CS231: Project 3

Sudoku

**Abstract**

In the project, a Sudoku solver has been coded. The current project reads from the test.txt file and prints the solution out to the terminal, the solved board. In the meantime,

**Result**

On running the simulation 1000 times thrice, the following results were produced:

Graphical user interface, text

Description automatically generated

**Simulation 1:** Player wins 43.6% of the times. Dealer wins 47.9% of the times. And it is a draw 8.5% of the times.

Graphical user interface

Description automatically generated with medium confidence

**Simulation 3:** Player wins 40.7% of the times. Dealer wins 49.7% of the times. And it is a draw 9.6% of the times.

Graphical user interface, text, application

Description automatically generated with medium confidence

**Simulation 2:** Player wins 42% of the times. Dealer wins 50.6% of the times. And it is a draw 7.4% of the times.

In summary,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Game 1 | | Game 2 | | Game 3 | |
| Games | % | Games | % | Game | % |
| Player win | 436 | 43.5 | 420 | 42.0 | 407 | 40.7 |
| Dealer win | 479 | 47.9 | 506 | 50.6 | 497 | 49.7 |
| Push | 85 | 8.5 | 74 | 7.4 | 96 | 9.6 |

Henceforth, it can be concluded that over a long period of time/a lot of games, it is always the house that wins.

These results make sense because since the player goes first, they can go bust more easily than the dealer who will go lose only when the player doesn’t bust first. Therefore, the win of the dealer does not depend on their hand if the player already went bust. As such, the dealer gains a competitive edge over the player.

As such, it does not make good financial sense to keep playing Blackjack in the hopes that one will take home more money than they came with or win their money back, if they’ve lost any.

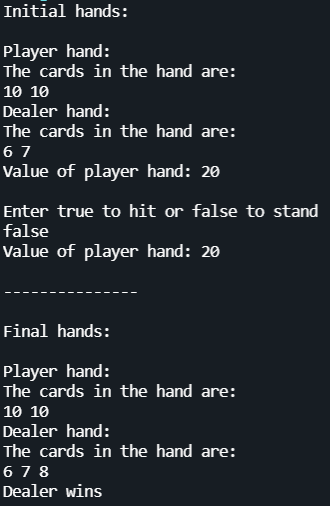
**Extension 1**

For my first extension, I have chosen to make the Blackjack game interactive. To do this, I have created a new method called *playerTurnInteractive*. This method allows for a user to use *true* as input to get hit with another card or *false* as input to stand. To get the input, I made the use of the *Scanner* class. A class called *Interactive* was then created to create an object of the *Blackjack* class and use it to play a game.

The method called *dealerTurn* was also modified to make the dealer keep hitting until the value of the dealer’s hand is less than 21. In the end, if the player has not gone bust, the dealer’s hand is checked for its value to identify the result.

The house wins when:

1. The player busts
2. The player does not bust and the dealer has 21
3. The player does not bust but the dealer’s hand is higher in value.

 Text

Description automatically generated

**Sample games**

**Extension 2**

For this extension, I increased the number of decks to six. This was done by modifying the *build()* function in the *Deck* class to include 24 each of cards with values 2-9 and 11, and 96 cards with the value 10. This makes it six decks in total

In a table below, I have plotted the results (win %) for 5 games and compared it with the results for using just one deck. An average of the five games has been taken to average out and statistical anomalies.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Serial Number | 1 deck | | | 6 decks | | |
| Player | House | Push | Player | House | Push |
| 1 | 41.7 | 49.7 | 8.6 | 41.2 | 49.6 | 9.2 |
| 2 | 42.2 | 47.3 | 10.5 | 41.8 | 48.0 | 10.2 |
| 3 | 38.9 | 52.8 | 8.3 | 41.4 | 49.0 | 9.6 |
| 4 | 40.3 | 51.9 | 7.8 | 42.7 | 49.1 | 8.2 |
| 5 | 43.3 | 48.4 | 8.3 | 44.4 | 45.8 | 9.8 |
| Average | 41.28 | 50.02 | 8.7 | 42.3 | 48.3 | 9.4 |

**Observations**

The findings from the table above are that even though there has been the difference of a few decimal places between the values of the house winning and the game ending in a push, the player wins approximately the same number of times. It can be concluded from the above data that changing the number of decks from 1 to 6 does not have a substantial effect on the chances of winning/losing/drawing this version of Blackjack.

**References/Acknowledgements**

Prof. Al Madi discussed a shuffling algorithm in class that served as a reference for the *Shuffle* class in the project. I also consulted the Java documentation for some help with ArrayList functions. No other resources/people were consulted.