WELCOME TO BLACKJACK Designed by: Harrison, Asher, Cooper



Game description and rules / images

- Objective: Achieve a hand value close to 21 without exceeding it.
- Initial Dealing: Players receive two cards, and the dealer has one face-up and one face-down card.
- Card Values:
 - Number cards: Face value.
 - Face cards (King, Queen, Jack): 10.
 - Aces: 1 or 11.
- Player's Turn:
 - Choose to "hit" for more cards or "stand" with the current hand.
 - Aim for a hand value closest to 21.
 - Busting (exceeding 21) results in a loss.
- Dealer's Turn:
 - Follow specific rules (e.g., hit until the hand is at least 17).
 - Busting means players who haven't busted win.
- Winning:
 - Closest hand to 21 wins.
 - Winning pays 3:2
 - Ties: A "push" results in the player's bet being returned.

Project Requirements

Functional

- Card dealing tracking a deck properly and shuffling when needed
- Game logic win/losses and dealer algorithm
- User interaction player hit and stand

Non-functional

- GUI background
- GUI layout

Project Solution Approach

Major Components

- Tracking a deck to ensure accurate odds of winning
- Working GUI
- Working settings
- Accurate player and dealer logic

Game/UI features

- Card layouts
- Terminal printing info
- Game settings

GUI improvements



Continue



BlackJack

background: Imagelcon starticon: Imagelcon starticon: Imagelcon starticon: Imagelcon backgroundScreen: Jlabel introScreen: Jlabel round: Round deck: Deck bankroli; Bet round/over: boolean autoDealSettling: boolean bettlingSettling: boolean hitCount: int firstRound: boolean playerWinsTotal: int dealerWinsTotal: int dealerWinsTotal: int

Dealer

totalCardScore: Integer
 dealerHand: Hand

+ Dealer():

Integer

+ getDealerHand(): Hand + setInitialScore(): void

+ setScore(Integer): void + getTotalCardScore():

Bet

- bankRoll: int - betValue: double - payout: double - payoutRateNormal: <u>double</u>

+ Bet():

+ addBet(Integer): void + setBetValue(): void + getBetValue(): double

+ betLoss(): void + getPayout(): double

+ setPayoutGamingMode(): void

+ betWin(): void + getBankRollAmount():

/

playerName: String - playerCardNum1: Card playerCardNum2: Card dealerCardNum1: Card - dealerCardNum2: Card startingScreenFrame: JFrame - settingScreenFrame: JFrame - startingScreenPanel: JPanel settingScreenPanel: JPanel blackJackScreenFrame: JFrame - blackJackScreenPanel: JPanel betButton: JButton betLabel: JLabel betAmountTextField: JTextField playerNameTextField: JTextField hitButton: JButton standButton: JButton continueButton: JButton startButton: JButton settingsButton: JButton

startButton: JButton
bettingToggle: JCheckBox
autoDeal: JCheckBox

returnButton: JCheckBox playerCard1: JLabel playerCard2: JLbael

- dealerCard1: JLabel - dealerCard2: JLabel - plaverText: JLabel

- dealerText: JLabel - playerScoreLabel: JLabel - dealerSoreLabe;: JLabel

- playerWinsLabel: JLabel - dealerWinsLabel: JLabel - roundHighlights: String - textArea: JTextArea

+ main(string[] args): static void

+ BlackJack(): + startNeytPour

+ startNextRound(): void

+ runGUI(): void

+ startingScreenGUI: void + blackJackGUI: void

- genStartingScreenPanel(): JPanel - genBlackJackGUI(): JPanel

+ dealerTurn(): void

- doAutoDeal(): void - addButtonCallbackHandlers: void

Hand

+ BLACKJACK: static final int - entityCards: ArrayList<Card>

+ Hand():

Round

+ deck: Deck

+ playerHand: Hand

+ dealerHand: Hand

+ playerCardScore: int

+ dealerCardScore: int

+ dealerStartScore: int

+ BLACKJACK: static final int

+ winStatus: int

+ getCard(); Card

+ playRound(): int

+ getDealerCard(): Card

+ getPlayerScore(): int

+ playerTurn(): Card

Deck

+ cards: ArrayList<Card>

+ suits: ArrayList<String>

+ availableCards: Integer

+ givePlayerCard: Card

+ availableCard(): Boolean

+ shuffleDeck(): void + giveDealerCard(): Card

+ totalCards: Integer

ArrayList<String>

+ faceCards

+ Deck():

+ getDealerCardScore(): int

+ Round():

+ calculateScore(): int + isBlackJack(): Boolean

+ getPlayerCard(): Card

+ getDealerCard(): Card

Card

- dealerHand: Boolean - playerHand: Boolean - discard: Boolean - suit: String - value: Integer - cardName: String

+ Card(Integer, String, String):

+ getCardName(): String + setDealerHand(Boolean):

void + setPlayerHand(Boolean) void

+ getDealerHand(): Boolean + getPlayerHand(): Boolean

+ getPlayerHand(): Boole + getDiscard(): Boolean + getSuit(): String

+ getSuit(): String + getValue(): Integer



- Messaging Used
 - Snapchat
 - iMessage
 - Discord
- A few initial issues with merging
- We used branches a ton
- Coding as a group was super useful when combining features
- Coded 60% independently and 40% together

Testing, Validation, and Acceptance Plan

Testing used
Unit tests
User tests

#1 way we decided if the project was finished was through user testing





Lessons Learned

- Start with the GUI
- Group coding was very efficient
- Branches are powerful

Positive Experiences

- We are proud of what we've built
- It was great to see everyone's efforts come together