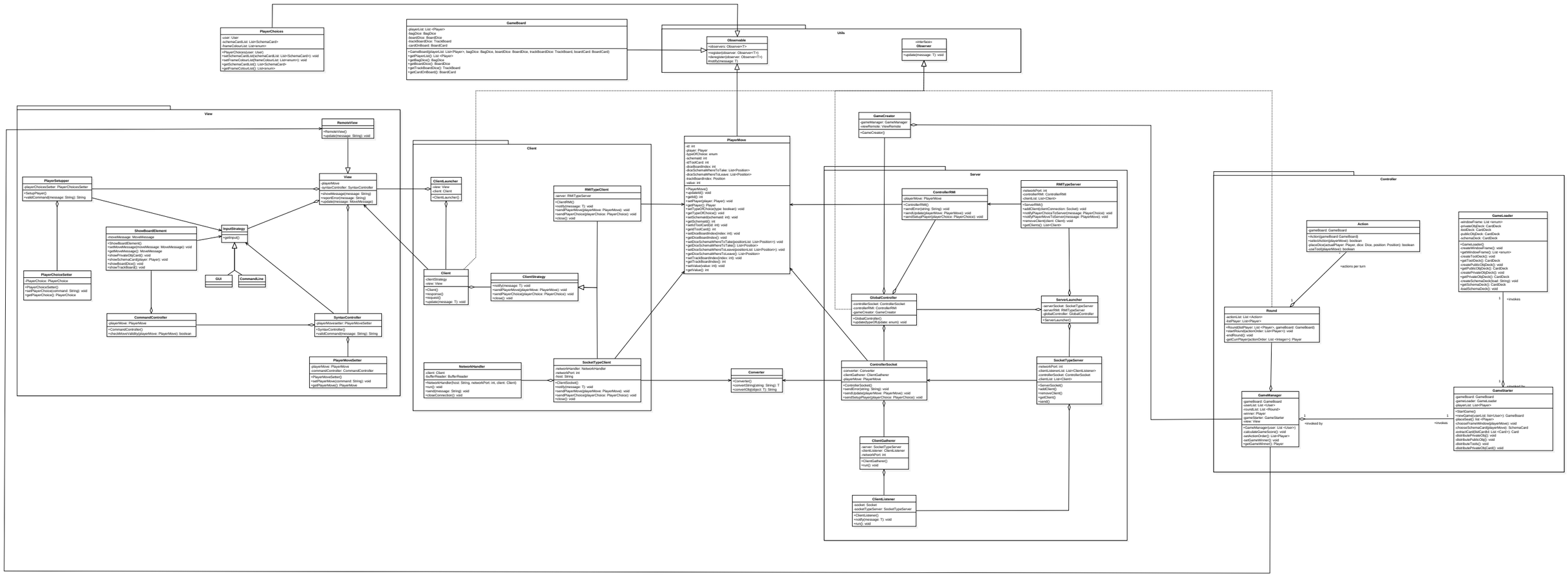
PlayerChoices
-user: User
-schemaCardList: List<SchemaCard>
-frameColourList: List<enum>
+PlayerChoices(user: User)
+setSchemaCardList(schemaCardList: List<SchemaCard>): void
+setFrameColourList(frameColourList: List<enum>): void
+getSchemaCardList(): List<SchemaCard>
+getFrameColourList(): List<enum>
GameBoard
-playerList: List <Player>
-bagDice: BagDice
-boardDice: BoardDice
-trackBoardDice: TrackBoard
-cardOnBoard: BoardCard
+GameBoard(playerList: List<Player>, bagDice: BagDice, boardDice: BoardDice, trackBoardDice: TrackBoard, boardCard: BoardCard)
+getPlayerList(): List <Player>
+getBagDice(): BagDice
+getBoardDice(): BoardDice
+getTrackBoardDice(): TrackBoard
+getCardOnBoard(): BoardCard
Utils
Observable
+observers: Observer<T>
+register(observer: Observer<T>)
+deregister(observer: Observer<T>)
#notify(message: T)
«interface»
Observer
+update(message: T): void
View
RemoteView
+RemoteView()
+update(message: String): void
View
-playerMove
-syntaxController: SyntaxController
+showMessage(message: String)
+reportError(message: String)
+update(message: MoveMessage)
PlayerSetupper
-playerChoicesSetter: PlayerChoicesSetter
+SetupPlayer()
+validCommand(message: String): String
InputStrategy
+getInput()
ShowBoardElement
-moveMessage: MoveMessage
+ShowBoardElement()
+setMoveMessage(moveMessage: MoveMessage): void
+getMoveMessage(): MoveMessage
+showPrivateObjCard(): void
+showSchemaCard(player: Player): void
+showBoardDice(): void
+showTrackBoard(): void
GUI
CommandLine
PlayerChoiceSetter
-PlayerChoice: PlayerChoice
+PlayerChoice(Setter)
+setPlayerChoice(command: String): void
+getPlayerChoice(): PlayerChoice
CommandController
-playerMove: PlayerMove
+CommandController()
+checkMoveValid(playerMove: PlayerMove): boolean
SyntaxController
-playerMovesetter: PlayerMoveSetter
+SyntaxController()
+validCommand(message: String): String
PlayerMoveSetter
-playerMove: PlayerMove
-commandController: CommandController
+PlayerMoveSetter()
+setPlayerMove(command: String): void
+getPlayerMove(): PlayerMove
Client
ClientLauncher
-view: View
-client: Client
+ClientLauncher()
Client
-clientStrategy
-view: View
+Client()
+response()
+request()
+update(message: T): void
ClientStrategy
+notify(message: T): void
+sendPlayerMove(playerMove: PlayerMove): void
+sendPlayerChoice(playerChoice: PlayerChoice): void
+close(): void
RMITypeClient
-server: RMITypeServer
+RMIClient()
+notify(message: T): void
+sendPlayerMove(playerMove: PlayerMove): void
+sendPlayerChoice(playerChoice: PlayerChoice): void
+close(): void
NetworkHandler
-client: Client
-bufferReader: BufferReader
+NetworkHandler(host: String, networkPort: int, client: Client)
+run(): void
+send(message: String): void
+closeConnection(): void
SocketTypeClient
-networkHandler: NetworkHandler
-networkPort: int
-host: String
+ClientSocket()
+notify(message: T): void
+sendPlayerMove(playerMove: PlayerMove): void
+sendPlayerChoice(playerChoice: PlayerChoice): void
+close(): void
PlayerMove
-id: int
-player: Player
-typeOfChoice: enum
-schemaId: int
-idToolCard: int
-diceBoardIndex: int
-diceSchemaWhereToTake: List<Position>
-diceSchemaWhereToLeave: List<Position>
-trackBoardIndex: Position
-value: int
+PlayerMove()
+updateID(): void
+getId(): int
+setPlayer(player: Player): void
+getPlayer(): Player
+setTypeOfChoice(type: boolean): void
+getTypeOfChoice(): void
+setSchemaId(schemaId: int): void
+getSchemaId(): int
+setToolCardId(id: int): void
+getToolCardId: int
+setDiceBoardIndex(index: int): void
+getDiceBoardIndex(): void
+setDiceSchemaWhereToTake(positionsList: List<Position>): void
+getDiceSchemaWhereToTake(): List<Position>
+setDiceSchemaWhereToLeave(positionsList: List<Position>): void
+getDiceSchemaWhereToLeave(): List<Position>
+setTrackBoardIndex(index: int): void
+getTrackBoardIndex(): int
+setValue(value: int): void
+getValue(): int
Converter
+Converter()
+convertStringToString: String): T
+convertObj(object: T): String
GameCreator
-gameManager: GameManager
-viewRemote: ViewRemote
+GameCreator()
Server
ControllerRMI
-playerMove: PlayerMove
+ControllerRMI()
+sendErrorString: String): void
+sendUpdate(playerMove: PlayerMove): void
+sendSetupPlayer(playerChoice: PlayerChoice): void
RMITypeServer
-networkPort: int
-controllerRMI: ControllerRMI
-clientList: List<Client>
+ServerRMI()
+addClient(clientConnection: Socket): void
+notifyPlayerChoiceToServer(message: PlayerChoice): void
+notifyPlayerMoveToServer(message: PlayerMove): void
+removeClient(client: Client): void
+getClients(): List<Client>
GlobalController
-controllerSocket: ControllerSocket
-controllerRMI: ControllerRMI
-gameCreator: GameCreator
+GlobalController()
+update(typeOfUpdate: enum): void
ServerLauncher
-serverSocket: SocketTypeServer
-serverRMI: RMITypeServer
-globalController: GlobalController
+ServerLauncher()
ControllerSocket
-converter: Converter
-clientGatherer: ClientGatherer
-playerMove: PlayerMove
+ControllerSocket()
+sendError(errorString: String): void
+sendUpdate(playerMove: PlayerMove): void
+sendSetupPlayer(playerChoice: PlayerChoice): void
SocketTypeServer
-networkPort: int
-clientListenerList: List<ClientListener>
-controllerSocket: ControllerSocket
-clientList: List<Client>
+ServerSocket()
+addClient()
+removeClient()
+getClient()
+send()
ClientGatherer
-server: SocketTypeServer
-clientListener: ClientListener
-networkPort: int
+ClientGatherer()
+run(): void
ClientListener
-socket: Socket
-socketTypeServer: SocketTypeServer
+ClientListener()
+notify(message: T): void
+run(): void
Controller
Action
-gameBoard: GameBoard
+Action(gameBoard GameBoard)
+selectAction(playerMove): boolean
+placeDice(actualPlayer: Player, dice: Dice, position: Position): boolean
+useTool(playerMove): boolean
+actions per turn
GameLoader
-windowFrame: List <enum>
-privateObjDeck: CardDeck
-toolDeck: CardDeck
-publicObjDeck: CardDeck
-schemaDeck: CardDeck
+GameLoader()
+createWindowFrame(): void
+getWindowFrame(): List <enum>
-createToolDeck(): void
+getToolDeck(): CardDeck
-createPublicObjDeck(): void
+getPublicObjDeck(): CardDeck
-createPrivateObjDeck(): void
+getPrivateObjDeck(): CardDeck
-createSchemaDeck(): void
+getSchemaDeck(): CardDeck
-loadSchemaDeck(): void
+invokes
Round
-actionList: List <Action>
-toMPlayer: List<Player>
+Round(toMPlayer: List <Player>, gameBoard: GameBoard)
+startRound(actionOrder: List<Player>): void
-endRound(): void
-getCurrPlayer(actionOrder: List <integer>): Player
GameManager
-gameBoard: GameBoard
-userList: List <User>
-roundList: List <Round>
-winner: Player
-gameStarter: GameStarter
-view: View
+GameManager(user: List <User>)
-calculateGameScore(): void
-setActionOrder(): List<Player>
-setGameWinner(): void
-getGameWinner(): Player
+invoked by
+invokes
GameStarter
-gameBoard: GameBoard
-gameLoader: GameLoader
-playerList: List<Player>
+StartGame()
+newGame(userList: list<User>): GameBoard
-placeSeat(): list <Player>
-chooseFrameWindow(playerMove): void
-chooseSchemaCard(playerMove): SchemaCard
-extractCard(listCard): List <Card>): Card
-distributePrivateObj(): void
-distributePublicObj(): void
-distributeTools(): void
-distributePrivateObj(card): void