**Report: Blackjack Game Program**

**Abstract:**

My Blackjack Program is a Java implementation of the classic casino card game, Blackjack. This program allows users to play Blackjack with various customizable settings, including number of players, number of decks used, the presence of high-low aces, level of user control, and different AI strategies. The primary goal of this project was to develop our understanding and control of the ArrayList data structure; however, I used several other structures in addition to the ArrayList structure. Two different AI strategies were fully implemented and in simulation neither were able to get the house advantage below 2%, nor were they able to achieve a win rate greater than about 42.6%.

**Core CS Concepts Used**

**Data Structures**:

* **HashMap:** The program uses HashMaps to manage decks, player hands, game outputs, data storage, and various other small uses. HashMaps allow for efficient storage and retrieval of the data stored in them.
* **TreeMap:** The program uses the TreeMap data structure to store ArrayLists containing players who achieved the same score. It then uses the built in sorting in the TreeMap structure to determine winners.
* **ArrayList:** ArrayLists are used to store and manage collections of cards, hands, and other data structures throughout the game. They also serve as a wrapper for many other data types when they are being packaged for data extraction and management. Also used as vectors in the unimplemented MachineLearningAI Class.

**Algorithms:**

* **Decision-Making Algorithms:** The program employs decision-making algorithms for player and dealer actions. These algorithms determine when to hit or stand based on the current state of the game. Three different strategies were conceived; however, only two were fully implemented: simpleAI, and advancedAI.
* **Card Counting (Advanced AI):** The advanced AI strategy incorporates the Omega-II card counting algorithm to make decisions. This algorithm assigns values to cards and adjusts the player's strategy based on the count, an attempt at a more sophisticated AI approach which unfortunately fell short of expectations.

**Results:**

The program successfully simulated Blackjack games based on user input settings. Similarly, the Simulation Class was successful in simulating games for up to as many as one million iterations. It provides the following results:

* **Win Percentage:** The program calculates and displays the percentages of wins for the dealer, players, and ties over multiple game sessions. The win percentage across one million games quite consistently fell within 0.7% of 38.9% using the advancedAI. Sadly, this was a worse performance than simpleAI with a win percentage across one million games within 0.5% of 42.0%.   
  Advanced: A screenshot of a computer screen

  Description automatically generatedSimple: A screenshot of a computer screen

  Description automatically generated  
    
  The simpleAI’s comparatively better performance compared to the more advancedAI could likely be adjusted by increasing therisk tolerance on the advancedAI when dealing with a “soft” hand, a hand where an ace can be read as either a 1 or 11. Similarly tweaking the weights on the linear equation might help remedy this disparity.  
    
  **Update:** Having tweaked the weights in the advancedAI method and having implemented a “soft” hand handler, I was able to bump the advancedAI method’s performance up to consistently within 0.25% of 42.2%. While not a tremendous increase this at the very least indicates an increase over the simpleAI method. However, these tweaks had an unintended consequence covered next/  
  A number on a dark background

  Description automatically generated
* **House Advantage:** The tweaks made to the advancedAI while bringing the win rate slightly above that of the simpleAI were not able to bring its results in line with the house advantage exhibited by the simpleAI. However, after further tweaking, minimizing house advantage while making efforts not to cut into the gains provided by the previous optimization, we were able to reach a house advantage consistently within 0.5% of 2.5%, all while maintaining a win rate within 0.3% of 42.2%.  
  Tweaked advancedAI: **A screen shot of numbers

  Description automatically generated**

**Discussion:**

**Fairness of the Game**

As expected, blackjack, like all other casino games, is an inherently unfair game. While neither of my AI strategies were particularly stellar, they both performed far above how the average person plays, yet neither of these AIs were able to surmount the house advantage. The only way I could imagine beating the house would be the card counting MachineLearningAI which I ran out of time to implement.

**Extensions:**

**Multiplayer Play**

I

**Multi-Deck Play**

**Interactive Gameplay**

I implemented a feature for user interaction. If you run the Blackjack Class directly the main method takes a bunch of user input to create a custom

**Advanced NPC Strategy**

### Debugging and Testing

The program includes a debug output option that provides valuable information about the AI's decision-making process. This feature can be helpful for testing and understanding the program's internal logic.

## Extensions

While the core functionality of the Blackjack Game Program is complete, several extensions can be considered:

1. \*\*Machine Learning AI:\*\* Implementing a machine learning-based AI that learns and adapts its strategy over time could provide a more challenging and dynamic opponent for players.

2. \*\*Graphics and User Interface:\*\* Enhancing the program with a graphical user interface (GUI) and visual representations of the game could improve the overall user experience.

3. \*\*Multiplayer Support:\*\* Adding support for multiplayer games, either locally or online, would allow users to play against each other and enhance the social aspect of the game.

## Conclusion

The \*\*Blackjack Game Program\*\* successfully implements the classic card game with customizable settings and various AI strategies. It demonstrates the application of core computer science concepts such as data structures and decision-making algorithms. The program provides an engaging and interactive gaming experience while maintaining fairness and balance in gameplay. With potential extensions for further development, this program serves as a solid foundation for future enhancements and exploration of advanced AI strategies.