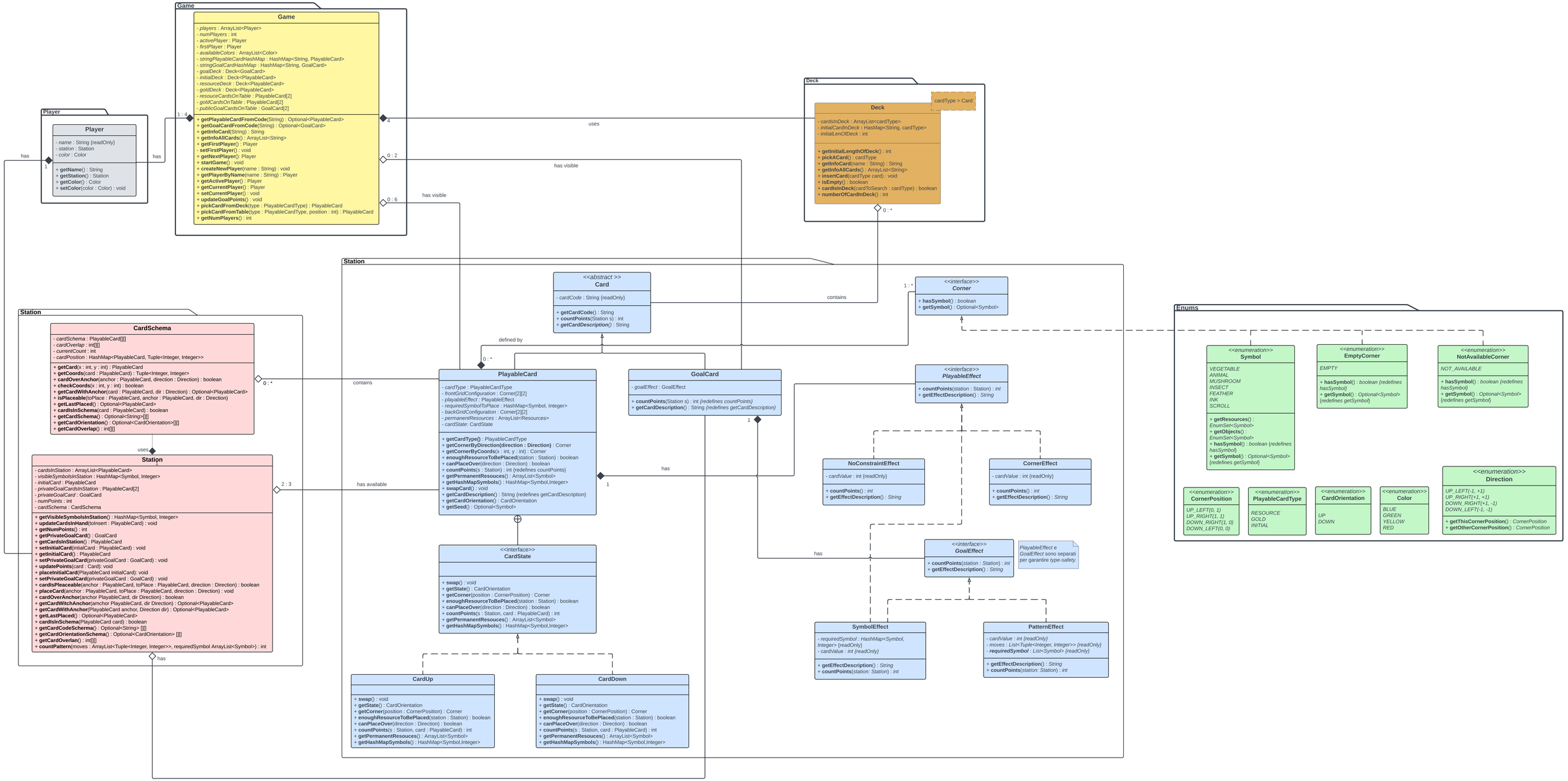
Game
Game
- players : ArrayList<Player>
- numPlayers : int
- activePlayer : Player
- firstPlayer : Player
- availableColors : ArrayList<Color>
- stringPlayableCardHashMap : HashMap<String, PlayableCard>
- stringGoalCardHashMap : HashMap<String, GoalCard>
- goalDeck : Deck<GoalCard>
- initialDeck : Deck<PlayableCard>
- resourceDeck : Deck<PlayableCard>
- goldDeck : Deck<PlayableCard>
- resoucecardsOnTable : PlayableCard[2]
- goldCardsOnTable : PlayableCard[2]
- publicGoalCardsOnTable : GoalCard[2]
+ getPlayableCardFromCode(String) : Optional<PlayableCard>
+ getGoalCardFromCode(String) : Optional<GoalCard>
+ getInfoCard(String) : String
+ getInfoAllCards() : ArrayList<String>
+ getFirstPlayer() : Player
- setFirstPlayer() : void
+ getNextPlayer() : Player
+ startGame() : void
+ createNewPlayer(name : String) : void
+ getPlayerByName(name : String) : Player
+ getActivePlayer() : Player
+ getCurrentPlayer() : Player
+ setCurrentPlayer() : void
+ updateGoalPoints() : void
+ pickCardFromDeck(type : PlayableCardType) : PlayableCard
+ pickCardFromTable(type : PlayableCardType, position : int) : PlayableCard
+ getNumPlayers() : int
Player
Player
- name : String {readOnly}
- station : Station
- color : Color
+ getName() : String
+ getStation() : Station
+ getColor() : Color
+ setColor(color : Color) : void
has
has
1 : 4
1
4
uses
Deck
Deck
cardType > Card
- cardsInDeck : ArrayList<cardType>
- initialCardInDeck : HasMap<String, cardType>
- initialLenOfDeck : int
+ getInitialLengthOfDeck() : int
+ pickACard() : cardType
+ getInfoCard(name : String) : String
+ getInfoAllCards() : ArrayList<String>
+ insertCard(cardType card) : void
+ isEmpty() : boolean
+ cardIsInDeck(cardToSearch : cardType) : boolean
+ numberOfCardInDeck() : int
0 : 2
has visible
0 : 6
has visible
0 : *
Station
Station
CardSchema
- cardSchema : PlayableCard[][]
- cardOverlap : int[][]
- currentCount : int
- cardPosition : HashMap<PlayableCard, Tuple<Integer, Integer>>
+ getCard(x : int, y : int) : PlayableCard
+ getCoords(card : PlayableCard) : Tuple<Integer, Integer>
+ cardOverAnchor(anchor : PlayableCard, direction : Direction) : boolean
+ checkCoords(x : int, y : int) : boolean
+ getCardWithAnchor(card : PlayableCard, dir : Direction) : Optional<PlayableCard>
+ isPlaceable(toPlace : PlayableCard, anchor : PlayableCard, dir : Direction)
+ getLastPlaced() : Optional<PlayableCard>
+ cardIsInSchema(card : PlayableCard) : boolean
+ getCardSchema() : Optional<String>[][]
+ getCardOrientation() : Optional<CardOrientation>[][]
+ getCardOverlap() : int[][]
0 : *
contains
uses
Station
- cardsInStation : ArrayList<PlayableCard>
- visibleSymbolsInStation : HashMap<Symbol, Integer>
- initialCard : PlayableCard
- privateGoalCardsInStation : PlayableCard[2]
- privateGoalCard : GoalCard
- numPoints : int
- cardSchema : CardSchema
+ getVisibleSymbolsInStation() : HashMap<Symbol, Integer>
+ updateCardsInHand(toInsert : PlayableCard) : void
+ getNumPoints() : int
+ getPrivateGoalCard() : GoalCard
+ getCardsInStation() : PlayableCard
+ setInitialCard(initialCard : PlayableCard) : void
+ getInitialCard() : PlayableCard
+ setPrivateGoalCard(privateGoalCard : GoalCard) : void
+ updatePoints(card : Card): void
+ placeInitialCard(PlayableCard initialCard): void
+ setPrivateGoalCard(privateGoalCard : GoalCard) : void
+ cardIsPleaceable(anchor : PlayableCard, toPlace : PlayableCard, direction : Direction) : boolean
+ placeCard(anchor : PlayableCard, toPlace : PlayableCard, direction : Direction) : void
+ cardOverAnchor(anchor PlayableCard, dir Direction) : boolean
+ getCardWitchAnchor(amchor PlayableCard, dir Direction) : Optional<PlayableCard>
+ getCardWithAnchor(PlayableCard anchor, Direction dir) : Optional<PlayableCard>
+ getLastPlaced() : Optional<PlayableCard>
+ cardIsInSchema(PlayableCard card) : boolean
+ getCardCodeSchema() : Optional<String> [][]
+ getCardOrientationSchema() : Optional<CardOrientation> [][]
+ getCardOverlan() : int[][]
+ countPattern(moves : ArrayList<Tuple<Integer, Integer>>, requiredSymbol ArrayList<Symbol>) : int
2 : 3
has available
has
<<abstract >>
Card
- cardCode : String {readOnly}
+ getCardCode() : String
+ countPoints(Station s) : int
+ getCardDescription() : String
contains
1 : *
<<interface>>
Corner
+ hasSymbol() : boolean
+ getSymbol() : Optional<Symbol>
defined by
0 : *
PlayableCard
- cardType : PlayableCardType
- frontGridConfiguration : Corner[2][2]
- playableEffect : PlayableEffect
- requiredSymbolToPlace : HashMap<Symbol, Integer>
- backGridConfiguration : Corner[2][2]
- permanentResources : ArrayList<Resources>
- cardState: CardState
+ getCardType() : PlayableCardType
+ getCornerByDirection(direction : Direction) : Corner
+ getCornerByCoords(x : int, y : int) : Corner
+ enoughResourceToBePlaced(station : Station) : boolean
+ canPlaceOver(direction : Direction) : boolean
+ countPoints(s : Station) : int {redefines countPoints}
+ getPermanentResouces() : ArrayList<Symbol>
+ getHashMapSymbols() : HashMap<Symbol,Integer>
+ swapCard() : void
+ getCardDescription() : String {redefines getCardDescription}
+ getCardOrientation() : CardOrientation
+ getSeed() : Optional<Symbol>
GoalCard
- goalEffect : GoalEffect
+ countPoints(Station s) : int {redefines countPoints}
+ getCardDescription() : String {redefines getCardDescription}
1
1
has
<<interface>>
PlayableEffect
+ countPoints(station : Station) : int
+ getEffectDescription() : String
NoConstraintEffect
- cardValue : int {readOnly}
+ countPoints() : int
+ getEffectDescription() : String
CornerEffect
- cardValue : int {readOnly}
+ countPoints() : int
+ getEffectDescription() : String
<<interface>>
CardState
+ swap() : void
+ getState() : CardOrientation
+ getCorner(position : CornerPosition) : Corner
+ enoughResourceToBePlaced(station : Station) : boolean
+ canPlaceOver(direction : Direction) : boolean
+ countPoints(s : Station, card : PlayableCard) : int
+ getPermanentResouces() : ArrayList<Symbol>
+ getHashMapSymbols() : HashMap<Symbol,Integer>
has
<<interface>>
GoalEffect
+ countPoints(station : Station) : int
+ getEffectDescription() : String
PlayableEffect e GoalEffect sono separati per garantire type-safety.
SymbolEffect
- requiredSymbol : HashMap<Symbol, Integer> {readOnly}
- cardValue : int {readOnly}
+ getEffectDescription() : String
+ countPoints(station: Station) : int
PatternEffect
- cardValue : int {readOnly}
- moves : List<Tuple<Integer, Integer>> {readOnly}
- requiredSymbol : List<Symbol> {readOnly}
+ getEffectDescription() : String
+ countPoints(station: Station) : int
CardUp
+ swap() : void
+ getState() : CardOrientation
+ getCorner(position : CornerPosition) : Corner
+ enoughResourceToBePlaced(station : Station) : boolean
+ canPlaceOver(direction : Direction) : boolean
+ countPoints(s : Station, card : PlayableCard) : int
+ getPermanentResouces() : ArrayList<Symbol>
+ getHashMapSymbols() : HashMap<Symbol,Integer>
CardDown
+ swap() : void
+ getState() : CardOrientation
+ getCorner(position : CornerPosition) : Corner
+ enoughResourceToBePlaced(station : Station) : boolean
+ canPlaceOver(direction : Direction) : boolean
+ countPoints(s : Station, card : PlayableCard) : int
+ getPermanentResouces() : ArrayList<Symbol>
+ getHashMapSymbols() : HashMap<Symbol,Integer>
Enums
<<enumeration>>
Symbol
VEGETABLE
ANIMAL
MUSHROOM
INSECT
FEATHER
INK
SCROLL
+ getResources() : EnumSet<Symbol>
+ getObjects() : EnumSet<Symbol>
+ hasSymbol() : boolean {redefines hasSymbol}
+ getSymbol() : Optional<Symbol> {redefines getSymbol}
<<enumeration>>
EmptyCorner
EMPTY
+ hasSymbol() : boolean {redefines hasSymbol}
+ getSymbol() : Optional<Symbol> {redefines getSymbol}
<<enumeration>>
NotAvailableCorner
NOT_AVAILABLE
+ hasSymbol() : boolean {redefines hasSymbol}
+ getSymbol() : Optional<Symbol> {redefines getSymbol}
<<enumeration>>
CornerPosition
UP_LEFT(0, 1)
UP_RIGHT(1, 1)
DOWN_RIGHT(1, 0)
DOWN_LEFT(0, 0)
<<enumeration>>
PlayableCardType
RESOURCE
GOLD
INITIAL
<<enumeration>>
CardOrientation
UP
DOWN
<<enumeration>>
Color
BLUE
GREEN
YELLOW
RED
<<enumeration>>
Direction
UP_LEFT(-1, +1)
UP_RIGHT(+1, +1)
DOWN_RIGHT(+1, -1)
DOWN_LEFT(-1, -1)
+ getThisCornerPosition() : CornerPosition
+ getOtherCornerPosition() : CornerPosition