Eli Giglietti

Abstract:

In this project we simulated blackjack. We started by making classes for the different components that we would need. We made classes for the cards, the deck of cards, one's hand, the game, and a blackjack class. Finally, we wanted to simulate many games of blackjack and see the outcomes. For this, we made a simulation class that played many games of blackjack and stored the results. In most simulations I did, ties were pretty infrequent, and the dealer won more than the player. In one example, the player won 41% of the time, the dealer won around 50% of the time, and then ties were around 9% of the time. The new data structure we worked with were lists. We used ArraysLists, which acts like a list due to the import ArrayList we used.

Results:

```
Player Hand: 10,9,Total: 19
Dealer Hand: 2,4,10,11,Total: 27
17
```

This is the first result I got for a game of blackjack. This is before I printed the initial and final hands and only had it simulate the game in one go.

```
Player Hand: 4,11,Total: 15
Dealer Hand: 6,4,Total: 10
Player Hand: 4,11,2,Total: 17
Dealer Hand: 6,4,6,4,Total: 20
Dealer Wins
```

This is the second result I got. I tested the BlackJack method but had it run a game. My main method deals, and then calls toString to print the initial hand. Then, it deals more cards if the initial total is under the given value, and the toString method is used again to print the final hands. Finally, it figures out who won and prints the correct statement.

```
Player points: 415.0 Ties: 86.0 Dealer pionts: 499.0
Player points: 0.415 Ties: 0.086 Dealer pionts: 0.499
```

This is the result of my simulation. The simulation runs many games of blackjack using a for loop and stores the data. The dealer ends up winning a bit more than the player. This is because in the game method, I have the player go first. So, if the player gets over 21, the dealer automatically wins. If the player doesn't go over, then the dealer has a chance to catch up. If neither goes over 21, then it checks who has the higher hand total, and returns who wins.

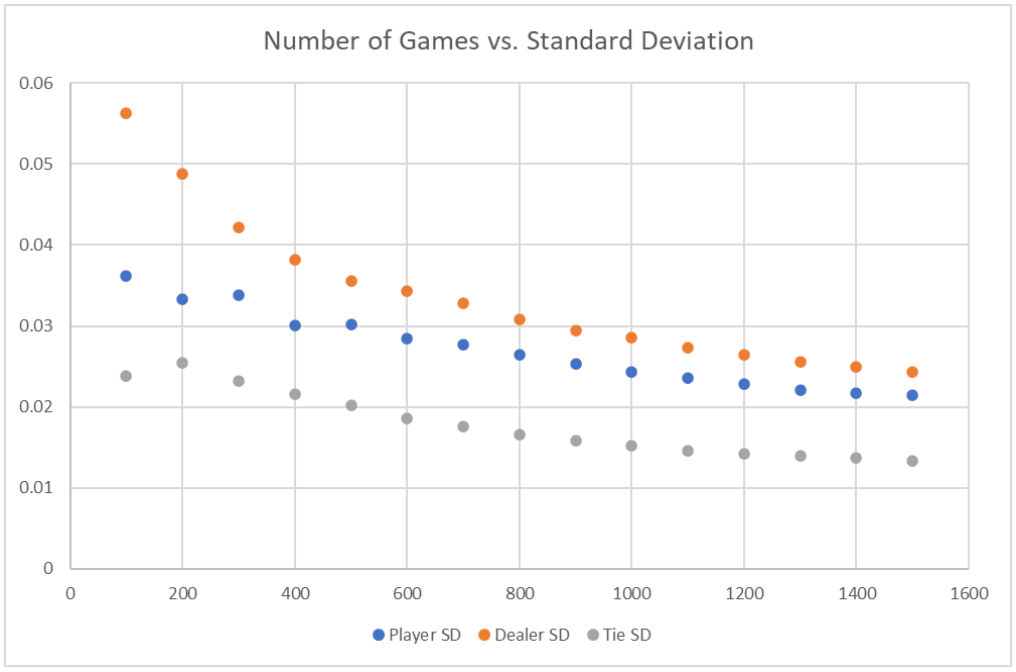
Extensions:

1.

```
How many sets do you want?
10
How many games per set?
100
0.03943665807342198
0.03159850119511648
0.033391396189829683
0.030282603813906094
0.02835406476836944
0.02677688534975196
0.025488516009724668
0.024274199801847515
0.023925841394888216
0.023183179274867207
```

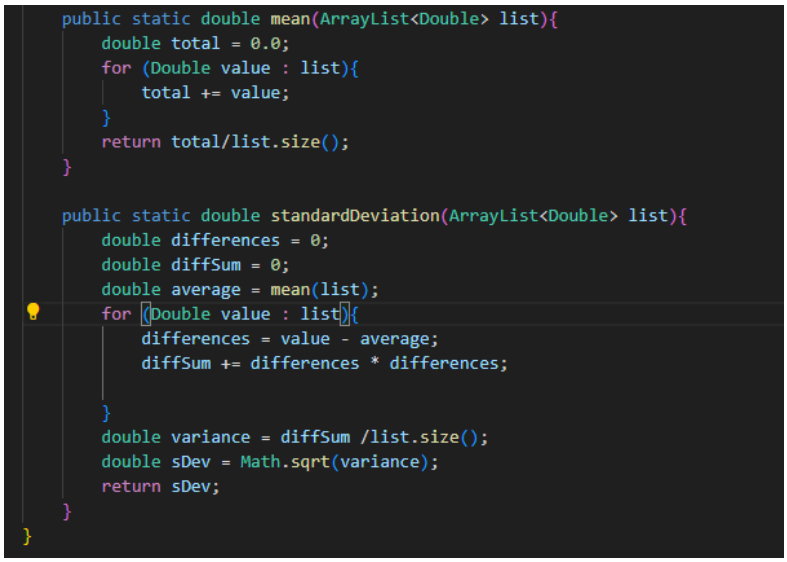
I used a java package called scanner to enable text prompts to the user. In this case, I wanted to use the text prompts to allow me to change the number of games the simulation runs. This will be helpful for my next extension.

2.



In this extensions, using the scanner, I changed the number of games the simulation was played. I would run the number of games 10 times, and calculate the standard deviation between the 10 simulations. In excel, I then plotted the standard deviation by the number of games played. As the graph shows, for the player, dealer, and ties, the standard deviation

decreases as more games are played. Additionally, the standard deviation seems to be the greatest for the dealer, second most for the player, and then the smallest for the ties.



```
    public static double mean(ArrayList<Double> list){
        double total = 0.0;
        for (Double value : list){
            total += value;
        }
        return total/list.size();
    }

    public static double standardDeviation(ArrayList<Double> list){
        double differences = 0;
        double diffSum = 0;
        double average = mean(list);
        for (Double value : list){
            differences = value - average;
            diffSum += differences * differences;

        }
        double variance = diffSum /list.size();
        double sDev = Math.sqrt(variance);
        return sDev;
    }
}
```

References/Acknowledgements:

Ryan Mogauro
Liam Cotter