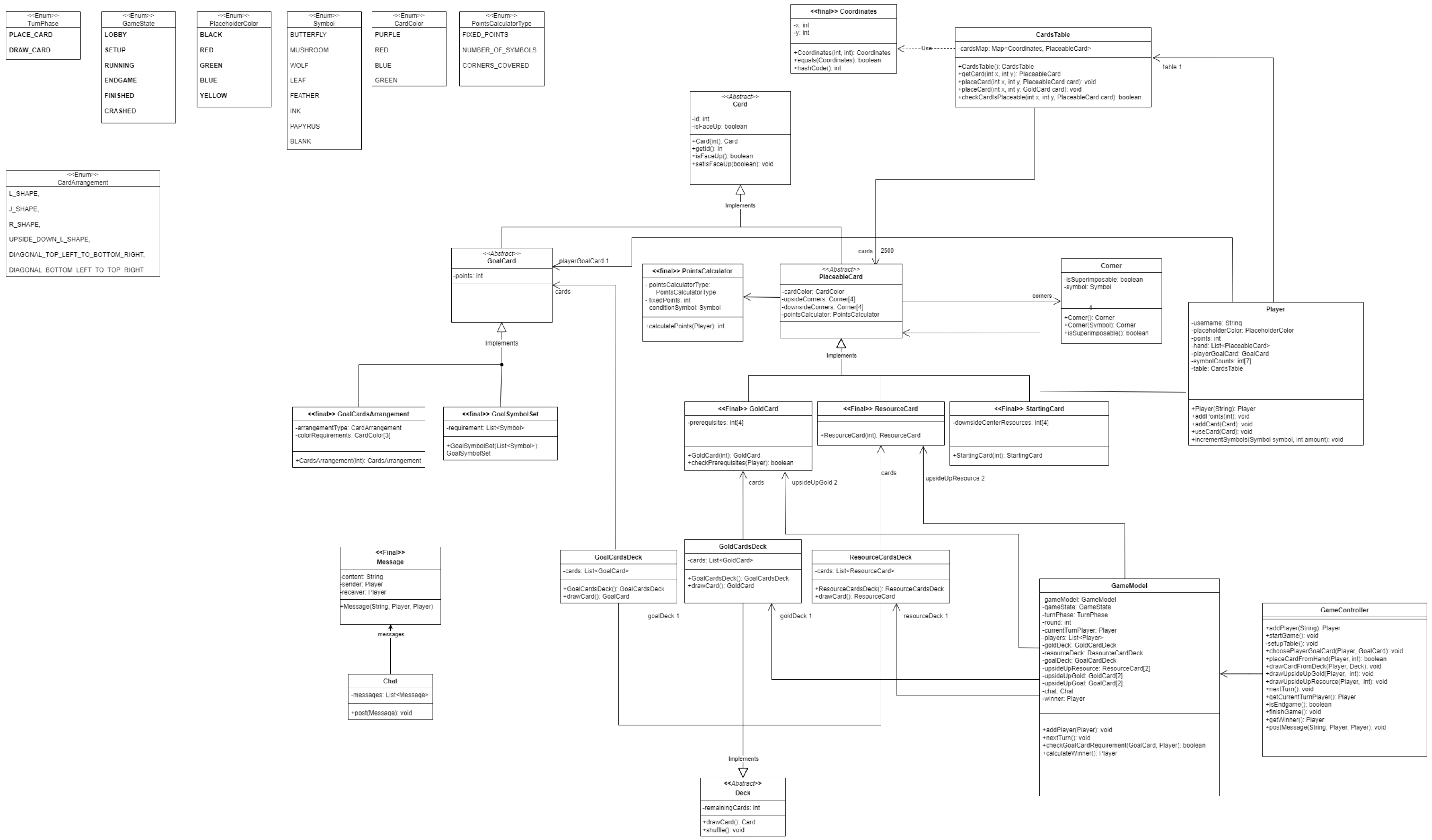

<<Enum>> TurnPhase
PLACE_CARD
DRAW_CARD
<<Enum>> GameState
LOBBY
SETUP
RUNNING
ENDGAME
FINISHED
CRASHED
<<Enum>> PlaceholderColor
BLACK
RED
GREEN
BLUE
YELLOW
<<Enum>> Symbol
BUTTERFLY
MUSHROOM
WOLF
LEAF
FEATHER
INK
PAPYRUS
BLANK
<<Enum>> CardColor
PURPLE
RED
BLUE
GREEN
<<Enum>> PointsCalculatorType
FIXED_POINTS
NUMBER_OF_SYMBOLS
CORNERS_COVERED
<<Enum>> CardArrangement
L_SHAPE,
J_SHAPE,
R_SHAPE,
UPSIDE_DOWN_L_SHAPE,
DIAGONAL_TOP_LEFT_TO_BOTTOM_RIGHT,
DIAGONAL_BOTTOM_LEFT_TO_TOP_RIGHT
<<final>> Coordinates
-x: int
-y: int
+Coordinates(int, int): Coordinates
+equals(Coordinates): boolean
+hashCode(): int
Use
CardsTable
-cardsMap: Map<Coordinates, PlaceableCard>
+CardsTable(): CardsTable
+getCard(int x, int y): PlaceableCard
+placeCard(int x, int y, PlaceableCard card): void
+placeCard(int x, int y, GoldCard card): void
+checkCardIsPlaceable(int x, int y, PlaceableCard card): boolean
table 1
<<Abstract>> Card
-id: int
-isFaceUp: boolean
+Card(int): Card
+getId(): in
+isFaceUp(): boolean
+setIsFaceUp(boolean): void
Implements
cards
2500
<<Abstract>> GoalCard
-points: int
playerGoalCard 1
cards
<<final>> PointsCalculator
- pointsCalculatorType: PointsCalculatorType
- fixedPoints: int
- conditionSymbol: Symbol
+calculatePoints(Player): int
<<Abstract>> PlaceableCard
-cardColor: CardColor
-upsideCorners: Corner[4]
-downsideCorners: Corner[4]
-pointsCalculator: PointsCalculator
corners
4
Corner
-isSuperimposable: boolean
-symbol: Symbol
+Corner(): Corner
+Corner(Symbol): Corner
+isSuperimposable(): boolean
Player
-username: String
-placeholderColor: PlaceholderColor
-points: int
-hand: List<PlaceableCard>
-playerGoalCard: GoalCard
-symbolCounts: int[7]
-table: CardsTable
+Player(String): Player
+addPoints(int): void
+addCard(Card): void
+useCard(Card): void
+incrementSymbols(Symbol symbol, int amount): void
Implements
Implements
<<final>> GoalCardsArrangement
-arrangementType: CardArrangement
-colorRequirements: CardColor[3]
+CardsArrangement(int): CardsArrangement
<<final>> GoalSymbolSet
-requirement: List<Symbol>
+GoalSymbolSet(List<Symbol>): GoalSymbolSet
<<Final>> GoldCard
-prerequisites: int[4]
+GoldCard(int): GoldCard
+checkPrerequisites(Player): boolean
<<Final>> ResourceCard
+ResourceCard(int): ResourceCard
<<Final>> StartingCard
-downsideCenterResources: int[4]
+StartingCard(int): StartingCard
cards
upsideUpGold 2
cards
upsideUpResource 2
<<Final>> Message
-content: String
-sender: Player
-receiver: Player
+Message(String, Player, Player)
messages
Chat
-messages: List<Message>
+post(Message): void
GoalCardsDeck
-cards: List<GoalCard>
+GoalCardsDeck(): GoalCardsDeck
+drawCard(): GoalCard
GoldCardsDeck
-cards: List<GoldCard>
+GoldCardsDeck(): GoldCardsDeck
+drawCard(): GoldCard
ResourceCardsDeck
-cards: List<ResourceCard>
+ResourceCardsDeck(): ResourceCardsDeck
+drawCard(): ResourceCard
goalDeck 1
goldDeck 1
resourceDeck 1
GameModel
-gameModel: GameModel
-gameState: GameState
-turnPhase: TurnPhase
-round: int
-currentTurnPlayer: Player
-players: List<Player>
-goldDeck: GoldCardDeck
-resourceDeck: ResourceCardDeck
-goalDeck: GoalCardDeck
-upsideUpResource: ResourceCard[2]
-upsideUpGold: GoldCard[2]
-upsideUpGoal: GoalCard[2]
-chat: Chat
-winner: Player
+addPlayer(Player): void
+nextTurn(): void
+checkGoalCardRequirement(GoalCard, Player): boolean
+calculateWinner(): Player
GameController
+addPlayer(String): Player
+startGame(): void
-setupTable(): void
+choosePlayerGoalCard(Player, GoalCard): void
+placeCardFromHand(Player, int): boolean
+drawCardFromDeck(Player, Deck): void
+drawUpsideUpGold(Player, int): void
+drawUpsideUpResource(Player, int): void
+nextTurn(): void
+getCurrentTurnPlayer(): Player
+isEndgame(): boolean
+finishGame(): void
+getWinner(): Player
+postMessage(String, Player, Player): void
Implements
<<Abstract>> Deck
-remainingCards: int
+drawCard(): Card
+shuffle(): void