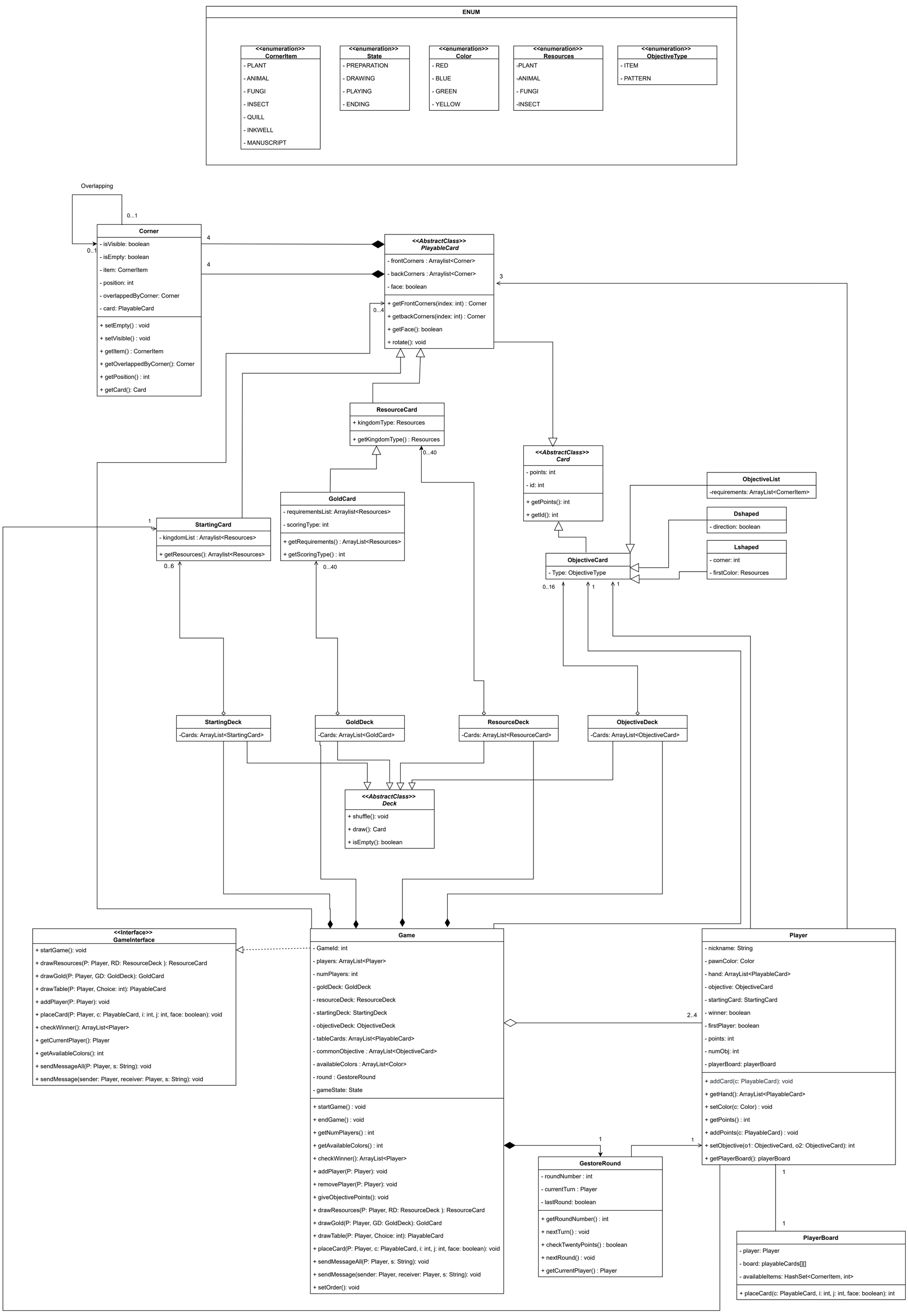
ENUM
<<enumeration>> CornerItem
- PLANT
- ANIMAL
- FUNGI
- INSECT
- QUILL
- INKWELL
- MANUSCRIPT
<<enumeration>> State
- PREPARATION
- DRAWING
- PLAYING
- ENDING
<<enumeration>> Color
- RED
- BLUE
- GREEN
- YELLOW
<<enumeration>> Resources
-PLANT
-ANIMAL
- FUNGI
-INSECT
<<enumeration>> ObjectiveType
- ITEM
- PATTERN
Overlapping
0...1
Corner
- isVisible: boolean
- isEmpty: boolean
- item: CornerItem
- position: int
- overlappedByCorner: Corner
- card: PlayableCard
+ setEmpty() : void
+ setVisible() : void
+ getItem() : CornerItem
+ getOverlappedByCorner(): Corner
+ getPosition() : int
+ getCard(): Card
0...1
4
4
<<AbstractClass>> PlayableCard
- frontCorners : Arraylist<Corner>
- backCorners : Arraylist<Corner>
- face: boolean
+ getFrontCorners(index: int) : Corner
+ getbackCorners(index: int) : Corner
+ getFace(): boolean
+ rotate(): void
0...4
3
ResourceCard
+ kingdomType: Resources
+ getKingdomType() : Resources
0...40
<<AbstractClass>> Card
- points: int
- id: int
+ getPoints(): int
+ getId(): int
ObjectiveList
-requirements: ArrayList<CornerItem>
Dshaped
- direction: boolean
Lshaped
- corner: int
- firstColor: Resources
GoldCard
- requirementsList: Arraylist<Resources>
- scoringType: int
+ getRequirements() : ArrayList<Resources>
+ getScoringType() : int
1
StartingCard
- kingdomList : Arraylist<Resources>
+ getResources(): Arraylist<Resources>
0..6
0...40
ObjectiveCard
- Type: ObjectiveType
0..16
1
1
StartingDeck
-Cards: ArrayList<StartingCard>
GoldDeck
-Cards: ArrayList<GoldCard>
ResourceDeck
-Cards: ArrayList<ResourceCard>
ObjectiveDeck
-Cards: ArrayList<ObjectiveCard>
<<AbstractClass>> Deck
+ shuffle(): void
+ draw(): Card
+ isEmpty(): boolean
<<Interface>> GameInterface
+ startGame(): void
+ drawResources(P: Player, RD: ResourceDeck): ResourceCard
+ drawGold(P: Player, GD: GoldDeck): GoldCard
+ drawTable(P: Player, Choice: int): PlayableCard
+ addPlayer(P: Player): void
+ placeCard(P: Player, c: PlayableCard, i: int, j: int, face: boolean): void
+ checkWinner(): ArrayList<Player>
+ getCurrentPlayer(): Player
+ getAvailableColors(): int
+ sendMessageAll(P: Player, s: String): void
+ sendMessage(sender: Player, receiver: Player, s: String): void
Game
- GameId: int
- players: ArrayList<Player>
- numPlayers: int
- goldDeck: GoldDeck
- resourceDeck: ResourceDeck
- startingDeck: StartingDeck
- objectiveDeck: ObjectiveDeck
- tableCards: ArrayList<PlayableCard>
- commonObjective : ArrayList<ObjectiveCard>
- availableColors : ArrayList<Color>
- round : GestoreRound
- gameState: State
+ startGame() : void
+ endGame() : void
+ getNumPlayers() : int
+ getAvailableColors() : int
+ checkWinner(): ArrayList<Player>
+ addPlayer(P: Player): void
+ removePlayer(P: Player): void
+ giveObjectivePoints(): void
+ drawResources(P: Player, RD: ResourceDeck): ResourceCard
+ drawGold(P: Player, GD: GoldDeck): GoldCard
+ drawTable(P: Player, Choice: int): PlayableCard
+ placeCard(P: Player, c: PlayableCard, i: int, j: int, face: boolean): void
+ sendMessageAll(P: Player, s: String): void
+ sendMessage(sender: Player, receiver: Player, s: String): void
+ setOrder(): void
2..4
Player
- nickname: String
- pawnColor: Color
- hand: ArrayList<PlayableCard>
- objective: ObjectiveCard
- startingCard: StartingCard
- winner: boolean
- firstPlayer: boolean
- points: int
- numObj: int
- playerBoard: playerBoard
+ addCard(c: PlayableCard): void
+ getHand(): ArrayList<PlayableCard>
+ setColor(c: Color) : void
+ getPoints() : int
+ addPoints(c: PlayableCard) : void
+ setObjective(o1: ObjectiveCard, o2: ObjectiveCard): int
+ getPlayerBoard(): playerBoard
1
1
1
GestoreRound
- roundNumber : int
- currentTurn : Player
- lastRound: boolean
+ getRoundNumber() : int
+ nextTurn() : void
+ checkTwentyPoints() : boolean
+ nextRound() : void
+ getCurrentPlayer() : Player
1
1
PlayerBoard
- player: Player
- board: playableCards[][]
- availableItems: HashSet<CornerItem, int>
+ placeCard(c: PlayableCard, i: int, j: int, face: boolean): int