Tristan Biggs

Doctor Michael Findler

CSC110AB

27 April 2025

Final Project Design Document: Blackjack

# COLOR KEY

This color key did not work out as well as I had hoped. From what I can tell, dark mode now simply draws a new screen using the old as a reference and performs some kind of operations on the detected colors to make THEM dark mode too – So these colors are actually completely useless for anyone in light mode. I will leave it here to serve as an example. – TB 5/4

Completed

Needs Action

Code



Parent / Super Classes : Parent Objects : Objects

Child / Sub Classes : Child Objects

* Always used to reference object if specification is required by referencing code
* Always formatted as Parent.Child (Parent Color(Parent), Operator Color( . ), Child Color (Child)) if format is required, such as above scenario

Fields

Constructors

* Always refers to a Object or Object, exclusively that which is it designed to create

Methods

Operators, Keywords, conventional solution Shorthand

* All syntax associated specifically with the above are also this color

User Experience

Technical

Design

# Project Objectives

## REQUIRED

1. Description of the problem/goal/task your project is intended to accomplish. (1 point)
2. Make a plan for the project. Give the list of the steps your program takes. This will be your plan. Think of it as a way to write out what you need to tell the computer to do. This is the most important part of programming. [Plan should be more detailed than original plan.] (1 point)

1. I need to see your working project coded in Java. Must run in VS Code. (1 point)

Presentation

1. Test your program. Include test cases you used to test your project, so someone else could do the same. (1 point)
2. Include Documentation. Documentation explains the specific features of the project informs how one can work with it, helps on to understand the project's functionality. Be sure to add comments throughout your code. This helps you and anyone that looks at your code understand your thought process for doing what you did. Think of documentation as a user manual and system manual combined. (1 point)
3. Include future project enhancements. What changes would you make to your program next in the future. What expectations of uses do you foresee? How would these expectations determine your next updates? (1 point)

Design Documentation – Highlight unimplemented features as future features

1. Develop a presentation (5 minute maximum) that could be used to explain your project to others. (1 point)

Complete

1. Record a presentation (5 minute maximum) that explains your project. (1 point)

Complete

1. Develop Java program(s) that use Swing GUI. (1  point)

Complete

All Swing elements -> SortCards.java



## MINIMUM 6 FROM THE FOLLOWING

1. Write Java program(s) that process user input. (1 point)



Private Static String UserName = GameArgs.nextLine;

Several Swing ActionListeners attached to buttons that are to be used by Player to pass input to program

1. Write Java program(s) that display formatted output. (1 point)

No Thank You

1. Write Java program(s) that utilize branching control structures. (1 point)

Idk how you make any Java program without them tbh

1. Write Java program(s) that utilize loops. (1 point)

Yuuuuup

1. Develop Java program(s) that use various data type

Yarp

1. Develop Java program(s) that use advanced features such as arrays, ArrayList, Queues, or Stacks... (1 point) [Can be one of these or multiple]

Yuuuuuuuuup

1. Develop Java program(s) that use advanced features defining and using methods. (1 point)

Hmmm I planned on it but I don’t think any of those features are actually going to be implimented

1. Develop Java program(s) that use Object-Oriented programming (OOP) for custom classes.(1 point)

HAAAh, man. Yea I did that. Table, Bank, Dealer, Player, Deck, Hand.

1. Develop Java program(s) that use Exception Handling. (1 point)

Yahar

# Blackjack

## Assignments

Tristan Biggs

* All of it.

## Objectives

* Create a program that utilizes objects designed after playing cards from a typical 52-card deck to virtually play Blackjack at a one-person ‘table’ against the ‘dealer’, or program.
* Player will be dealt two cards whose ranks will be added together as the current ‘score’ of the player. Player must as get as close, or equal to, a total value of 21 from the cards in their ‘hand’, which they will choose to either add to (‘hit’) or stay, and keep their current score.
* The dealer will do the same, although the dealer’s hand will be hidden from the player.
* The game ends when any of the following occurs:
  + The dealer ‘busts’, or has a rank sum greater than 21
  + The player ‘busts’, or has a rank sum greater than 21
  + Both players have decided to keep their current score
* Determining a winner is done by comparing the rank sum of each player’s hand and evaluating which is closest to 21 without going over.
* Player will then be prompted to choose whether to continue playing or stand.

# Project Outline

1. Create typed drafts for:

~~STARTED 4/28 TB~~

## ~~Pseudo-Code Planning~~

~~STARTED 4/28 TB~~

~~(4/28 – TB):~~

### Summary Outline

5/4 TB

Comments on Design 0-1 (original design)  
  
The approach the first design, referred to as Design 0-1 only in name as the code was retired to serve as future implementations, was slow, methodical, and meticulously detail oriented (when I wasn’t raving mad from lack of sleep).

What I failed to realize was that no matter what pace I had chose that I had made a critical error in my Development Plan – I started with the top level classes and had intended on ENDING with card being the last non-UI or main class.

I am certain that had I made the decision to begin without my group, I would have no issues producing the originally intended program.   
  
In the original design, 11 classes were present. These classes can now be found in the code base under Future\Casino 0-0-0-1\. In the encapsulation update that was planned before Design 0-1 was scrapped, it was to be reduced to potentially 7, but during that process there were many issues with incorrect inheritance as I had not actually realized yet that (and am actually still not quite sure about) older versions of Java being still in use enough to be the top result on something like google when searching for information about methods and classes. This added another layer of difficulty, as in my carelessness I assumed it would work out alright as it had for every other assignment. So, when I was unable to reverse engineer the supplies swing code to expand and apply for the purposes of having the ability to run any, ANY kind of test, I realized it was time to swallow my pride.

At this point, I had made the choice to go back and to again the right way.   
I started with the card start that had been provided by Dr. Findler

## Technical Writing

STARTED 4/28 TB

COMPLETED 4/28 TB

##### Technical Writing Draft – MECHANICS

Play against a Digital Dealer in Blackjack

Beat the dealer’s hand without ‘**busting’** (going over 21)

Win the bet by beating the dealer’s hand – tally up the point value of your cards. Ranks 2 through 10 are valued at the same as their rank (2 is 2, etc.) **Jack**, **Queen**, and **King** are **10 points**. **Ace** is **1 or 11**, decided by the player. An **Ace** and a **Face card** (**Jack**, **Queen**, **King**) are worth a sum of **21 points**, and is known as a **Blackjack**! If Dealer does not have **Blackjack**, that means you’re a **Winner** and your payout will be multiplied by the highest value of Payout Multiplier

###### Each player makes a minimum bet

Don’t worry, no real money will be lost during this game! Try to see how much you can win.

###### The dealer gives a card to each player as well as themselves

Each player is dealt a face-up card, and the dealer deals themselves a face-down card. (!Since the number of cards in a 52 card deck that have a point value of ten is almost a third, it’s safe to assume that the dealer’s face down card has a value of 10. However, there’s no way to guarantee what the card is.!)

###### The dealer hands out a second card to every player

Each player receives a face-up card from the deck just as before, and the dealer this time deals themselves a face-up card. (!If the dealer’s face up card is 6 or fewer, they will likely bust. This is because the dealer is required to hit if their hand is 16 or less, and the probability of a sum of 21 is not in their favor in this scenario.!)

###### The player to the left of the dealer starts gameplay

###### Decide if you want to stay or hit

To ‘stay’ simply means you do not want the dealer to give you another card. (!You might want to stay if your card total is already quite close to 21 – 17 or 18, for example. The probability you’ll get an Ace, 2, or 3 is much lower than a 4 or higher.!)

To ‘hit’ means exactly the opposite – the dealer will deal you another card, which is added to the total among ranks in the cards in your hand.

###### Each player takes their turn in this manner until each has concluded their turn.

###### The dealer reveals their second card and winners are determined

The dealer flips their face-down card. The dealer has to play by different rules: If their total is 16 or less, they are required to hit. If the total is 17 or higher, they are required to stay. If the dealer gets a Blackjack, all players lose the round unless they also have a Blackjack – this results in the player ‘pushing’, which really just means that they get their bet back.

###### Bets are paid out and a new round begins.

Any winning player receives 1:1 winnings; for every 1 bet, 1 is rewarded. In the case of a Blackjack, the player receives 3:2 winnings; for every 2 bet, 3 are rewarded. The cards are then collected and another round begins.

##### Special Techniques

###### Splitting

If you are dealt two identical cards, or any cards with identical point values, you are given the option to ‘split’ your hand into two – playing the round with each hand as if they were individual players. In order to split, you must bet the same as the initial minimum buy-in. If you choose to split two Aces, you will only be allowed to hit each hand once. If one hand gets a Blackjack, it will not be considered as such for the payout but will allow you to push if the dealer has a Blackjack.

###### Doubling Down

Before the dealer gives you another card, you have the option to double your initial bet to ‘Double Down’ in exchange for your ability to hit. This means that your final score will be based on the initial two cards received, but winning on a Double Down rewards a significant payout. Doubling Down can be down on any hand that is lower than 21.

###### Surrendering

In the case that you are dealt an especially bad hand, you have the option to ‘surrender’, essentially folding to the dealer immediately in order to retain 50% of your initial bet.

Side-Betting/Insurance

If the Dealer’s face-up card is an Ace, you are allowed the option to place a Side-Bet that can be lesser than or equal to the original bet. The bet is that the dealer’s face-down card has a value of 10, giving them a Blackjack. If you are correct, your initial bet loses unless you have a Blackjack as well and your Side-Bet rewards a 2:1 payout. If you are wrong, the Side-Bet is lost.



## Design Draft

STARTED 4/30 – TB

COMPLETED 4/30 – TB

This remains here until I find a good place to put it. The header will allow you to collapse the entire thing with the small black arrow at the left of the String

### ~~GAME FLOW – USER EXPERIENCE~~

~~Contains some Code and Design elements~~

~~First Draft – Prebuild Design~~

#### ~~FIRST PROGRAM LAUNCH~~

1. User opens program which loads to Main Menu

User object initialization is hard coded and constants/instance variables are deep stored

* 1. If this is the first time the User has launched the Game:
     1. The Game instead loads First Launch Screen that will show them around the UI and give basic Game details
     2. User is prompted to enter their User Name

Static final String<

* + - 1. Represented in the UI by a label attached to User’s Player Avatar
    1. User is prompted to select a Player Avatar
       1. Represented in the UI while in the Main Menu and Game Menus by the selected Avatar positioned in the UI where visual noise is on average lowest for non-game windows
    2. User is given a User Bank that can hold chips

Bank UserBank = new Bank;

Create a file in local directory and write values of all fields owned by User

Executes at Launch, conclusion of every Table.Game, and with caller to exit program

* + - 1. Represented in the UI while in the Main Menu by
    1. User is given an amount of chips that will be used for Games

User static int UserBankBal is hardcoded in User Class

* + - 1. If the User Bank is ever reduced to 0, the Game will give User an amount identical to the amount given at first launch.

Set Private Boolean isBrokeBitch = true



* + 1. User leaves First Launch Screen and Game loads Main Menu

#### ~~ACTUAL PROGRAM LAUNCH~~

1. User selects “New Game”

#### ~~GAME CREATION~~

1. User leaves Main Menu and game loads Game Options Menu
2. User cannot select “Start Game” until the following options are chosen

Scanner GameArgs = new Scanner(System.in);

* 1. Desired Game to play

EventListener for ClickEvent on button for each button that represents a Game in the Game Type

Selection portion of the Game Creation Menu

* + 1. Represented in the UI by
    2. User Bank must contain at least as much as the minimum Buy-In of a Game in order to select it

Evaluated as condition of If statement at the top of Game Creation Loop

If User.isBrokeBitch == True;

Break;

* 1. Desired Number of computer players up to 7

MaxPlayers = Scanner.GameArgs(NextLine);

* 1. Low Stakes game or High Stakes game

EventListener for onClickEvent of Button on LowStakesButton

1. User selects “Start Game”, User leaves Game Options Menu and game loads Table

#### ~~GAME START (POPULATE)~~

1. User is given a Player to represent them at the Table

Player constructed by Table at initialization

* 1. Represented by User’s Avatar the is labelled with User’s Name

ui

1. Table is given a Dealer to control the Game Progression

Bank.Dealer BlackJackTheBlackjackDealer = new Table.Dealer;

1. Table is given the number of computer Players specified by Game Options

#### ~~GAME START (DECK)~~

1. Game gives Dealer a Score
   1. Represented in the UI by
      1. Invisible to Players until Dealer’s Face-Down Card is flipped Face-Up
   2. Score holds sum of Point Value of each Card in the Hand
2. Game gives each Player two Scores
   1. Represented in the UI by
      1. Score Two is not visible to Players unless Player’s Hand Two has a Card in it
   2. Score One holds sum of Point Value of each Card in the Hand One
      1. Score Two is identical
3. Dealer creates a Deck and shuffles it exactly 8 times for literally no reason (We are going to try to use random access (maybe, I don’t actually know what it does specifically yet))
   1. Represented in the UI by
4. Dealer is given a Hand to hold Cards in play
   1. Represented in the UI by
5. Each Player is given Two Hands to hold Cards in play
   1. Represented in the UI by
6. Each Player is given a Score to hold the sum of the point values among Cards in each their Hands respectively
   1. Represented in the UI by

#### ~~GAME START (BANK)~~

1. Bank gives Table:
   1. A Dealer Bank holding an amount of Professor Land FunBux that is small if Low Stakes and big if High Stakes
      1. Represented in the UI by an image of a bank with the value of Dealer Bank displayed on it labeled “House Balance”
   2. A Pot to hold all the Bets of Players for each Round
      1. Represented in the UI by an image of a Pot with the integer value of amount in Pot
      2. Winnings are paid out from the Pot
         1. If the Pot is unable to make a Payout, Table Bank adds a maximum value of chips to it, or the remainder of what it contains
2. Each Player is given a Balance to hold chips and User Bank is emptied into it
   1. Represented in the UI by
   2. The game is over when either:
      1. There is no Player that can pay minimum Buy-In because their Balance is zero
      2. Table Bank is unable to give Pot any chips to Pay out because its Balance is zero

#### ~~FIRST ROUND STARTING~~

1. Dealer starts the first round of Blackjack
2. Dealer takes the minimum Buy-In of the Table from each Player and puts it in the Pot

#### ~~ROUND FIRST DEAL~~

1. Dealer deals each Player one Card from the Deck to the Player’s Hand as well as themselves
   1. The Players are unable to see the Rank or Suit of Dealers “face-down” card
2. All Players are prompted to choose whether to Double Down
   1. Players that Double Down automatically choose to Stay on each of their Turns for the rest of the Round, but their Payout Multiplier for the Round is significantly increased

#### ~~ROUND SECOND DEAL~~

1. Dealer Deals each Player a second Card from Deck to the Player’s Hand as well as themselves
   1. The Dealer’s second Card is visible to Players
2. If Player’s Hand contains an Ace and a Card with a 10-Point Value, then they have a Blackjack and they end each of their turns immediately until the last Phase of the Round
   1. If Player wins with a Blackjack, Payout multiplier is set to highest value for that Player for the Round
   2. If Dealer has a Blackjack, then the Player does not Win or Lose but instead they Push
      1. When a Player Pushes, their original Bet is returned to them
3. Players are prompted to choose whether they would like to surrender (essentially fold until the next Round and receive 50% of the Buy-In back)
4. If a Player has two Cards of identical Rank values, they are Prompted to choose whether to Split
   1. If they do:
      1. Player must make an additional Bet the same value as the minimum Buy-In
      2. One Card from Hand One of the Player is moved to Hand Two
      3. Player is now Playing two Hands as if they are Two Players until the end of the Round
5. If the “Face-Up” Card in Dealer’s Hand has a Rank of Ace, all Players are prompted to choose if they want to make an Insurance bet

#### ~~ROUND TURN ORDER ( [PLAYER] )~~

This Phase repeats for each Player at Table, plus Dealer

1. Player whose Avatar representation is at the left-most position of the UI representation of Table goes first and Turn Order progresses in the same direction
2. Player of Turn is Prompted to choose an Action between Hit or Stay
   1. If the Player chooses to Hit:
      1. The Dealer Deals a Card from the Deck to the Hand of Player of Turn
         1. If the new sum of Score is greater than 21, then the Player has Bust and has Lost for the Round.
   2. If Player chooses to Stay:
      1. The Player’s Turn immediately ends

#### ~~ROUND DEALER TURN~~

1. The Dealer flips the Face-Down Card in their Hand and their Score is revealed.
   1. The Dealer takes their Turn differently than Players:
      1. If Dealer’s Score is less than or equal to 16, they **must** choose to Hit
      2. If Dealer’s Score is greater than or equal to 17, they **must** choose to Stay
      3. If the Dealer gets a Blackjack, all Players Lose the Round unless they also have a Blackjack
         1. Players with a Blackjack in the same round as Dealer Push, and their Bet is returned to their Balance during Payout

#### ~~ROUND END (PAYOUT)~~

1. Pot (or Table Bank) adds an amount of chips referred to as Payout Final to the Balance of each Winning Player equal to the product of the Player’s initial Bet for the Round and the Player’s Payout Multiplier for the Round, which then has the amount of the Player’s initial Bet added to that amount, from the chips contained in Pot.
   1. If Pot does not contain an equal or greater amount than the sum of all Winning Player’s Payout Final, then Table Bank adds chips to the Pot such that the resulting amount is equal to total of all Winning Player’s Payout Final.

#### ~~ROUND END (CLEANUP)~~

1. Dealer takes all Cards from each Hand of Each Player and returns them to Deck before shuffling exactly 8 times for no particular reason
2. Dealer then Ends the Round

# Reference

* + - 1. <https://www.wikihow.com/Play-Blackjack>

(2) <https://docs.oracle.com/en/java/javase/21/docs/api/index.html>