Black Jack Project
design phase
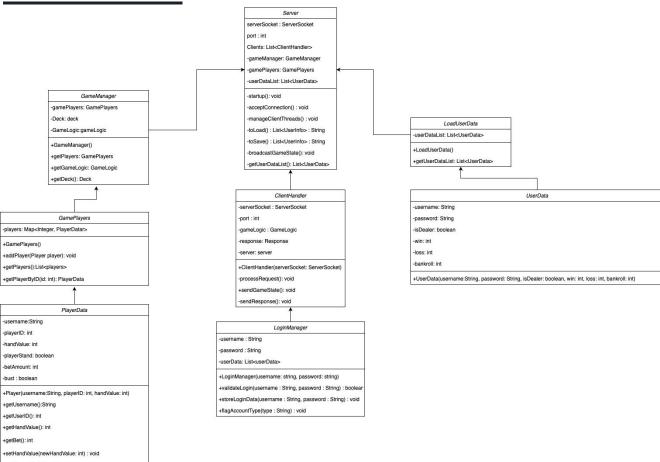
Andrew Nguyen
Sandeep Deoja
Ishwdeep Singh

Class Diagram
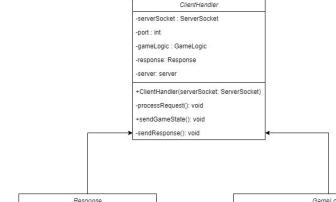
MessageType
- HIT
- STAND
- DOUBLE_DOWN
- START_ROUND
- END_ROUND
- LOGIN
- LOGOUT
- DEPOSIT
- WITHDRAW
- BET

ClientMessage
- -username:String
- -password:String
- -isDealer: boolean
- -depAmount: int
- -withdrawAmount: int
- -type: MessageType
- -playerID: int
- -win: int
- -loss: int
- -bet: int

- +ClientMessage()
- +getUsername():String
- +getPassword():String
- +getIsDealer(): boolean
- +getType(): type
- +getWithDraw(): int
- +getPlayerID(): int
- +getWin(): int
- +getLoss(): int
- +getBet(): int
- +setType(): void
- +setUserName(username: String): void
- +setPassword(password: String): void
- +setPlayerID(playerID: int): void
- +setIsDealer(isDealer: boolean): void
- +setDepAmount(amount: int): void
- +setWithdrawAmount(amount: int): void
- +setBet(amount: int): void

Player
- -username:String
- -playerID: int
- -handValue: int
- -playerStand: boolean
- -betAmount: int

- +Player(username:String, playerID: int, handValue: int)
- +getUsername():String
- +getUserID(): int
- +getHandValue(): int
- +getBet(): int
- +setHandValue(newHandValue: int) : void
- +setPlayerStand(): void
- +setBet(amoutn: int): void

GameManager(SetupGame environment)
- playerInformation: PlayerInformation
- Deck: deck
- GameLogic:gameLogic

- GameManager()

Server
- serverSocket : ServerSocket
- port : int
- Clients: List<ClientHandler>
- -gameManager: GameManager
- -PlayerInformation playerinformation

- acceptConnection() : void
- manageClientThreads() : void
- toLoad() : List<UserInfo> : String
- toSave() : List<UserInfo> : String
- broadcastGameState(): void

Client
- socket : Socket
- serverAddress : String
- port : int
- message: ClientMessage

- connectToServer() : void
- sendLoginRequest(username: string, password: string) : void
- sendLogOutRequest(username: string, playerID: int) : void
- sendHitRequest(username: string, playerID: int) : void
- sendStandRequest(username: string, playerID: int) : void
- sendDoubleRequest(username: string, playerID: int) : void
- sendStartRoundRequest() : void
- sendEndRoundRequest() : void
- sendDepositRequest(username: string, amount: int) : void
- sendWithdrawRequest(username: string, amount: int) : void
- sendPlaceBetRequest(username: string, playerID: int, amount: int): void
- receiveResponse() : String
- recieveGameState(): void

PlayerInformation
- players: Map<Integer, Player>

- +PlayerInformation()
- addPlayer(Player player): void
- getPlayers():List<Players>

Response
- -responseType: ResponseType
- -validated: boolean
- -playerID: int
- -isDealer: boolean
- -count: int
- -playerInfo: PlayerInformation

- +Response()
- +getPlayerID(): int
- +getResponseType(): ResponseType
- +getIsDealer(): boolean
- +getAllPlayers(): PlayerInformation
- +setPlayerID(playerID: int): void
- +setIsDealer(isDealer: boolean): void
- +setPlayerInformation(): void

ResponseType
- SUCCESS
- FAIL
- DEALER
- PLAYER
- GAME_STATE

ClientHandler
- serverSocket : ServerSocket
- port : int
- gameLogic : GameLogic

- ClientHandler(serverSocket: ServerSocket)
- processRequest(): void
- sendGameState(): void

GUIManager
- currentScreen : String
- client: Client

- GUIManager(client: Client)
- displayLoginScreen() : void
- chooseDisplay() : void
- showErrorMessages(message : String) : void

Card
- value : int

- Card(int value)
- getValue(): int

Deck
- cards: List<Card>

- +Deck()
- initializeDeck(): void
- shuffle() : void
- drawCard() : Card

GameLogic
- deck: Deck
- currentPlayer : Player
- playerInformation: playerInformation
- startRound: boolean

- +GameLogic(playerInformation: PlayerInformation, deck: Deck)
- dealCards(): void
- checkHandValue(hand : List<Card>) : int
- determineOutcome() : void
- shuffleDeck() : void
- doHit(playerID: int)

LoginManager
- username : String
- password : String

- validateLogin(username : String, password : String) : boolean
- storeLoginData(username : String, password : String) : void
- flagAccountType(type : String) : void

Dealer
- isDealer: boolean
- client: Client

- Dealer(isDealer: boolean, client: Client)
- startRound() : void
- Hit() : void
- finishRound() : void

Player
- username : String
- accountBalance : double
- playerID: int
- numWin: int
- numLoss: int
- client: Client

- Player(username: String, accountBalance: int, playerID: int, win: int, loss: int, client: Client)
- joinGame() : boolean
- leaveGame() : boolean
- placeBet(amount : double) : boolean
- hit() : void
- stand() : void
- doubleDown() : boolean
- addWin(): void
- addLoss(): void

Server Side

Server

serverSocket : ServerSocket

port : int

Clients: List<ClientHandler>

-gameManager: GameManager

-gamePlayers: GamePlayers

-userDataList: List<UserData>

-startup(): void

-acceptConnection() : void

-manageClientThreads() : void

-toLoad() : List<UserInfo> : String

-toSave() : List<UserInfo> : String

-broadcastGameState(): void

-getUserDataList(): List<UserData>

GameManager

-gamePlayers: GamePlayers

-Deck: deck

-GameLogic:gameLogic

+GameManager()

+getPlayers: GamePlayers

+getGameLogic: GameLogic

+getDeck(): Deck

LoadUserData

-userDataList: List<UserData>

+LoadUserData()

+getUserDataList: List<UserData>

GamePlayers

-players: Map<Integer, PlayerDatar>

+GamePlayers()

+addPlayer(Player player): void

+getPlayers():List<players>

+getPlayerByID(id: int): PlayerData

ClientHandler

-serverSocket : ServerSocket

-port : int

-gameLogic : GameLogic

-response: Response

-server: server

+ClientHandler(serverSocket: ServerSocket)

+processRequest(): void

+sendGameState(): void

-sendResponse(): void

UserData

-username: String

-password: String

-isDealer: boolean

-win: int

-loss: int

-bankroll: int

+UserData(username:String, password: String, isDealer: boolean, win: int, loss: int, bankroll: int)

PlayerData

-username:String

-playerID: int

-handValue: int

-playerStand: boolean

-betAmount: int

-bust : boolean

+Player(username:String, playerID: int, handValue: int)

+getUsername():String

+getUserID(): int

+getHandValue(): int

+getBet(): int

+setHandValue(newHandValue: int) : void

+setPlayerStand(): void

+setBet(amoutn: int): void

LoginManager

-username : String

-password : String

-userData: List<userData>

+LoginManager(username: string, password: string)

+validateLogin(username : String, password : String) : boolean

+storeLoginData(username : String, password : String) : void

+flagAccountType(type : String) : void

Server Side Cont.

ClientHandler

-serverSocket : ServerSocket

-port : int

-gameLogic : GameLogic

-response: Response

-server: server

+ClientHandler(serverSocket: ServerSocket)

-processRequest(): void

+sendGameState(): void

-sendResponse(): void

ResponseType

LOGIN_SUCCESS

LOGIN_FAIL

PLACE_BET_SUCCESS

PLACE_BET_FAIL

WITHDRAW_SUCCESS

WITHDRAW_FAIL

DEALER

PLAYER

GAME_STATE
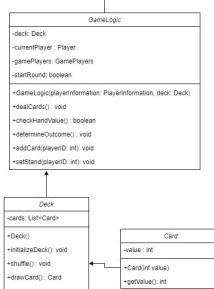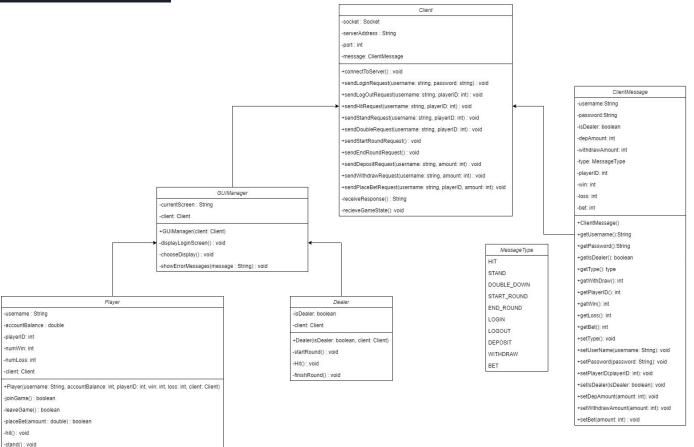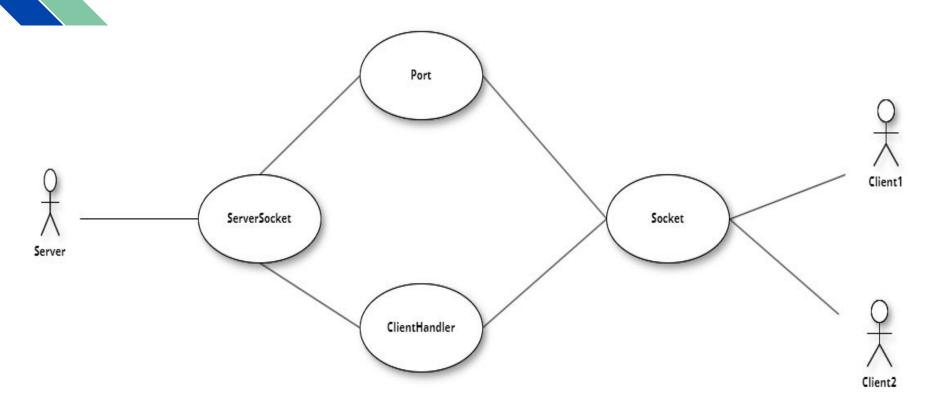
PLAYER_TURN

Response

-responseType: ResponseType

-validated: boolean

-playerID: int

-isDealer: boolean

-count: int

-gamePlayers; GamePlayers

+Response()

+getPlayerID(): int

+getResponseType(): ResponseType

+getIsDealer(): boolean

+getAllPlayers(): GamePlayers

+setPlayerID(playerID: int): void

+setIsDealer(isDealer: boolean): void

+setPlayerInformation(): void

GameLogic

-deck: Deck

-currentPlayer : Player

-gamePlayers: GamePlayers

-startRound: boolean

+GameLogic(playerInformation: PlayerInformation, deck: Deck)

+dealCards() : void

+checkHandValue() : boolean

+determineOutcome() : void

+addCard(playerID: int): void

+setStand(playerID: int): void

Deck

-cards: List<Card>

+Deck()

+initializeDeck(): void

+shuffle() : void

+drawCard() : Card

Card

-value : int

+Card(int value)

+getValue(): int

Client Side

Client

-socket : Socket
-serverAddress : String
-port : int
-message: ClientMessage

+connectToServer() : void
+sendLoginRequest(username: string, password: string) : void
+sendLogOutRequest(username: string, playerID: int) : void
+sendHitRequest(username: string, playerID: int) : void
+sendStandRequest(username: string, playerID: int) : void
+sendDoubleRequest(username: string, playerID: int) : void
+sendStartRoundRequest() : void
+sendEndRoundRequest() : void
+sendDepositRequest(username: string, amount: int) : void
+sendWithdrawRequest(username: string, amount: int) : void
+sendPlaceBetRequest(username: string, playerID, amount: int): void
-receiveResponse() : String
-recieveGameState(): void

ClientMessage

-username:String
-password:String
-isDealer: boolean
-depAmount: int
-withdrawAmount: int
-type: MessageType
-playerID: int
-win: int
-loss: int
-bet: int

+ClientMessage()
+getUsername():String
+getPassword():String
+getIsDealer(): boolean
+getType(): type
+getWithDraw(): int
+getPlayerID(): int
+getWin(): int
+getLoss(): int
+getBet(): int
+setType(): void
+setUserName(username: String): void
+setPassword(password: String): void
+setPlayerID(playerID: int): void
+setIsDealer(isDealer: boolean): void
+setDepAmount(amount: int): void
+setWithdrawAmount(amount: int): void
+setBet(amount: int) : void

GUIManager

-currentScreen : String
-client: Client

+GUIManager(client: Client)
-displayLoginScreen() : void
-chooseDisplay() : void
-showErrorMessages(message : String) : void

MessageType

HIT
STAND
DOUBLE_DOWN
START_ROUND
END_ROUND
LOGIN
LOGOUT
DEPOSIT
WITHDRAW
BET

Player

-username : String
-accountBalance : double
-playerID : int
-numWin : int
-numLoss : int
-client: Client

+Player(username: String, accountBalance: int, playerID: int, win: int, loss: int, client: Client)
-joinGame() : boolean
-leaveGame() : boolean
-placeBet(amount : double) : boolean
-hit() : void
-stand() : void
-doubleDown() : boolean
-deposit(): void
-withdraw(): void

Dealer

-isDealer: boolean
-client: Client

+Dealer(isDealer: boolean, client: Client)
-startRound() : void
-Hit() : void
-finishRound() : void

Program Design:Multiple Client Connection

Server Client Error Handler

One-Way Message Between a Client and Server

Sequence Diagrams

Use Case: Player Options → Place Bet

Use Case: Dealer Action → Start Game and Dealer Hit

GUI

Dealer Class	Client	Server	Client Handler	GAME LOGIC

USER

StartGame() triggered

send Game Start MESSAGE

processes gamestart object

add card to dealer hand

Validation Successful

gamestate Update

gamestate Update

gamestate Update

if NOT Bust OR 17-21
trigger DealerHit()

DealerHit() triggered

send DealerHit() MESSAGE

processes dealerhit object

add card to dealer hand

If Bust or 17-21 = TRIGGER EndGame()

Use Case: Dealer Action → End Game

GUI

Dealer Class | Client | Server | Client Handler | GAME LOGIC | Client(s) PLAYER(s)

USER

EndGame() triggered

send EndGame MESSAGE

processes endgame object

process EndGame()

determine WINNER update file/data

Validation Successful

gamestate Update

gamestate Update

gamestate Update

SEND GAME DATA TO PLAYER CLIENTS

Trigger GameStart() ???