Documentation for BlackJackGameSimulator.java:

a program simulating the Blackjack game

Zixuan Yu

Johns Hopkins University

[605.201] Introduction to Programming Using Java

Mini Project 2

Dr. Sidney I. Rubey.

Oct.25, 2022

Documentation for BlackJackGameSimulator.java: a program simulating Blackjack game

General program design

This program simulates a Blackjack Game which consists of a UserPlayer Object and a Dealer object, a continuePlay variable (Boolean value, true if the user choose to continue play), and an endRound variable (Boolean value, true if the winner of this round has been determined). The two players will have a hand of cards each round, named userHand, dealerHand, respectively. The UserPlayer also has budget (amount of money he has) and bid (bid for each round) attributes. UserPlayer's methods include the get and set methods of budget, bid, and userHand, respectively; winBid(), loseBid() which will calculate the budget after each round end; noBudget() method will check and return a Boolean value indicating if the user has run out of his budget. The dealer only has the get/set dealerHand methods. UserHand and DealerHand are subclasses of Hand which include an Arraylist of Card objects. Hand and its subclasses have the toString method which has a Boolean value (displayAll) as argument: when displayAll equals true, the cards in Hand and the sum will be displayed; when displayAll is set to false in DealerHand, only the first card will be displayed, otherwise, the program will display a message informing that this is only for the DealerHand subclass. Hand's other methods include getcardSum() (return the sum of cards, Ace could be either 1 or 11), countAce() (returns the number of Ace in Hand), exceed21() (returns a Boolean value, true if sum greater than 21), winAt21 (returns a Boolean value, true if sum equals 21). The cards are randomly drawn from the Deck, which has the shuffle(), draw(), and toString (return card's suit and face) methods. Each card has the face attribute (value used when calculating Hand sum), the suit attribute, and the corresponding get and set methods. Face and Suit are both enumeration classes.

A UserPlayer and a Dealer are initialized at the beginning of the game. The user first enters his budget, then the game begins. At the beginning of each round, the UserHand of the

UserPlayer and the DealerHand of the Dealer are initialized, and Deck is shuffled. The user is prompted for a bid, then he is dealt two cards. If the sum is 21, he will win and this round will end. If not, the dealer is dealt two cards, and the user will be asked to hit (get one more card each time) until he chooses to stay (no more new card). If the user's sum does not exceed 21 (blasts), the dealer begins his round. The dealer will hit until his sum is 17 or more. If the dealer does not bust or reach 21, the program will compare the two sums and the user will lose/win his bid. If it is a draw, the user's budget will remain unchanged. If either of the two players reaches 21 at any point, the other player will lose, if either one blasts, the other player will win. Then this round will end, and the user will be asked to start a new round or quit until the user has lost all his money (the game will quit).

Possible Alternative Approaches

It would be better if I can write more methods in the BlackJackGameSimulator. For example, I could have developed a method named check endRound () to examine if there is a determined winner instead of using a variable to indicate the round status. Second, I could add methods like runUserRound / runDealerRound which will mimic the user/dealer's turn. It would be better to practice a more object-oriented way of thinking and use that in the project. As a novice, this is my limitation.

What I learned from this project and what I would do differently

I learned how to use the Objected-Oriented way to simulate a Blackjack game. I practiced how to realize inheritance and polymorphism in this project. A GUI would be more enjoyable than using the command line. I also want to create a .jar file so users do not need to compile the file by themselves. Finally, I want to create a better Javadoc next time. However, due to the limitations of time and my limited programming skills, I did not realize that.