Section 5: Poker Project

The poker project was by far the most time-consuming part of this project. For whatever reason, I had such a difficult time chasing bugs in my code and it was extremely time consuming. My Poker Project consists of 8 classes. The tester class only institutes the player class, and the player class calls almost everything that needs to be called. The player class provides output and takes player input. The first thing player does is instantiate the dealer class.

The dealer class is a class that performs all the judgements and is used to ensure that card objects get from the players hand to the deck, instead of being discarded. The dealer object is meant to do most of the work under the hood. When the dealer is instantiated, a new deck is also instantiated and declared, this deck is what the dealer uses throughout the game. The deck is not discarded after every turn, so that the experience would be more authentic. When the dealer is instantiated, there are also a lot of global variable Booleans that are set to false, more on that later.

When the player has indicated that they're ready to play, the play class will call the draw 5 class of the dealer. When this method is called, it instantiates a hand object and removes the first 5 cards from the top of the deck. Throughout the game players may remove cards from their hands or the game may do it automatically throughout the game, this is done through the returnCards method; which not only removes the card from the players hand, but it puts it back in the deck allowing it to be drawn by a different player. The toString in the hand object iterates through all the cards in the hand and calls a method in card called printCard. All the other methods in hand should only be used by the dealer.

The dealer evaluate function is the method where all the fun is when programming. Admittedly, I did implement this in a peculiar way. The evaluate method calls each method that determines what a hand is in a very particular order. Some of the comparison methods remove cards from the hand and put them back in the deck. I did this with the isPair, isTriple and isQuadruple methods because if I don't remove the cards from the player's hand, then when other methods call those methods later, it sets off false positives that aren't easy to check for. The evaluate method adds the results in a particular order to an ArrayList of type Boolean which gets fed into the scrubResults method. This method figures out what hand you have and classifies it. It will set a global String variable which can be obtained from the toString, or referenced directly, because it's public. There is also a resultID method that returns an int which represents a hand and is very easy to compare. A reset method must be called between evaluating hands so that all the strings can be set back to false.

One thing that is strange in the dealer method is that there is a lot of passing hands back and forth. I am aware that I am passing object references and that everything that alters the hand does it everywhere, but as I was going to change it, I figured if I ever write code in the future, it is not a costly enough operation to impact the performance of my program.

The deck method is instantiated when the dealer method is instantiated. The deck method has 2 for each of the loops that loop through the values of my rank and suit enumerated types to create 52 cards in an ArrayList of type card. The deck class has a shuffle method, but there is a shuffle method in both the dealer class and the play class, this one is only for those classes to use. There is also a draw method that is only used by the dealer, a toString method which should only be used for diagnostics, and putBack method; which again should only be used by the dealer.

The card method is instantiated 52 times when the deck is initialized. The card constructor takes a Rank value and a Suit value from the enumerated types and stores those values within the object. The card class has plenty of methods that aren't really supposed to be used except by the deck, or when you're getting the value for comparison (even then, most of that is done by the hand class). Other than the getters and setters, and toString which produces the cards information in text format. I have a printCard method which prints the Unicode suit symbol for the card as well as their value number or letter. This makes the game a little easier to follow because it removes a level of abstraction when playing with familiar symbols instead of just text. This printCard method is called by the hand and concatenated to print out a symbol and card value for every card in the hand.

The two enumerated types I made for this game were Suit and Rank enumerated types. I chose to do this because it was easy to customize them and it makes the programming a little more of a high-level language, as opposed to representing the suit with an integer and doing all the comparisons and computations with that method. It's also good to get some practice working with them.

My game has 3 modes that you will be prompted to play. The first option is to play 2 different hands against each other. The dealer will let you see your cards and ask if you would like to return any of them to the deck. Once that's all over, it will evaluate the hands and declare a winner.

The second option is the Monte Carlo Statistics mode. This will play the game as many times as you would like, it tabulates the data and prints them out in a ASCII chart with percentages. I have run it 100,000,000 times with no issues.

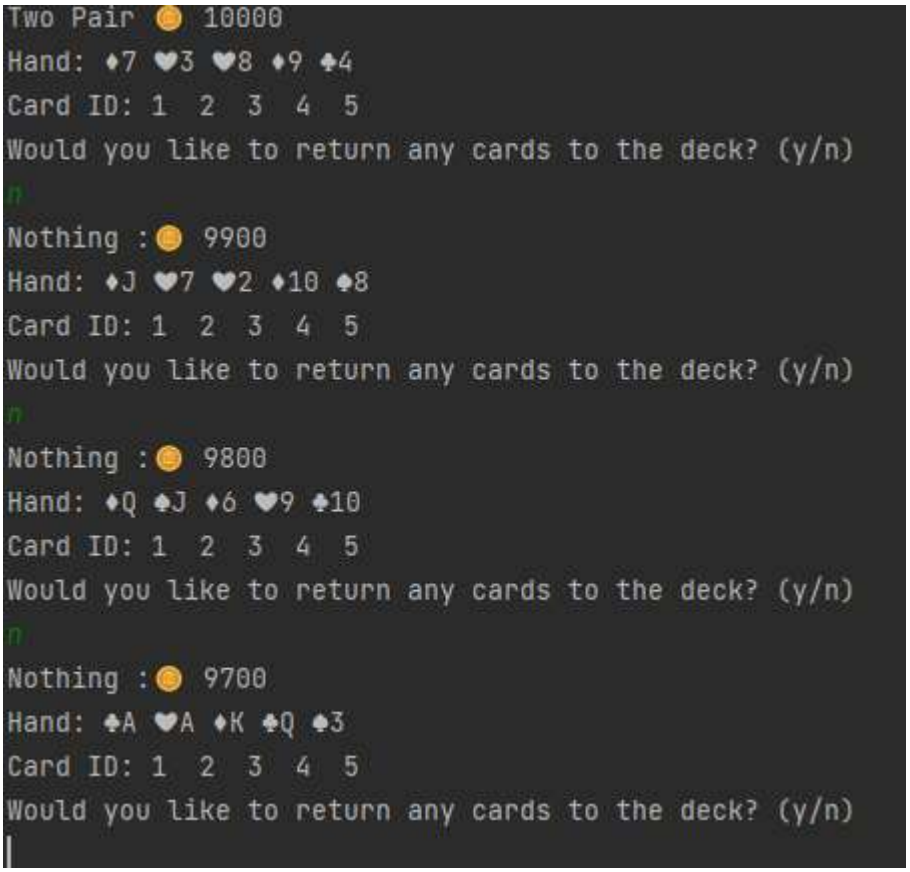
The third option is the extra credit arcade mode. You start off with 1000 coins and everytime you get nothing, you lose 100 coins. Different hands have different payouts. The game keeps running until you run out of money, or you close the program.

Monte Carlo Statistics Mode:

Game played: 10000000 times

Hand	Number of Times	Percent of Times
Royal Flush	17	0.000170
Straight Flush	138	0.001380
Four of a Kind	2353	0.023530
Full House	14437	0.144370
Flush	19745	0.197450
Straight	39195	0.391950
Three of a Kind	211148	2.111480
Two Pair	475566	4.755660
Pair	4225035	42.250350
Nothing	5012366	50.123660

Arcade Mode:


Two Pair 10000
Hand: ♦7 ♥3 ♥8 ♦9 ♠4
Card ID: 1 2 3 4 5
Would you like to return any cards to the deck? (y/n)
n
Nothing : 9900
Hand: ♦J ♥7 ♥2 ♦10 ♠8
Card ID: 1 2 3 4 5
Would you like to return any cards to the deck? (y/n)
n
Nothing : 9800
Hand: ♦Q ♠J ♦6 ♥9 ♠10
Card ID: 1 2 3 4 5
Would you like to return any cards to the deck? (y/n)
n
Nothing : 9700
Hand: ♠A ♥A ♦K ♣Q ♠3
Card ID: 1 2 3 4 5
Would you like to return any cards to the deck? (y/n)