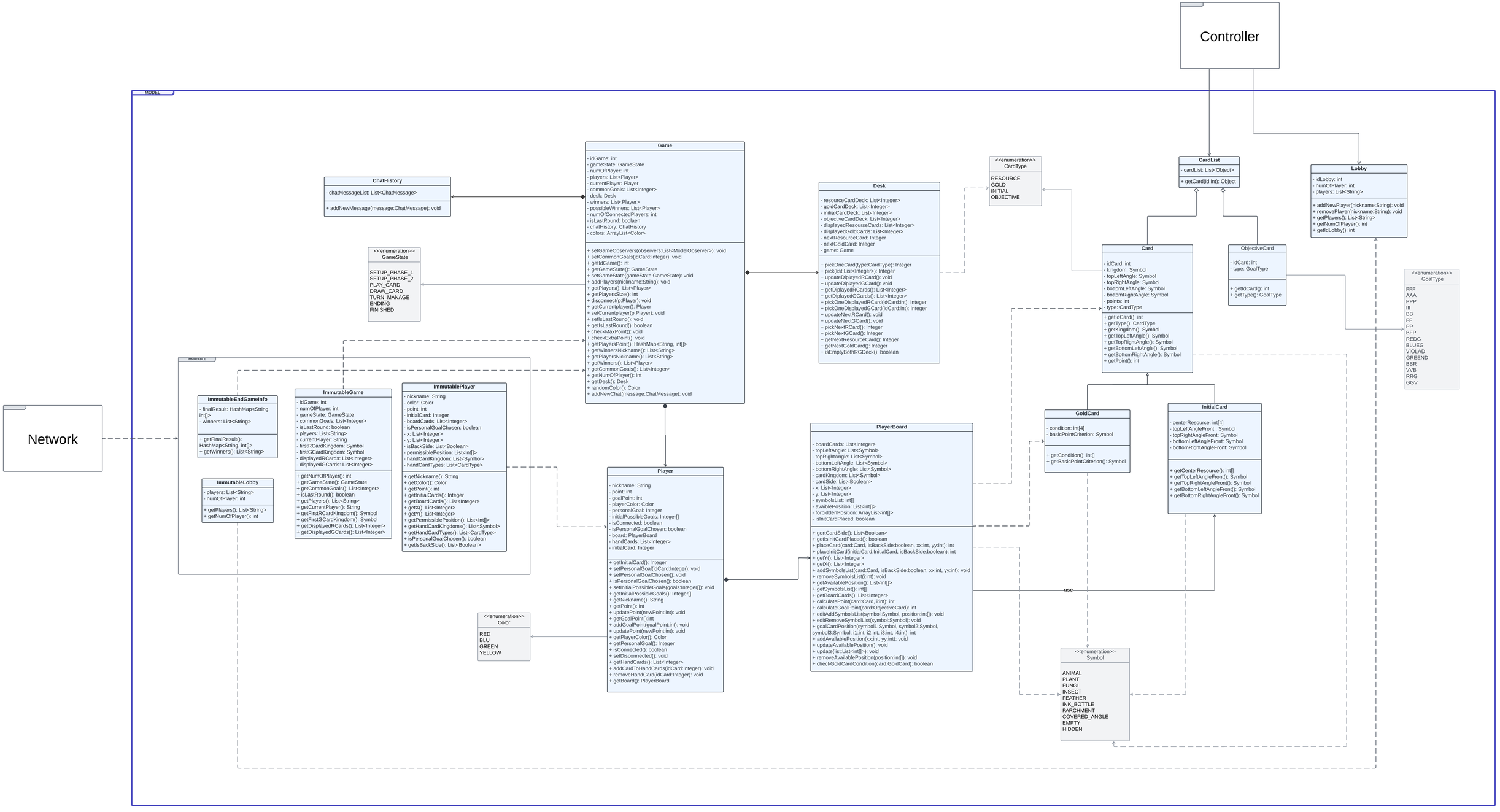
Controller
MODEL
Network
IMMUTABLE
ChatHistory
- chatMessageList: List<ChatMessage>
+ addNewMessage(message:ChatMessage): void
Game
- idGame: int
- gameState: GameState
- numOfPlayer: int
- players: List<Player>
- currentPlayer: Player
- commonGoals: List<Integer>
- desk: Desk
- winners: List<Player>
- possibleWinners: List<Player>
- numOfConnectedPlayers: int
- isLastRound: boolean
- chatHistory: ChatHistory
- colors: ArrayList<Color>
+ setGameObservers(observers:List<ModelObserver>): void
+ setCommonGoals(idCard:Integer): void
+ getIdGame(): int
+ getGameState(): GameState
+ setGameState(gameState:GameState): void
+ addPlayers(nickname:String): void
+ getPlayers(): List<Player>
+ getPlayersSize(): int
+ disconnect(p:Player): void
+ getCurrentplayer(): Player
+ setCurrentplayer(p:Player): void
+ setIsLastRound(): void
+ getIsLastRound(): boolean
+ checkMaxPoint(): void
+ checkExtraPoint(): void
+ getPlayersPoint(): HashMap<String, int[]>
+ getWinnersNickname(): List<String>
+ getPlayersNickname(): List<String>
+ getWinners(): List<Player>
+ getCommonGoals(): List<Integer>
+ getNumOfPlayer(): int
+ getDesk(): Desk
+ randomColor(): Color
+ addNewChat(message:ChatMessage): void
<<enumeration>>
GameState
SETUP_PHASE_1
SETUP_PHASE_2
PLAY_CARD
DRAW_CARD
TURN_MANAGE
ENDING
FINISHED
Desk
- resourceCardDeck: List<Integer>
- goldCardDeck: List<Integer>
- initialCardDeck: List<Integer>
- objectiveCardDeck: List<Integer>
- displayedResourseCards: List<Integer>
- displayedGoldCards: List<Integer>
- nextResourceCard: Integer
- nextGoldCard: Integer
- game: Game
+ pickOneCard(type:CardType): Integer
+ pick(list:List<Integer>): Integer
+ updateDiplayedRCard(): void
+ updateDiplayedGCard(): void
+ getDiplayedRCards(): List<Integer>
+ getDiplayedGCards(): List<Integer>
+ pickOneDisplayedRCard(idCard:int): Integer
+ pickOneDisplayedGCard(idCard:int): Integer
+ updateNextRCard(): void
+ updateNextGCard(): void
+ pickNextRCard(): Integer
+ pickNextGCard(): Integer
+ getNextResourceCard(): Integer
+ getNextGoldCard(): Integer
+ isEmptyBothRGDeck(): boolean
<<enumeration>>
CardType
RESOURCE
GOLD
INITIAL
OBJECTIVE
CardList
- cardList: List<Object>
+ getCard(id:int): Object
Lobby
- idLobby: int
- numOfPlayer: int
- players: List<String>
+ addNewPlayer(nickname:String): void
+ removePlayer(nickname:String): void
+ getPlayers(): List<String>
+ getNumOfPlayer(): int
+ getIdLobby(): int
Card
- idCard: int
- kingdom: Symbol
- topLeftAngle: Symbol
- topRightAngle: Symbol
- bottomLeftAngle: Symbol
- bottomRightAngle: Symbol
- points: int
- type: CardType
+ getIdCard(): int
+ getType(): CardType
+ getKingdom(): Symbol
+ getTopLeftAngle(): Symbol
+ getTopRightAngle(): Symbol
+ getBottomLeftAngle(): Symbol
+ getBottomRightAngle(): Symbol
+ getPoint(): int
ObjectiveCard
- idCard: int
- type: GoalType
+ getIdCard(): int
+ getType(): GoalType
<<enumeration>>
GoalType
FFF
AAA
PPP
III
BB
FF
PP
BFP
REDG
BLUEG
VIOLAD
GREEND
BBR
VVB
RRG
GGV
ImmutableEndGameInfo
- finalResult: HashMap<String, int[]>
- winners: List<String>
+ getFinalResult(): HashMap<String, int[]>
+ getWinners(): List<String>
ImmutableLobby
- players: List<String>
- numOfPlayer: int
+ getPlayers(): List<String>
+ getNumOfPlayer(): int
ImmutableGame
- idGame: int
- numOfPlayer: int
- gameState: GameState
- commonGoals: List<Integer>
- isLastRound: boolean
- players: List<String>
- currentPlayer: String
- firstRCardKingdom: Symbol
- firstGCardKingdom: Symbol
- displayedRCards: List<Integer>
- displayedGCards: List<Integer>
+ getNumOfPlayer(): int
+ getGameState(): GameState
+ getCommonGoals(): List<Integer>
+ isLastRound(): boolean
+ getPlayers(): List<String>
+ getCurrentPlayer(): String
+ getFirstRCardKingdom(): Symbol
+ getFirstGCardKingdom(): Symbol
+ getDisplayedRCards(): List<Integer>
+ getDisplayedGCards(): List<Integer>
ImmutablePlayer
- nickname: String
- color: Color
- point: int
- initialCard: Integer
- boardCards: List<Integer>
- isPersonalGoalChosen: boolean
- x: List<Integer>
- y: List<Integer>
- isBackSide: List<Boolean>
- permissiblePosition: List<int[]>
- handCardKingdom: List<Symbol>
- handCardTypes: List<CardType>
+ getNickname(): String
+ getColor(): Color
+ getPoint(): int
+ getInitialCards(): Integer
+ getBoardCards(): List<Integer>
+ getX(): List<Integer>
+ getY(): List<Integer>
+ getPermissiblePosition(): List<Int[]>
+ getHandCardKingdoms(): List<Symbol>
+ getHandCardTypes(): List<CardType>
+ isPersonalGoalChosen(): boolean
+ getIsBackSide(): List<Boolean>
Player
- nickname: String
- point: int
- goalPoint: int
- playerColor: Color
- personalGoal: Integer
- initialPossibleGoals: Integer[]
- isConnected: boolean
- isPersonalGoalChosen: boolean
- board: PlayerBoard
- handCards: List<Integer>
- initialCard: Integer
+ getInitialCard(): Integer
+ setPersonalGoal(idCard:Integer): void
+ setPersonalGoalChosen(): void
+ isPersonalGoalChosen(): boolean
+ setInitialPossibleGoals(goals:Integer[]): void
+ getInitialPossibleGoals(): Integer[]
+ getNickname(): String
+ getPoint(): int
+ updatePoint(newPoint:int): void
+ getGoalPoint():int
+ addGoalPoint(goalPoint:int): void
+ updatePoint(newPoint:int): void
+ getPlayerColor(): Color
+ getPersonalGoal(): Integer
+ isConnected(): boolean
+ setDisconnected(): void
+ getHandCards(): List<Integer>
+ addCardToHandCards(idCard:Integer): void
+ removeHandCard(idCard:Integer): void
+ getBoard(): PlayerBoard
<<enumeration>>
Color
RED
BLU
GREEN
YELLOW
PlayerBoard
- boardCards: List<Integer>
- topLeftAngle: List<Symbol>
- topRightAngle: List<Symbol>
- bottomLeftAngle: List<Symbol>
- bottomRightAngle: List<Symbol>
- cardKingdom: List<Symbol>
- cardSide: List<Boolean>
- x: List<Integer>
- y: List<Integer>
- symbolsList: int[]
- avaiblePosition: List<int[]>
- forbiddenPosition: ArrayList<int[]>
- isInitCardPlaced: boolean
+ gertCardSide(): List<Boolean>
+ getIsInitCardPlaced(): boolean
+ placeCard(card:Card, isBackSide:boolean, xx:int, yy:int): int
+ placeInitCard(initialCard:InitialCard, isBackSide:boolean): int
+ getY(): List<Integer>
+ getX(): List<Integer>
+ addSymbolsList(card:Card, isBackSide:boolean, xx:int, yy:int): void
+ removeSymbolsList(i:int): void
+ getAvailablePosition(): List<int[]>
+ getSymbolsList(): int[]
+ getBoardCards(): List<Integer>
+ calculatePoint(card:Card, i:int): int
+ calculateGoalPoint(card:ObjectiveCard): int
+ editAddSymbolsList(symbol:Symbol, position:int[]): void
+ editRemoveSymbolsList(symbol:Symbol): void
+ goalCardPosition(symbol1:Symbol, symbol2:Symbol, symbol3:Symbol, i1:int, i2:int, i3:int, i4:int): int
+ addAvailablePosition(xx:int, yy:int): void
+ updateAvailablePosition(): void
+ update(list:List<int[]>): void
+ removeAvailablePosition(position:int[]): void
+ checkGoldCardCondition(card:GoldCard): boolean
use
GoldCard
- condition: int[4]
- basicPointCriterion: Symbol
+ getCondition(): int[]
+ getBasicPointCriterion(): Symbol
InitialCard
- centerResource: int[4]
- topLeftAngleFront : Symbol
- topRightAngleFront: Symbol
- bottomLeftAngleFront: Symbol
- bottomRightAngleFront: Symbol
+ getCenterResource(): int[]
+ getTopLeftAngleFront(): Symbol
+ getTopRightAngleFront(): Symbol
+ getBottomLeftAngleFront(): Symbol
+ getBottomRightAngleFront(): Symbol
<<enumeration>>
Symbol
ANIMAL
PLANT
FUNGI
INSECT
FEATHER
INK_BOTTLE
PARCHMENT
COVERED_ANGLE
EMPTY
HIDDEN