CS 485: Assignment 7 — 90 points: Execution: 36 pts, Coding Standards: 18 pts, Design 36 points

Blackjack ♠♣♥♦ — DUE: UML Diagram, Monday Apr 22, 3:30pm **(Hard Copy)**

DUE: Design Presentation: Monday Apr 29 3: 3:30pm

DUE: Completed Solution/Demo, Monday, May 6, 3:30 pm

Goals for this assignment

1. Design a BlackJack[1] game with both an SDL and TextUI front end.
2. A well designed set of Cards[2, 3] and Deck are necessary
3. Use the MVP architecture pattern to build your application.
4. Apply the design principles studied and practiced in this course.

The Game of Blackjack

► Blackjack is a simple card game where the player plays against the Dealer (or House). The goal of Blackjack is for a player to put together a hand of cards with a point value greater than the hand the Dealer puts together. The total points in a hand also must not go over 21 points. Points are assigned based on the number on the card (2-10) with 10 points for Jack, Queen, or King. An Ace is worth either 1 or 11 points (whichever point value gives the hand the highest point total without going over 21). An Ace can be 11 at one point in time and become a 1 later on. For instance, 3H, AH, 10D, 7S is Blackjack. The AH went from 11 to 1 as the hand progressed.

The game is played with at least one ordinary deck of 52 playing cards. More commonly, though, multiple decks (as many as 6) are shuffled together and used at the same time.

A Round of Blackjack runs as follows:

- Players bet on the round. Legal bets are $1 to the total money in the Player's bank.
- The Dealer gives each player, in order, one card face down. The Dealer receives a card, also face down, last.
- The Dealer gives each player, in order, one card face up. The Dealer receives a card, also face up, last.
- Each player gets a turn to build on their hand. At this point, the player can see all of his or her cards but only the Dealer's face up card. The player can take the following **actions**:
    - The player can **draw**: add one more face up card
    - The player can **stay**: this ends their turn and their hand is set
    - The player can **split**: if the first two cards a player receives are the same denomination, the player has the option to split the cards into two hands. Each hand is played independently from this point. The original bet is now copied to each hand. The player is effectively doubling the bet that is in play. If the player loses both hands this could cause the player to have a negative bank. This is allowed but once the player has a negative bank they are considered **broke** and are kicked out of the game.
    - Splitting is only allowed once per round per player and only on the first action from the player.
    - Continue drawing until the player: goes bust (accumulates over 21 points), gets Blackjack (scores exactly 21 points), or chooses to **stay**. If a player goes bust they lose their bet immediately. If the player hits Blackjack they gain their bet immediately. For Blackjack, the player wins 150% of their bet. Once the player settles their bet they are considered **settled**. At this point, only players going bust or getting Blackjack are settled. Players can only draw to the point they have 10 cards in their hand.
- Play continues for each player.
- The Dealer plays last. The Dealer does not bet and must follow the Dealer AI rules as described below.

[1] `https://web.stanford.edu/class/cs107l/handouts/06-Blackjack-Assignment.pdf`

[2] http://zeus.cs.pacificu.edu/ryand/cs250/2017/Assignments/04Card.pdf

[3] https://github.com/notpeter/Vector-Playing-Cards

- Once the Dealer's play ends (either through going bust, gaining Blackjack, or staying), the **unsettled** players settle up their bets.
- If the Dealer goes bust, any unsettled players win their bet. If the Dealer gains Blackjack, any unsettled players lose their bet. If the Dealer stays, any unsettled player with more points than the Dealer wins their bet, any unsettled player with fewer points than the Dealer loses their bets. Any unsettled player with exactly the same number of points as the Dealer is considered a Push and does not gain or lose money.
- At this point, players can bet again to start a new round or quit.
    - If before starting a new round, you have fewer than 10*(number of players + 1) [the +1 is for the Dealer] cards remaining in the deck, you must reshuffle all the cards and start over. The Card Counting AI must be notified that any state that has been built up is now invalid.

Your Requirements

► You must produce a well designed object oriented solution to Blackjack. Further, you must use the Model-View-Presenter architecture pattern to build both an SDLApp2 and TextUI front end using the same Model. You may choose to have one Presenter or two.

► Choose one group member's Money, Currency, and associated Exceptions to use in your project. Each player's bank must be in USD. All betting is in USD.

► You must allow the user to choose to play with 1-6 decks.

► Your game must support 1-5 players (plus the Dealer). Each player gets to specify a name and an initial bank amount. Any number of players may be Computer players. Each computer player can be set to any of the AIs described below. Each computer player is given a name and initial bank as well, either randomly or via human input.

► You must produce a UML diagram of your design. This must be the design for the entire system including both front ends.

► You must produce at least two AIs.

1. Required: The **Dealer AI** uses only the player's own hand to make a decision. Draw while the hand is worth 16 or fewer points. Stay when the hand is worth 17 or more points. The Dealer never splits. The Dealer never bets but if another computer player uses the Dealer AI the bet should be 1/5 of the player's total bank.
2. Choice: You may produce an AI that uses the player's own hand and some **table information** (table information is any information available by looking at the current table: up cards, number of cards on the table, etc) from outside of the player's hand. This AI is focused on how to decide to draw/split/stay. This AI must have access to all the face up cards on the table. For instance, the AI might draw if the Dealer has an up card worth at least (the player's total - 7). For instance, if the dealer is showing a 7 and the player currently has a total of 13, the player will draw. Otherwise the player will stay. Betting happens before any cards are shown so the betting strategy does not need to rely on any table information.

   OR

   You may also produce an AI that uses **past table information** to make a decision. This may be an algorithm for card counting[4]. This past information influences betting more than the draw/split/stay decision. You may choose any draw/split/stay decision algorithm.

[4] https://en.wikipedia.org/wiki/Card_counting

BONUS AI: Casinos often supply free or reduced-price drinks to players at the table. Typically, this leads to erratic behavior (bet it all!) that benefits the House. Build an AI that simulates a player slowly receiving free drinks between rounds and playing in a more and more erratic fashion.

Suggestions:

Build the Model first.

Identify the events that need to be generated by the UI and how those events will filter down to the Model via the Presenter. Settle on the same set of events that can be generated from both the TextUI and SDLApp2 front end. Build sequence diagrams to understand how events and data flow through the system. Be sure to pay special attention to how Human and Computer players interact with the Model. Remember, only Human players generate UI events.

You will need a method to shuffle your cards. You will also need a method to stack the deck so you can easily test a player going bust, hitting Blackjack, beating the Dealer, and losing to the Dealer.

► Name your Visual Studio solution: CS485_BlackJack_Group#.

Name your Visual Studio projects: BlackJack_Model, BlackJack_SDL, BlackJack_TextUI. Your Visual Studio solution must include and use the SDLApp2 and TextUI projects from TicTacToe-MVP. You are free to use any example code distributed this semester. Just be sure to comment the code with a citation of its origin.

You must use Git and share your repository with the instructor.

Follow the coding standards posted on the class website. Be sure to use VLD to check for memory leaks.

► Come see me early and often with design issues. At least two group members must be present to ask me questions.

► Thanks to Doug Ryan for help with this assignment.

► Thanks to the members of the CS 485 Spring 2017 course who had to deal with this assignment first!

Design Presentations:

Each group will have a max of **16 minutes** to describe their design. Group members must take turns speaking for no more than 4 consecutive minutes.

Focus on the main classes of the Model and their interactions, the Events generated by the UI and how those events interact with the Model, and how you have applied knowledge from previous projects in this course. Describe how your design has changed since April 22th. Display at least one sequence diagram.

Expect 5 minutes of questions. Your presentation must be practiced and take from 14-16 minutes.