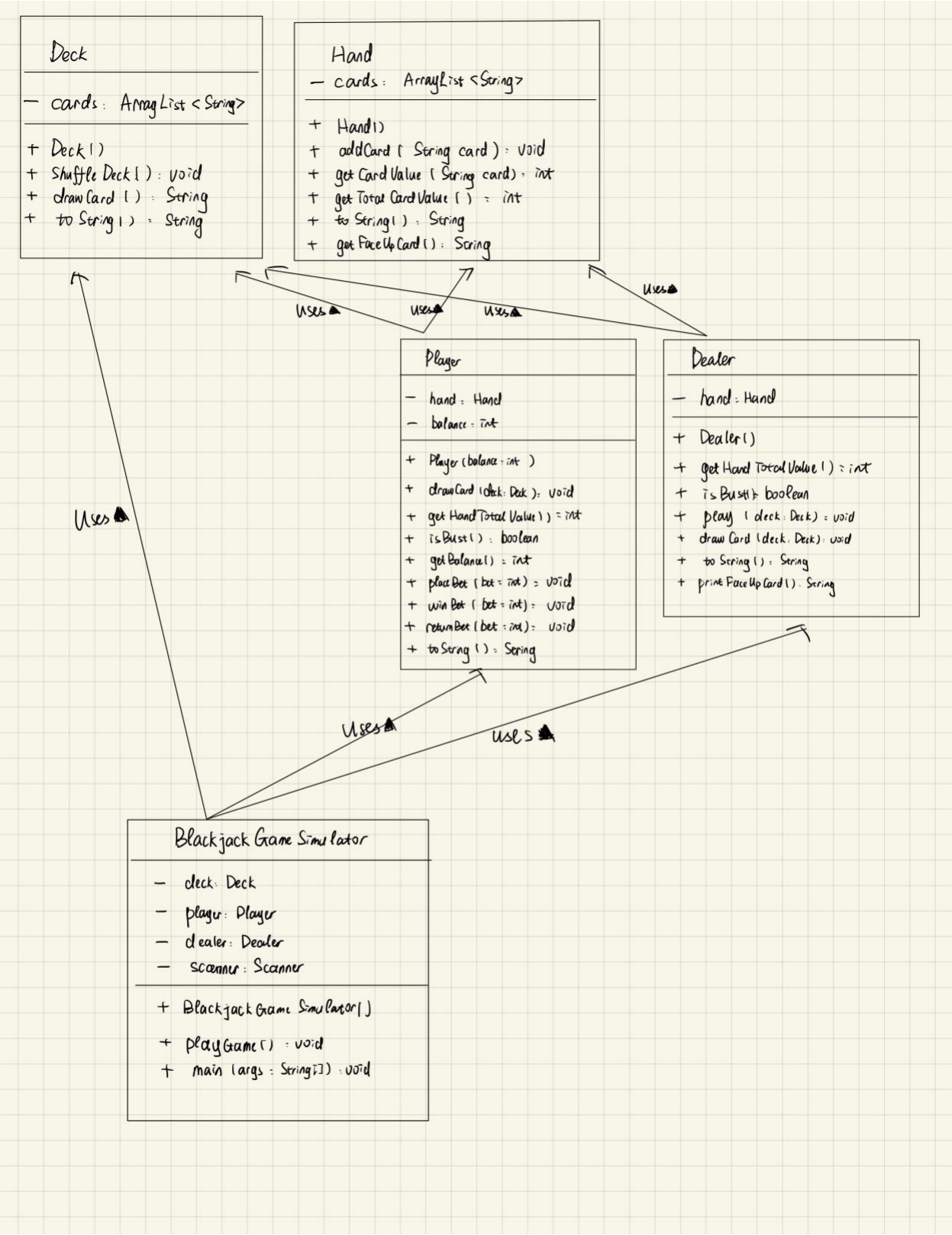
Blackjack Card Game Simulation

Yvonne(Ziyu) Lin

Whiting School of Engineering, Johns Hopkins University

EN.605.201.81.FA24

UML Diagram


Deck
- cards: ArrayList<String>
+ Deck()
+ shuffleDeck(): void
+ drawCard(): String
+ toString(): String
Hand
- cards: ArrayList<String>
+ Hand()
+ addCard(String card): void
+ getCardValue(String card): int
+ getTotalCardValue(): int
+ toString(): String
+ getFaceUpCard(): String
Player
- hand: Hand
- balance: int
+ Player(balance: int)
+ drawCard(deck: Deck): void
+ getHandTotalValue(): int
+ isBust(): boolean
+ getBalance(): int
+ placeBet(bet: int): void
+ winBet(bet: int): void
+ returnBet(bet: int): void
+ toString(): String
Dealer
- hand: Hand
+ Dealer()
+ getHandTotalValue(): int
+ isBust(): boolean
+ play(deck: Deck): void
+ drawCard(deck: Deck): void
+ toString(): String
+ printFaceUpCard(): String
BlackjackGameSimulator
- deck: Deck
- player: Player
- dealer: Dealer
- scanner: Scanner
+ BlackjackGameSimulator()
+ playGame(): void
+ main(args: String[]): void
Uses
Uses
Uses
Uses
Uses
Uses
Uses

Introduction

The blackjack game is a widely popular casino game in which players compete against a dealer to get as close as possible to a score of 21 without exceeding it. This project simulates a blackjack game using Java, with a focus on object-oriented programming.

Program Organization

The program is designed using an object-oriented programming approach. It consists of the following classes: Deck, Hand, Player, Dealer, and BlackjackGameSimulator. These classes represent the main components of the blackjack program and its relevant behaviors or methods, providing modularity and separation.

Deck is a class that is used to represent a deck of cards. In the blackjack game, we have cards from 2 to 10, Jack, Queen, King, and Ace in four suits: diamonds, hearts, clubs, and spades. In the deck class, one deck has 52 shuffled cards and a card will be represented as, for example, “9 of Diamonds."

The hand class represents the collection of cards held by a player or a dealer in the game. In the hand class, it allows the player or the dealer to draw a card and add this card to the hand. The hand class includes a method to turn one card into its respective point value. It also has a method to calculate the total value of the hand, which takes account of the ace situation. Ace typically stands for 11, but if the hand total goes above 21 and an Ace is one of the cards, the Ace can revert to a point value of 1.

The player class simulates the behaviors of a player in the blackjack game. A player has a hand of cards and can perform actions such as drawing cards and calculating the total value of the hand. The class also manages the betting behavior, allowing the player to place bets, win or lose bets, and return bets if it is a tie. Similarly, the dealer class simulates the behaviors of the

dealer in the game. The dealer follows specific rules: during the dealer's turn, cards must be drawn until the total value reaches 17 or higher. The separation of the dealer class and the player class helps to simulate different behaviors of the dealer and the player, providing the program with modularity.

BlackjackGameSimulator is the class that manages the flow of the game. It coordinates the player, the dealer and the deck, handling key decisions such as when to hit, stay and calculate outcomes and bets.

Other Considerations

Initially, I considered using an array instead of an ArrayList. However, I found that ArrayList provides methods like shuffle and remove, which we could take advantage of. I also started with a single Player class to represent both the player and the dealer, but this approach made it difficult to differentiate their actions. Separating them into two distinct classes allowed for a cleaner code structure and better modularity. This design not only improved readability but also simplified the implementation of unique behaviors for each role.

Lessons Learned

This program focuses on object-oriented programming. By breaking the game into its components and organizing them into different modules I had a better understanding of separations and modularity and gained experience in object-oriented programming processes.

Conclusion

In this project, it is a good practice to apply the knowledge we learned to real-world blackjack game situations and to carefully construct the correct object-oriented processes.

References

Purdue Online Writing Lab. (n.d.). *APA sample paper*. Purdue University. https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_formatting_and_style_guide/apa_sample_paper.html

Scribbr. (n.d.). *APA format for academic papers and essays*. https://www.scribbr.com/apa-style/format/

The Venetian Resort Las Vegas. (n.d.). *How to play blackjack*. https://www.venetianlasvegas.com/resort/casino/table-games/how-to-play-blackjack.html

Bicycle Playing Cards. (n.d.). *How to play blackjack*. https://bicyclecards.com/how-to-play/blackjack/