# Alex Berliner

# A2737438

# Handout 14 Assignment 3

## Due: Friday March 1, 2013 at 11:59 p.m.

Assignment 3, Parts 1 and 2 are reasonably challenging. Once you have completed them, you ought tobe 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 pokerhand. Then modify class *DeckOfCards* of Fig. 7.10 to include methods that determine whether ahand 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)

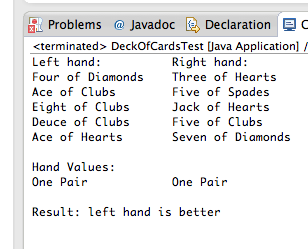
g) a full house (i.e., two cards of one face value and three cards of another face value)

[**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.

**SAMPLE OUTPUT:**



**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.)

There’s no real “input” for this program since the program itself does everything with random number generation, so the closest thing would probably be the randomly generated cards that are dealt by the program.

The output is the hands of both players playing poker, followed by the highest rank that each player’s hand got and finally which player got a higher ranking hand. If the players got the same ranking hand, high card is usually used to rank and if those are all equal the pot is split.

**Design:**

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

I started out by writing the card class and then the hand class, with the relevant data fields being the suit and the face, and then I wrote methods for each to make manipulating both classes easier.

Afterwards, I began to write methods that would determine for each hand independently whether the desired hand was found.

After writing a couple of methods that did this I realized that it would be easier to do a few things. I realized that comparing the hands independently of each other is very difficult, so I moved the hand checking code into one function that would analyze both hands.

Afterwards, I realized that it would be easier to create a class that housed these functions instead of having them in my main so that I could make better use of global variables, among other things.

After this, I tried to make each function that determines a hand type call the next, for example four-of-a-kind() would call full-house() and full-house() would call flush(). This turned out to be bad because I lose the ability to check the lower ranking hand’s best rank.

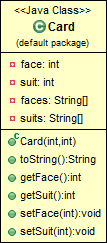
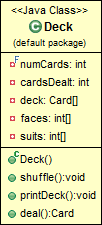
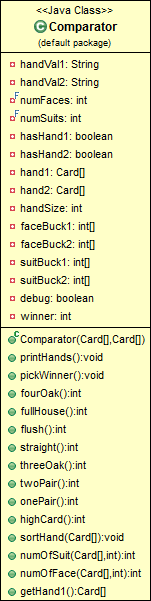
I finished by making all the rank functions independent of each other and then making a function that would call each rank function to determine all the data relevant to determine the outcome of the hand.

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

I tested this program in two different ways. The first was the unit testing of each ranking function with hands that were designed to make it return that the hand was found and not found. This was most of the testing that I did because it’s not hard to go through a bunch of different scenarios when the hands are sorted internally.

After unit testing, I tested that the comparisons of the hands were working correctly with more rigged hands.

Finally, I tested the randomly generated hands by just running the program a bunch of times and seeing if the data was making intact to other parts of the program and also seeing if the randomly generated values gave the correct answers.

****

**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.