# CIST2361 Software Design Document

Chris Grady

Blackjack Emulator

4/3/2016

# Introduction

This program will emulate the card game Blackjack, also known as Twenty-One. There are two modes of the game: Human vs Human (Multiple players against the dealer) and Human vs Computer (Single user playing against the dealer). In both modes, an option of choosing difficulty is provided. There is also a sign-in screen, which allows user information to be stored and referenced throughout the game. It will contain many functions, but the most high-level view would include the dealer dealing out two cards to each player, taking bets from each player, allowing each player to choose whether or not to “hit” based on their initial hand (also including an option to double-down, split hands, and take a side-bet on the dealer), keeping a tally of how many points each player has, determining winners (if player hits 21; if not, did he exceed 21 or is he still the closest one to reaching 21, etc.), and, most obviously, displaying the cards to the end-users. As I follow the top-down approach, these high-level functions will become much more specific. The users/players are always playing against the dealer, but not against each other. Also, more than one player can win as long as the dealer doesn’t. For example, if two players have a score closer to 21 than the dealer: they would both win.

**Definitions:**

* **Betting:** Function called before each round that allows each player to place a bet of any amount (As long as the number is above 0)
* **Hit:** Function meaning that the player would like to be dealt another card
* **Stand:** Function to do nothing, or take no more cards
* **Double-Down:** Function for a player to double their initial bet (only under the condition that they have to stand after their next dealt card)
* **Split:** In the event of a player getting two cards of the same value, a split hand is a function that allows each card to be separated into its own unique hand (Player must also match the initial bet for the second hand), and another card is dealt to each hand. The two hands are considered completely separate in terms of winning and losing.
* **Insurance:** Side-Bet function available when the dealer’s “side-up” card is an Ace. The bet is that the dealer will get Blackjack after standing on his next turn. The return for winning this bet is 2:1 (If you bet $5 and the dealer gets blackjack, you get $10)
* **Bust:** When a player score exceeds 21; results in automatic loss
* **Soft Hand:** If player holds an ace (11), he cannot bust on next card dealt (11 + any card will be less or equal to 21)
* **Hard Hand:** Mathematically possible to bust on the next card dealt
* Each card will hold a value in points (e.g. Face cards are worth 10; Ace’s as either 1 or 11 depending on situation, all other cards are worth their own number In points)

# Use Cases

**Use Case: Typical**

1. The program displays game instructions and asks how many users will be playing
2. User enters a number
3. The program asks for player(s) name(s)
4. User(s) enters name(s)
5. Console displays initial cards dealt to player(s), also shows total points for each hand, then asks user to either Hit or Stay
6. User enters choice (Hit-Another card is dealt or Stay-Stick with current hand; dealer hits automatically in most cases)
7. Repeat steps 5 and 6 until somebody hits 21 or busts (If someone busts, the player closest to 21 wins)
8. User is prompted to enter whether or not they would like to play again
9. User enters choice
10. If user plays again, steps 5-9 again; if not, game quits

**Use Case: Error (Invalid number of players)**

1. The program displays game instructions and asks how many users will be playing
2. User enters a number that is greater than maximum number of players (7)
3. Error message is generated; User is prompted to try again

**Use Case: Error (Empty username)**

1. The program displays game instructions and asks how many users will be playing
2. User enters a valid number
3. The program asks for player(s) name(s)
4. User(s) enters empty field for name(s)
5. Error message is generated; User is prompted to try again

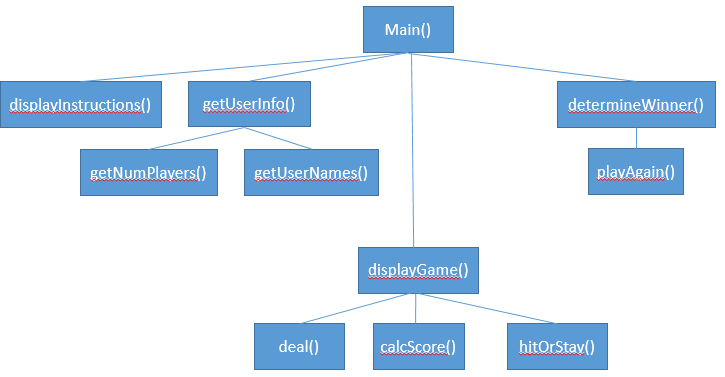
**Use Case: Error (Invalid choice when prompted to Hit or Stay)**

1. The program displays game instructions and asks how many users will be playing
2. User enters a number correctly
3. The program asks for player(s) name(s)
4. User(s) enters name(s) correctly
5. Console displays initial cards dealt to player(s), also shows total points for each hand, then asks user to either Hit or Stay
6. User enters invalid choice
7. Error message is generated; User is prompted to try again

**Use Case: Error (Invalid choice when asks to play again)**

1. The program displays game instructions and asks how many users will be playing
2. User enters a number correctly
3. The program asks for player(s) name(s)
4. User(s) enters name(s) correctly
5. Console displays initial cards dealt to player(s), also shows total points for each hand, then asks user to either Hit or Stay
6. User enters choice correctly (Hit-Another card is dealt or Stay-Stick with current hand; dealer hits automatically in most cases)
7. Repeat steps 5 and 6 until somebody hits 21 or busts (If someone busts, the player closest to 21 wins)
8. User is prompted to enter whether or not they would like to play again
9. User enters invalid choice
10. Error message is generated; User is prompted to try again

# Design Overview:



# System Tasks Description

## Function: main()

*Bool playAgain = true*

*While playAgain == true*

*{*

*Call displayInstructions()*

*Call getUserInfo()*

*Call DisplayGame()*

*Call Deal(userScore)*

*Display “Would you like to hit or stay?”*

*Input userMove*

*Call hitOrStay(userMove, userScore)*

*Call calcScore(userScore)*

*Call determineWinner(userScore, dealerScore)*

*Call playAgain(playAgain)*

*}*

## Function: displayInstructions()

*Display “Welcome to Blackjack!”*

*Display “Below are the rules:”*

*Display “Blah, Blah Blah, Detailed rules will be displayed here”*

## Function: getUserInfo()

*Call getNumPlayers()*

*Call getUserNames()*

## Function: getNumPlayers()

*Display “How many players?*

*Input numPlayers*

*If numPlayers > maxNumPlayers or numPlayers <= 0 then*

*Display “Error: Too many or too few players”*

*Input numPlayers*

*End If*

## Function: getUserNames()

*Display “Enter player’s name:”*

*Input userName*

## Function: displayGame()

*Displays virtual card game table with formatted output representing cards and values*

## Function: deal(userScore)

*Add random card from deck class to hand*

*Call calcScore(userScore)*

## Function: calcScore(userScore)

*userScore += valueOfCard*

## Function: hitOrStay(userMove, userScore)

*If userMove == “hit” then*

*deal(userScore)*

*else then*

*break*

## Function: determineWinner(userScore, dealerScore)

*If userScore == 21 then*

*Display “You Win!”*

*Else if userScore > 21 then*

*Display “BUST! Sorry, you lose”*

*Else if closerTo(userScore, dealerScore) then*

*Display “Your score was closer to 21 than the dealer, you win!”*

## Function: playAgain(playAgain)

*Display “Would you like to play again? (Enter Y or N)”*

*Input playAgain*

If playAgain == “Y” then

Return true

Else then

Return false