BlackJack game in Java

BlackJack game in Java

Jeffrey Wan

John’s Hopkins University

BlackJack game in Java

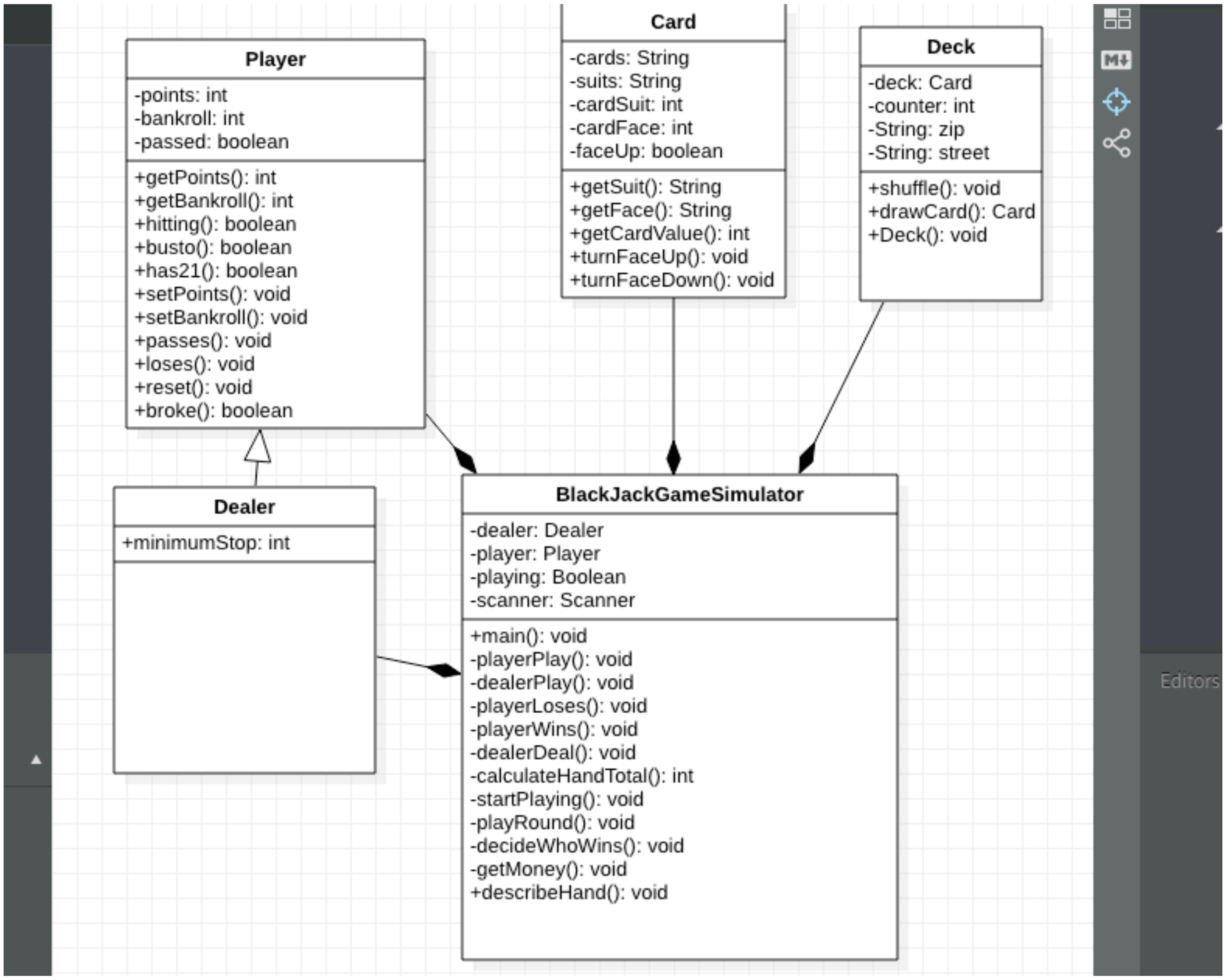
This program is a blackjack game simulation. It was designed with 5 major classes: a BlackJackGameSimulator class, a Player class, a Dealer class, a Deck class, and a Card class. The BlackJackGameSimulator class is used to manage the game play of both the dealer and the player, provide instructions, allow the player to bet and buy a bankroll, and repeat play. Player and Dealer class hold state of both the players that allow for certain actions to occur in the game. The Dealer is a subclass of Player with a different condition for hitting and staying. The Deck is composed of 52 cards and is held in an array. The Deck also is responsible for shuffling and drawing to provide a simulated perception of luck. When the program is run, a series of questions and actions can occur and actions are provided by console input. The game concludes when either the Player or Dealer has busted (gone over 21), when the dealer hits 21, or when the player has more points than the dealer.

I decided to design a deck of cards which was fun to do. The deck is composed to 52 cards and is held in a ListArray which allows for dynamic appending. The deck also implements shuffle and draw methods to better simulate a real game of blackjack. The constructor is key here; it creates an array of 52 unique cards using a nested loop with each loop representing the faceValue and suit.

The Card class creates objects that represent each card in the deck by suit and faceValue. Inside the class is an array of cardValues and suits. The constructor of the Card holds a faceValue and a suit and it is the Deck's responsibility to initiate unque values for each card, not the Card class'.

The BlackJackGameSimulation class is responsible for composing all of these classes in order to run the game. It first sets the player's bankroll and sets a betting amount which will be used to subtract from the bankroll in the event of a loss, and add to the bankroll in the event of a win. It also runs the logic responsible for creating hands for the player and the dealer. The hand is a ListArray of cards that can be used to calculate the point total. I considered creating an entirely separate class for the Hand, but decided to leave it in the BlackJackGameSimulation simply because I didn't have enough time to refactor. But I think the hand is perfect as a stateful, encapsulated. object that can be reset after every play and is unique for each player and dealer. The BlackJackGameSimulation class also is responsible for retrying the game if the player wants to bet again or replenish their bankroll. I considered extracting out the monetary parts of the game into a Banking class and using static methods in that banking class to add to the player's bankroll or retain betting amounts. Since there needs to only be a single Bank, I think a Banking class singleton would have been sufficient to operate these decisions. Again, I didn't have the time to refactor these choices.

I learned a lot about class organization, and separation of concerns during this project. This was a longer in length and more complex program than the tortoise and the hare project and I certainly felt the pain of not encapsulating certain pieces of logic that were then inappropriately manipulated by other parts of the program. I learned some lessons the hard way this time and will remember them in the future.



UML DIAGRAM OF BLACKJACK BY JEFFREY WAN