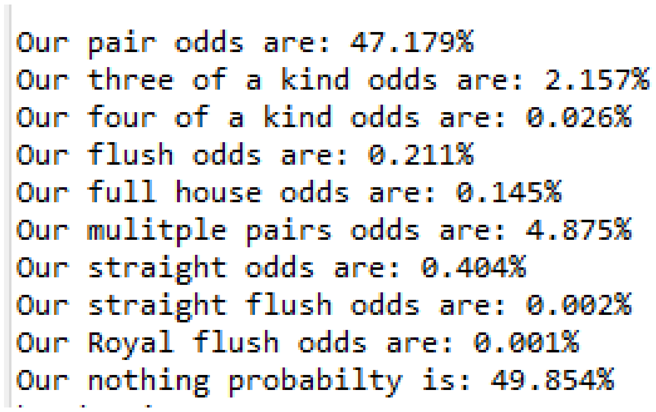
Patrick Niederhauser
Poker write up

This project asked us to create a poker simulation and find the probabilities for different cases. Although this seems straightforward this project challenged me in several different ways. For me this project was definitely the most challenging one that we have worked on throughout this semester.

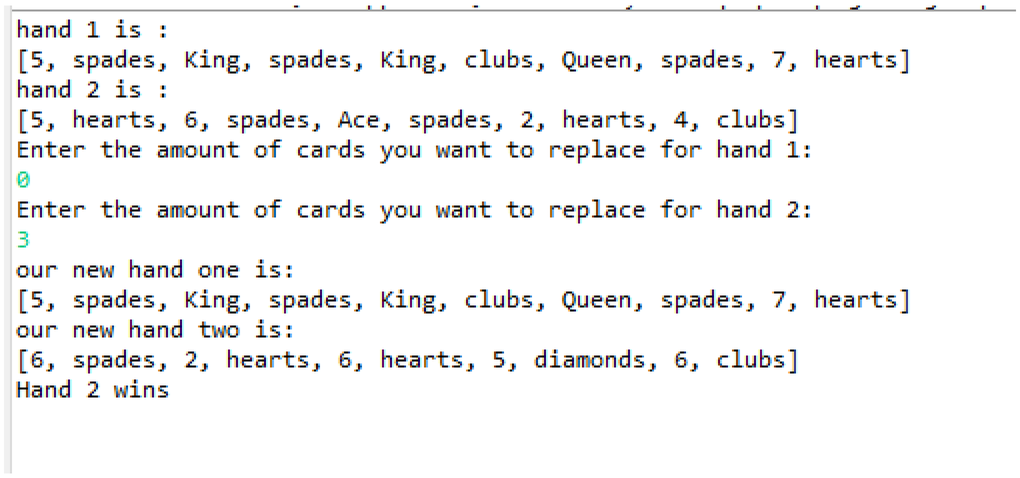
To start this project we first had to create a class called deck, this class would be responsible for creating a standard 52 card deck. To start this class I first declared a method that would create my deck, I named it CreateDeck(). This method ran a for loop that would create 52 different cards with the correct suits and numbers. I did this by creating an array with the four different suits and running a next for loop that assigned each number 4 different suits. Once I had my deck I needed a method to shuffle it, thus the creation of the method called SuffleDeck(). This method simply just used math.random() and a for loop to loop through my deck and shuffle each card randomly. After these two methods I created a tostring statement that would just print my output in a nicely ordered list separated by commas (this is important for later).

Next, I created the poker class, this class would be responsible for the actual simulation. To start this class I first needed to declare all my counter variables for my different evaluation methods. Each method would return true or false and I'm running this simulation 100,000 times so I need a way to count each positive result. After these variables I created a for loop which would run 100,000 times. I then created an object of type poker and called my river method on my new object. This method created our unique deck and gave us 5 cards from it. Since I now had my river created I wanted to split each card from its suit and number in order to create two different array lists. I wrote a for loop to do this that split on the comma i mentioned above. After this I now had two different array lists that I can now pass to each evaluation method. I First started with my pair evaluation, which simply compares the index and returns true if there is a pair. I next worked on my method to calculate three of a kind, which is similar to the pair method but just has a counter variable. If this counter variable hits 3 the method then returns true, because we then have three of a kind. Next we had to calculate four of a kind, this method is almost identical to the three of a kind, but the counter is set to 4 instead of 3. The next method I worked on was the method that evaluated a flush. This method took our suits array and just compared each element to see if it was the same, and if it was returned true. The next method I did was the method that evaluates a full house. To start this method I first had to parse all my string elements to integers. I did this by creating an algorithm that took each element and set it equal to an integer value and then added to the array. For example jacks were set to 11, queens were set to 12 and kings to 13s. Aces are a special case, because they can either be used as a 0 or a 14, so I wrote an if statement to assign it the correct value depending on the river. Once I had all my values parsed I then ran a nest for loop to see if it contained three of a kind. If it did, I then ran a loop to see if it contained a pair. If both loops produced a true result I then return true. The next method I wrote was a method for evaluating unique pairs. This method used the same algorithm as above. After I sorted my now parsed numbers I then ran a for loop to determine if we had a pair. If we do, I then removed it from the array and ran the loop again. If the for loop produced two true's then the method will return true. The next method I wrote was a method to evaluate a straight. To do this method I ran the algorithm above and called collection.sort to sort my data. I then ran a for loop to see if the new list was sequential, and if it was a return true. The final method that I wrote was the royal flush evaluation. This method first sees if the river is a flush if it is then checks to see if the river contains the needed numbers to be considered a royal flush. This would mean we would need a 10 jack queen king ace all the same suite. This is very uncommon and I didnt get one when I tested my program. My evaluation results are shown below.

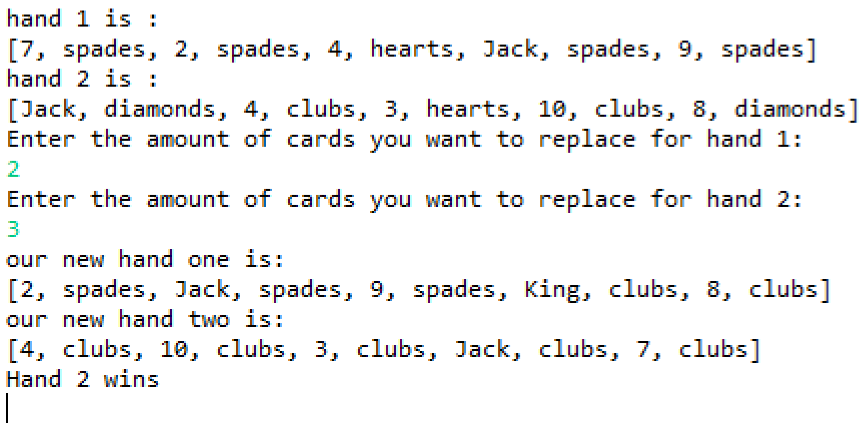


```
Our pair odds are: 47.179%
Our three of a kind odds are: 2.157%
Our four of a kind odds are: 0.026%
Our flush odds are: 0.211%
Our full house odds are: 0.145%
Our mulitple pairs odds are: 4.875%
Our straight odds are: 0.404%
Our straight flush odds are: 0.002%
Our Royal flush odds are: 0.001%
Our nothing probabilty is: 49.854%
```

The next part of this project was a game implementation, this "game" would create two different hands and compare them and determine a winner. These hands need to be made editable, meaning that players can decide to drop and draw new cards after they've seen their original hands. To start this class I first created a deck object and then created and shuffled a deck of cards. I then ran a for loop that created two different 5 card hands. I then printed these hands out so we could see what our output would look like. After this I then worked on the changing of cards system. I first took user input on how many cards the user would like to swap out. I then ran a for loop that would remove that amount of cards and add random cards back. I then repeated this process for both hands. Since I'm using the same deck I didn't have to worry about repeated cards. After this I then ran an evaluation method to determine which hand won. This evaluation method created a counter and added a value to each time one of our evaluations hit. For example a pair was one point, two unique pairs is 2 points, three of a kind is 3 points, a straight is 4 points, flush is 5 points, full house is 6 points, four of a kind is 7 points, a straight flush is 8 points, and a royal flush is 9 points. These unique point values are based off of my statistics I ran before, the rarer something is the more points it's worth. After my evaluations I return the amount of points each hand is worth. I then ran three if statements to determine which hand had the greater value, and printed the answer. An couple of example answer are displayed below:



```
hand 1 is :
[5, spades, King, spades, King, clubs, Queen, spades, 7, hearts]
hand 2 is :
[5, hearts, 6, spades, Ace, spades, 2, hearts, 4, clubs]
Enter the amount of cards you want to replace for hand 1:
0
Enter the amount of cards you want to replace for hand 2:
3
our new hand one is:
[5, spades, King, spades, King, clubs, Queen, spades, 7, hearts]
our new hand two is:
[6, spades, 2, hearts, 6, hearts, 5, diamonds, 6, clubs]
Hand 2 wins
```



```
hand 1 is :
[7, spades, 2, spades, 4, hearts, Jack, spades, 9, spades]
hand 2 is :
[Jack, diamonds, 4, clubs, 3, hearts, 10, clubs, 8, diamonds]
Enter the amount of cards you want to replace for hand 1:
2
Enter the amount of cards you want to replace for hand 2:
3
our new hand one is:
[2, spades, Jack, spades, 9, spades, King, clubs, 8, clubs]
our new hand two is:
[4, clubs, 10, clubs, 3, clubs, Jack, clubs, 7, clubs]
Hand 2 wins
```