card’s face and suit. getFace returns the card’s face. getSuit, returns the card’s suit. getValue returns the card’s numerical value, e.g. jack is 11, ten is 10 etc.

|  |
| --- |
| Card |
| - face: String  - suit: String  - value: int |
| <<constructor>> Card(cardFace : String, cardSuit : String, cardValue : int)  + toString() : String  + getFace() : String  + getSuit() : String  + getValue() : int |

CardHand:

This class is the template for a CardHand object.

|  |
| --- |
| CardHand |
| - hand : Card[]  - SIZE\_OF\_HAND : int |
| <<constructor>> CardHand(deck : DeckOfCards)  - checkFlush() : int  - checkPairs() : String  - checkStraight() : int  + getHand() : String  + getHandRank() : int  + getCard(index : int) : Card |

**Testing**: (Describe how you tested, or will test this program)

To test my program and make sure it was recognizing specific hands, I modified the constructor of the CardHand class to make specific hands that I knew contained a valid poker hand. This made it easy to ensure the getHand method was recognizing the proper poker hands. Furthermore, to check if a specific hand was being dealt, I used a while loop in the main method to keep dealing hands until the desired poker hand was recognized. Once the hand was found, I had the program print out the contents of the hand to the screen to verify the match was accurate.

**Submit the following items:**

1. Save this Word file; submit it via Canvas Assignments on or before the due date and time.

2. Compile, Run, and Submit your .java file(s) to Canvas. You must submit the program regardless whether it is complete or incomplete, correct or incorrect.