**BlackjackGameSimulator Design**

## I. General Program Design

This project involves writing a program to simulate a blackjack game. A player is provided with a sum of money with which to play with the dealer. A player can place a bet between $0 and the total bankroll that the player has. The objective of the game is to get as close to 21 points as possible without exceeding 21 points.

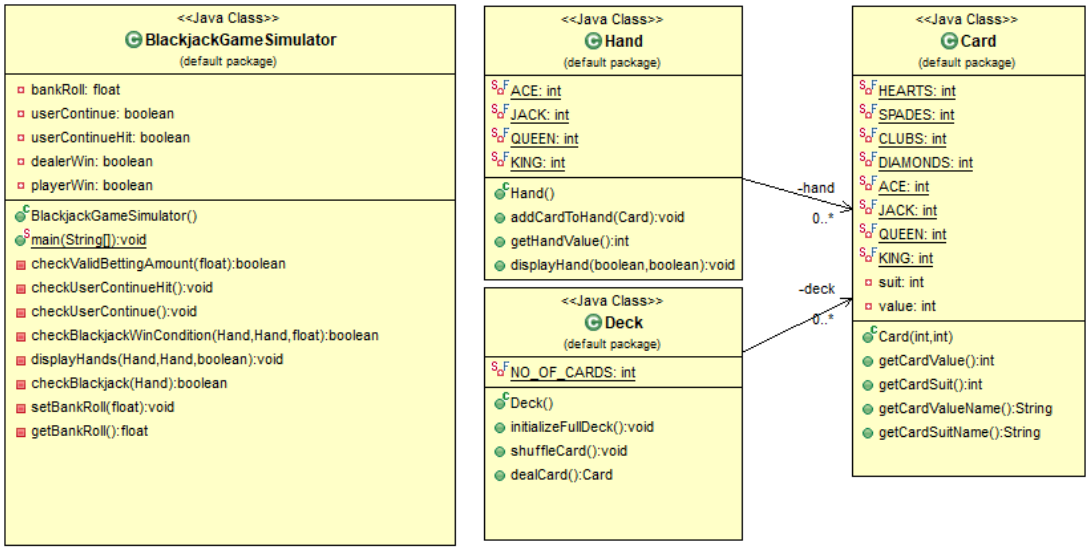
*Game Specifications:*

- A player is dealt two cards face up. If the point total is exactly 21 the player wins immediately.
- A player may decide to continue with "hits" or stop getting cards from dealers via "stay". If the player decides to stop "hitting", then the value of cards on the player's hand is his points.
- The dealer will then proceed to play the game. If the dealer gets to 21 then he immediately wins the game. The dealer must continue the game until his point is at least 17.
- If the dealer point is over 17 but not 21, compare the dealer's hand value versus the player's hand value. The one who has higher point value will win the game. If the player wins, his bankroll will be added and vice versa.
- The game will then ask whether the user wants to continue playing this game. The game will terminate when the player's bank roll is $0.

*Derived Implementation & Conceived Display (Self Interpretation)*

- The game will print out the dealer's hand initially at the start of the game (i.e. 3 of Hearts and a face down card) and print out the player's hand (i.e. King of Hearts and 2 of Clubs). Once it is the dealer's turn, the console will print out the full dealer's hand (previous example was 3 of Hears and 5 of Diamonds).

## II. Class Design




- There will be 4 classes in the program, which is BlackjackGameSimulator class, Hand class, Deck class and Card class. The Hand class and the Deck class are composed of Cards instances (composition relationship). Thus, there will be private ArrayList<Card> *hand* and *deck*, respectively in the Hand and Deck class.

**Card Class**

- Card constructor will create the card based on a suit value as well as a numerical value. The Card class contains standard getter functions to retrieve the card's value and suit numerically or by name (returning a String).

**Hand Class**

- Hand constructor will create a hand. The addCardToHand(Card) function will add a Card to the private ArrayList<Card> *hand* variable.
- The getHandValue() function will return the value of the hands of the player or the dealer. The displayHand function will handle the printout/display of *hand* (if conditions are first round and player's turn, then will only print 1 card; if it is dealer's turn, then display the full dealer hands).

**Deck Class**

- Contains the private ArrayList<Card> *deck* variable which contains the 52 decks in the game. The shuffleCard() function will shuffle the deck of the cards from the initially ordered positions. The dealCard() function will deal the card, at the same time remove that card from the deck.

**BlackjackGameSimulator Class**

- The main function containing the major logic for the game which will continue to run while player's bank roll > 0 and the user wants to continue to play the game == True). It will then create a dealer's hand and a player's Hand. The main function will then prompt for the amount of money to be bet this round. The main function will then monitor if the player's hand is blackjack by calling the function checkBlackjackWinCondition(Hand, Hand, float). If the player has blackjack hand, the player will win this round and extra money will be added to his bank roll based on the betting amount. If not blackjack, record the player's hand value until the player doesn't want to "hit" anymore. Switch context to the dealer's turn. The dealer will continue to hit until his hand satisfies the condition of 17 <= dealer's hand value. The main function will again check for blackjack conditions by calling checkBlackjackWinCondition(Hand, Hand, float). If the dealer has a blackjack hand, then the dealer wins the game. If not, we will have to compare the value of the dealer's hand and the player's hand and the winner is the one with the bigger hand value. The main function will prompt whether the user wants to continue the game after a round ends.

## III. Alternative Approach

Alternative approach would be:

1) One alternative approach was to forgo completely the creation of the Card class. Instead, a numerical value will be assigned to a Card (i.e. Ace of Heart = 1, Ace of Clubs = 2, Ace of Diamonds = 3, Ace of Spades = 4, 2 of Hearts = 5, 2 of Clubs = 6, etc.). In this manner, the Deck class and the Hand class will hold private integer arrays which contain the numerical representation/convention of each the card and handle the translation of numerical value of (i.e. 6) to a card value (2 of Clubs). This approach was rejected because it makes the program very unclear. A representation of a Card instance is much clearer than a numerical value and would be more valuable in the debugging process where the engineers must manually figure out what the card value and suit value that numerical value represents.