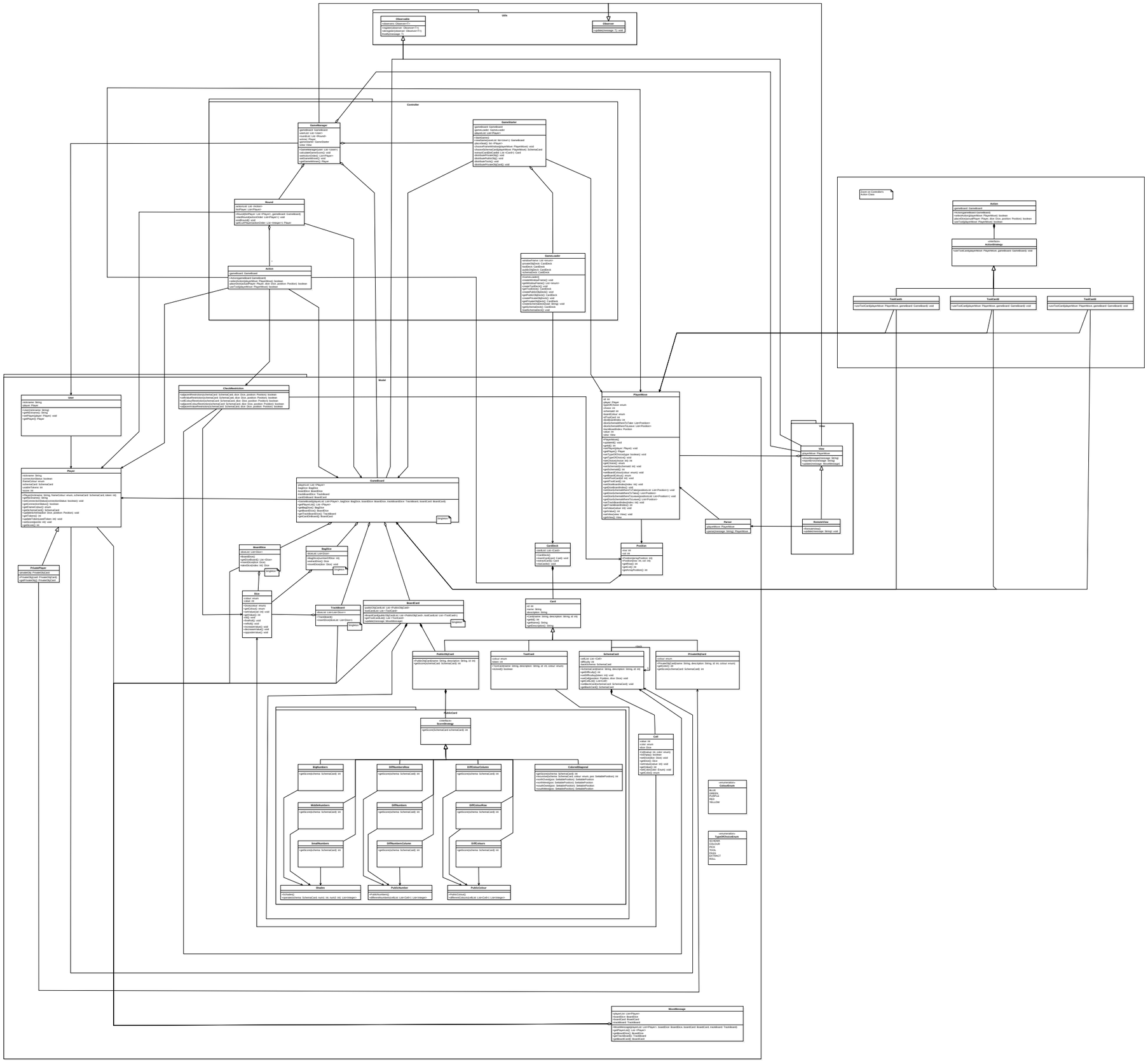
Utils
Observable
+observers: Observer<T>
+registerObserver: Observer<T>
+deregisterObserver: Observer<T>
+notifyObservers()
Observer
+update(message: T): void
Controller
GameManager
-gameboard: Gameboard
-gameLeader: GameLeader
-round: List <Round>
-winner: Player
-gameStarter: GameStarter
-view: View
+GameManager(Color: List <User>)
-calculateGameScore(): void
-setActiveOrder(): List<Player>
-getNextRound(): void
-getCurrentWinner(): Player
GameStarter
-gameboard: Gameboard
-gameLeader: GameLeader
-playerList: List<Player>
+StartGame()
-setPlayerOrder(playerList: List<User>): Gameboard
-placeDiscs(): list <Player>
-chooseHomeWinningsp(playerMove: PlayerMove): void
-setupCardDeck(publicCardDeck: List <PublicCard>, schemaCard: SchemaCard)
-setupCardDeck(Card: List <Card>): Card
-dentributePrivateBag(): void
-dentributeTrackBag(): void
-dentributeTiles(): void
-dentributePrivateBag(Card): void
Round
-setsList: List <Action>
-listPlayer: List<Player>
+Round(listPlayer: List <Player>, gameboard: Gameboard)
+getNextPlayer(): Player
+endRound(): void
-getListPlayersplaceOrder(Order: List <Integer>): Player
Action
-gameBoard: GameBoard
+Action(gameBoard: GameBoard)
+setactivationplayer(Player: Player|Null): boolean
+placeDisc(placeMove: Player, disc: Disc, position: Position): boolean
-canTileEqlayer(Player: Player|Null): boolean
GameLeader
-endowPlaces: List <Int>
-privateObjDeck: CardDeck
-tileDeck: CardDeck
-publicObjDeck: CardDeck
-schemaDeck: CardDeck
+GameLeader()
-createWinEndowPlace(): void
-setupEndowPlaces(Int: List <Int>)
-createTileDeck(): void
-getTileDeck(): CardDeck
-getPublicObjDeck(): CardDeck
-getPrivateObjDeck(): CardDeck
-createPrivateObjDeck(): void
-getSchemaDeck(): CardDeck
-createSchemaDeck(Load: String): void
-getSchemaDeck(): CardDeck
-setSchemaDeck(): void
Model
User
-nickname: String
-isactiveplayer: Boolean
-frameColour: enum
-schemaCard: SchemaCard
-totalTokens: int
-score: int
+User(nickname: String)
+getNickname(): String
+setPlayer(player: Player): void
+getPlayer(): Player
CheckRestriction
+adjacentRestriction(schemaCard: SchemaCard, dice: Dice, position: Position): boolean
+colRowRestriction(schemaCard: SchemaCard, dice: Dice, position: Position): boolean
+colRowAdjacentRestriction(schemaCard: SchemaCard, dice: Dice, position: Position): boolean
+adjacentColourRestriction(schemaCard: SchemaCard, dice: Dice, position: Position): boolean
+adjacentHeavenRestriction(schemaCard: SchemaCard, dice: Dice, position: Position): boolean
PlayerMove
-id: int
-player: Player
-typeOfChoice: enum
-dice: int
-schemaId: int
-boardColour: enum
-dice: Dice
-diceIndexBefore: int
-diceSchemaWhereToTake: List<Position>
-diceSchemaWhereToPlace: List<Position>
-trackBoardIndex: Position
-value: int
-boolean: int
+PlayerMove()
+update(): void
+getID(): int
+setPlayer(player: Player): void
+getPlayer(): Player
+setTypeOfChoice(type: boolean): void
+getTypeOfChoice(): void
+setChoice(choice: int): int
+getChoice(): int
+setSchemaId(schemaId: int): void
+getSchemaId(): int
+setBoardColour(colour: enum): void
+getBoardColour(): void
+setCardToCard(id: int): void
+getCardToCard()
+setDiceBeforeIndex(index: int): void
+getDiceBeforeIndex(): void
+setDiceSchemaWhereToTake(diceSchemaWhereToTake: List<Position>): void
+getDiceSchemaWhereToTake(): List<Position>
+setDiceSchemaWhereToPlace(diceSchemaWhereToPlace): void
+getDiceSchemaWhereToPlace(): List<Position>
+setTrackBoardIndex(index: int): void
+getTrackBoardIndex()
+setValue(value: int): void
+getValue(): int
+setBoolean(boolean): void
+getBool(): void
Parser
-playerMove: PlayerMove
+parse(message: String): PlayerMove
View
User
-playerMove: PlayerMove
+showMessage(message: String)
+getMove(): PlayerMove
+updateMessage: MoveMessage
RemoteView
+Run(socket)
+sendMessage(message: String): void
Player
-nickname: String
-connectionStatus: Boolean
-frameColour: enum
-schemaCard: SchemaCard
-totalTokens: int
-score: int
+Player(nickname: String, frameColour: enum, schemaCard: SchemaCard, token: int)
+getNickname(): String
+setConnectionStatus(connectionStatus: boolean): void
+getConnectionStatus(): boolean
+getFrameColour(): enum
+getSchemaCard(): SchemaCard
+setSchemaCard(schemaCard: SchemaCard)
+getTokens(): int
+setToken(numberToken: int): void
+getScore(): int
+setScore(): void
PrivatePlayer
-drawnObj: PrivateObjCard
+PrivatePlayer()
+setPrivateObjCard(): PrivateObjCard
+getPrivateObjCard(): PrivateObjCard
Gameboard
-playerList: List <Player>
-bagDice: BagDice
-boardDice: BoardDice
-trackBoardDice: TrackBoard
-boardCard: BoardCard
+GameBoard(playerList: List<Player>, bagDice: BagDice, boardDice: BoardDice, trackBoardDice: TrackBoard, boardCard: BoardCard)
+getBag(Dice): BagDice
+getBoardDice(): BoardDice
+getTrackBoard(): TrackBoard
+getCardOnBoard(): BoardCard
Singleton
BoardDice
-diceList: List<Dice>
+BoardDice()
+setDiceOnBoard(): List <Dice>
+Dice(Dice|index: Dice)
+takeDice(index: int): Dice
Singleton
BagDice
-diceList: List<Dice>
-BagDice(numberOfDice: int)
+setDiceDice: Dice): void
+removeDice(dice: Dice): void
Singleton
Dice
-colour: enum
-color: int
-Dice(colour: enum)
+getColour(): enum
+setColour(color): void
+getValue(): int
+roll(): void
+getRandom(): void
+increaseValue(): void
+decreaseValue(): void
+oppositeValue(): void
TrackBoard
-diceList: List<ListDice>
+TrackBoard()
+insertDice(diceList: List<Dice>)
Singleton
BoardCard
-publicObjCardList: List <PublicObjCard>
-toolCardList: List <ToolCard>
+BoardCard(publicObjCard: List <PublicObjCard>, toolCardList: List <ToolCard>)
+getToolCardList(): List <ToolCard>
+updateMessage: MoveMessage
Singleton
CardDeck
-cardList: List<Card>
+CardDeck()
+addCardToDeck(card: Card): void
+removeCard(): Card
+listCard(): void
Position
-row: int
-col: int
+Position(rowPosition: int)
+Position(row: int, col: int)
+getRow(): int
+setRow(): void
+getColumnPosition(): int
Card
-id: int
-name: String
-description: String
+Card(name: String, description: String, id: int)
+getID(): int
+getName(): String
+getDescription()
PublicObjCard
+PublicObjCard(name: String, description: String, id: int)
+getScore(schemaCard: SchemaCard): int
ToolCard
-colour: enum
-token: int
+ToolCard(name: String, description: String, id: int, colour: enum)
+isUsed(): boolean
SchemaCard
-cellList: List <Cell>
-difficulty: int
+SchemaCard(): SchemaCard
+SchemaCard(name: String, description: String, id: int)
+getDifficulty(): int
+setDifficulty(diff: int): void
+setCell(position: Position, dice: Dice): void
+getCellList(): List<Cell>
+setBlockCell(schemaCard): void
+getBlockCard(): SchemaCard
class
PrivateObjCard
-colour: enum
+PrivateObjCard(name: String, description: String, id: int, colour: enum)
+getColour()
+getScore(schemaCard: SchemaCard): int
PublicCard
<interface>
ScoreStrategy
+getScore(schemaCard: schemaCard): int
Cell
-value: int
-color: enum
-dice: Dice
+Cell(value: int, color: enum)
+getValue(): int
+setDice(dice: Dice): void
+getDice(): Dice
+setColor(color): void
+getColor(): int
+setColor(color: Enum): void
+getColor(): enum
EightPoints
+getScore(schema: SchemaCard): int
DiffRowRow
+getScore(schema: SchemaCard): int
DiffColumnColumn
+getScore(schema: SchemaCard): int
ColourDiagonal
+getScore(schema: SchemaCard): int
+countDiagonal(pos: SetablePosition): SetablePosition
+countRowDown(pos: SetablePosition): SetablePosition
+countRowRight(pos: SetablePosition): SetablePosition
+countRowLeft(pos: SetablePosition): SetablePosition
+countRowUp(pos: SetablePosition): SetablePosition
MatchNumbers
+getScore(schema: SchemaCard): int
DiffNumbers
+getScore(schema: SchemaCard): int
DiffColourRow
+getScore(schema: SchemaCard): int
SmallNumbers
+getScore(schema: SchemaCard): int
DiffNumbersColumn
+getScore(schema: SchemaCard): int
DiffColours
+getScore(schema: SchemaCard): int
Shades
+Shades()
+getShades(schema: SchemaCard, num1: int, num2: int): List<Integer>
PublicNumber
+PublicNumber()
+differentNumbers(cellsList: List<Cell>): List<Integer>
PublicColour
+PublicColour()
+differentColours(cellsList: List<Cell>): List<Integer>
<enumeration>
CardColour
BLUE
GREEN
PURPLE
RED
YELLOW
<enumeration>
TypeOfChoiceEnum
SCHEMA
TOOLCAR
PICK
TOOL
DICE
DICEACT
ROLL
MoveMessage
-gBlayerList: List<Player>
-boardDice: BoardDice
-boardCard: BoardCard
-trackBoard: TrackBoard
+MoveMessage(playerList: List<Player>, boardDice: BoardDice, boardCard: BoardCard, trackBoard: TrackBoard)
+getPlayerList(): List<Player>
+getBoardDice(): BoardDice
+getTrackBoard(): TrackBoard
+getBoardCard(): BoardCard