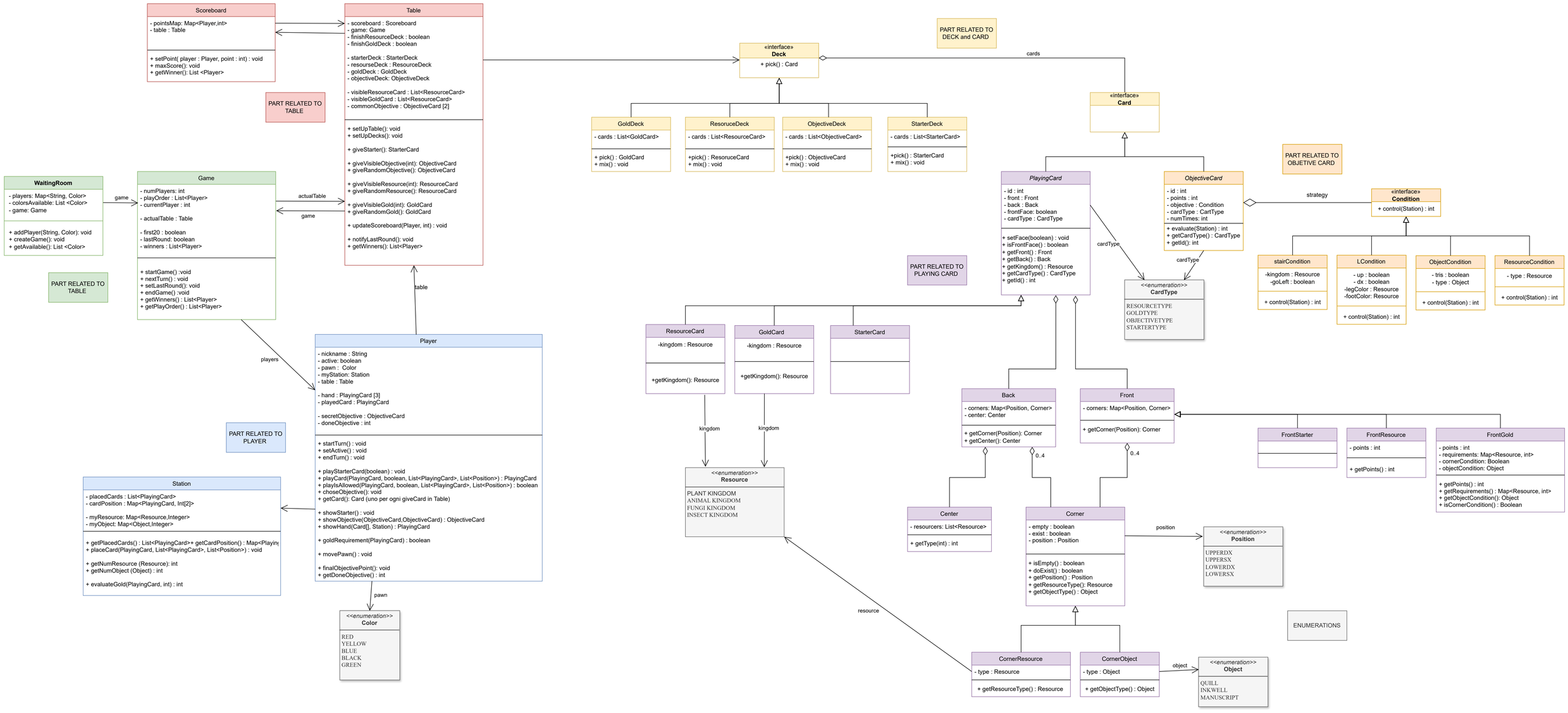
Scoreboard
- pointsMap: Map<Player,int>
- table : Table
+ setPoint(player : Player, point : int) : void
+ maxScore(): void
+ getWinner(): List <Player>
Table
- scoreboard : Scoreboard
- game: Game
- finishResourceDeck : boolean
- finishGoldDeck : boolean
- starterDeck : StarterDeck
- resourseDeck : ResourceDeck
- goldDeck : GoldDeck
- objectiveDeck: ObjectiveDeck
- visibleResourceCard : List<ResourceCard>
- visibleGoldCard : List<ResourceCard>
- commonObjective : ObjectiveCard [2]
+ setUpTable(): void
+ setUpDecks(): void
+ giveStarter(): StarterCard
+ giveVisibleObjective(int): ObjectiveCard
+ giveRandomObjective(): ObjectiveCard
+ giveVisibleResource(int): ResourceCard
+ giveRandomResource(): ResourceCard
+ giveVisibleGold(int): GoldCard
+ giveRandomGold(): GoldCard
+ updateScoreboard(Player, int) : void
+ notifyLastRound(): void
+ getWinners(): List<Player>
PART RELATED TO TABLE
WaitingRoom
- players: Map<String, Color>
- colorsAvailable: List <Color>
- game: Game
+ addPlayer(String, Color): void
+ createGame(): void
+ getAvailable(): List <Color>
game
Game
- numPlayers: int
- playOrder : List<Player>
- currentPlayer : int
- actualTable : Table
- first20 : boolean
- lastRound: boolean
- winners : List<Player>
+ startGame() :void
+ nextTurn() : void
+ setLastRound(): void
+ endGame() :void
+ getWinners() : List<Player>
+ getPlayOrder() : List<Player>
actualTable
game
table
PART RELATED TO TABLE
players
Player
- nickname : String
- active: boolean
- pawn : Color
- myStation: Station
- table : Table
- hand : PlayingCard [3]
- playedCard : PlayingCard
- secretObjective : ObjectiveCard
- doneObjective : int
+ startTurn() : void
+ setActive() : void
+ endTurn() : void
+ playStarterCard(boolean) : void
+ playCard(PlayingCard, boolean, List<PlayingCard>, List<Position>) : PlayingCard
+ playIsAllowed(PlayingCard, boolean, List<PlayingCard>, List<Position>) : boolean
+ choseObjective(): void
+ getCard(): Card (uno per ogni giveCard in Table)
+ showStarter() : void
+ showObjective(ObjectiveCard,ObjectiveCard) : ObjectiveCard
+ showHand(Card[], Station) : PlayingCard
+ goldRequirement(PlayingCard) : boolean
+ movePawn() : void
+ finalObjectivePoint(): void
+ getDoneObjective() : int
PART RELATED TO PLAYER
Station
- placedCards : List<PlayingCard>
- cardPosition : Map<PlayingCard, Int[2]>
- myResource: Map<Resource,Integer>
- myObject: Map<Object,Integer>
+ getPlacedCards() : List<PlayingCard>+ getCardPosition() : Map<Playin
+ placeCard(PlayingCard, List<PlayingCard>, List<Position>) : void
+ getNumResource (Resource): int
+ getNumObject (Object) : int
+ evaluateGold(PlayingCard, int) : int
pawn
<<enumeration>>
Color
RED
YELLOW
BLUE
BLACK
GREEN
PART RELATED TO DECK and CARD
«interface»
Deck
+ pick() : Card
cards
GoldDeck
- cards : List<GoldCard>
+ pick() : GoldCard
+ mix() : void
ResoruceDeck
- cards : List<ResourceCard>
+pick() : ResoruceCard
+ mix() : void
ObjectiveDeck
- cards : List<ObjectiveCard>
+pick() : ObjectiveCard
+ mix() : void
StarterDeck
- cards : List<StarterCard>
+pick() : StarterCard
+ mix() : void
«interface»
Card
PlayingCard
- id : int
- front : Front
- back : Back
- frontFace: boolean
- cardType : CardType
+ setFace(boolean) : void
+ isFrontFace() : boolean
+ getFront() : Front
+ getBack() : Back
+ getKingdom() : Resource
+ getCardType() : CardType
+ getId() : int
ObjectiveCard
- id : int
- points : int
- objective : Condition
- cardType : CartType
- numTimes: int
+ evaluate(Station) : int
+ getCardType() : CardType
+ getId(): int
PART RELATED TO OBJETIVE CARD
strategy
«interface»
Condition
+ control(Station) : int
stairCondition
-kingdom : Resource
-goLeft : boolean
+ control(Station) : int
LCondition
- up : boolean
- dx : boolean
-legColor : Resource
-footColor: Resource
+ control(Station) : int
ObjectCondition
- tris : boolean
- type : Object
+ control(Station) : int
ResourceCondition
- type : Resource
+ control(Station) : int
cardType
cardType
<<enumeration>>
CardType
RESOURCETYPE
GOLDTYPE
OBJECTIVETYPE
STARTERTYPE
PART RELATED TO PLAYING CARD
ResourceCard
-kingdom : Resource
+getKingdom(): Resource
GoldCard
-kingdom : Resource
+getKingdom(): Resource
StarterCard
kingdom
kingdom
<<enumeration>>
Resource
PLANT KINGDOM
ANIMAL KINGDOM
FUNGI KINGDOM
INSECT KINGDOM
Back
- corners: Map<Position, Corner>
- center: Center
+ getCorner(Position): Corner
+ getCenter(): Center
Front
- corners: Map<Position, Corner>
+ getCorner(Position): Corner
0..4
0..4
FrontStarter
FrontResource
- points : int
+ getPoints() : int
FrontGold
- points : int
- requirements: Map<Resource, int>
- cornerCondition: Boolean
- objectCondition: Object
+ getPoints() : int
+ getRequirements() : Map<Resource, int>
+ getObjectCondition(): Object
+ isCornerCondition() : Boolean
Center
- resourcers: List<Resource>
+ getType(int) : int
Corner
- empty : boolean
- exist : boolean
- position : Position
+ isEmpty() : boolean
+ doExist() : boolean
+ getPosition() : Position
+ getResourceType(): Resource
+ getObjectType() : Object
position
<<enumeration>>
Position
UPPERDX
UPPERSX
LOWERDX
LOWERSX
resource
ENUMERATIONS
CornerResource
- type : Resource
+ getResourceType() : Resource
CornerObject
- type : Object
+ getObjectType() : Object
object
<<enumeration>>
Object
QUILL
INKWELL
MANUSCRIPT