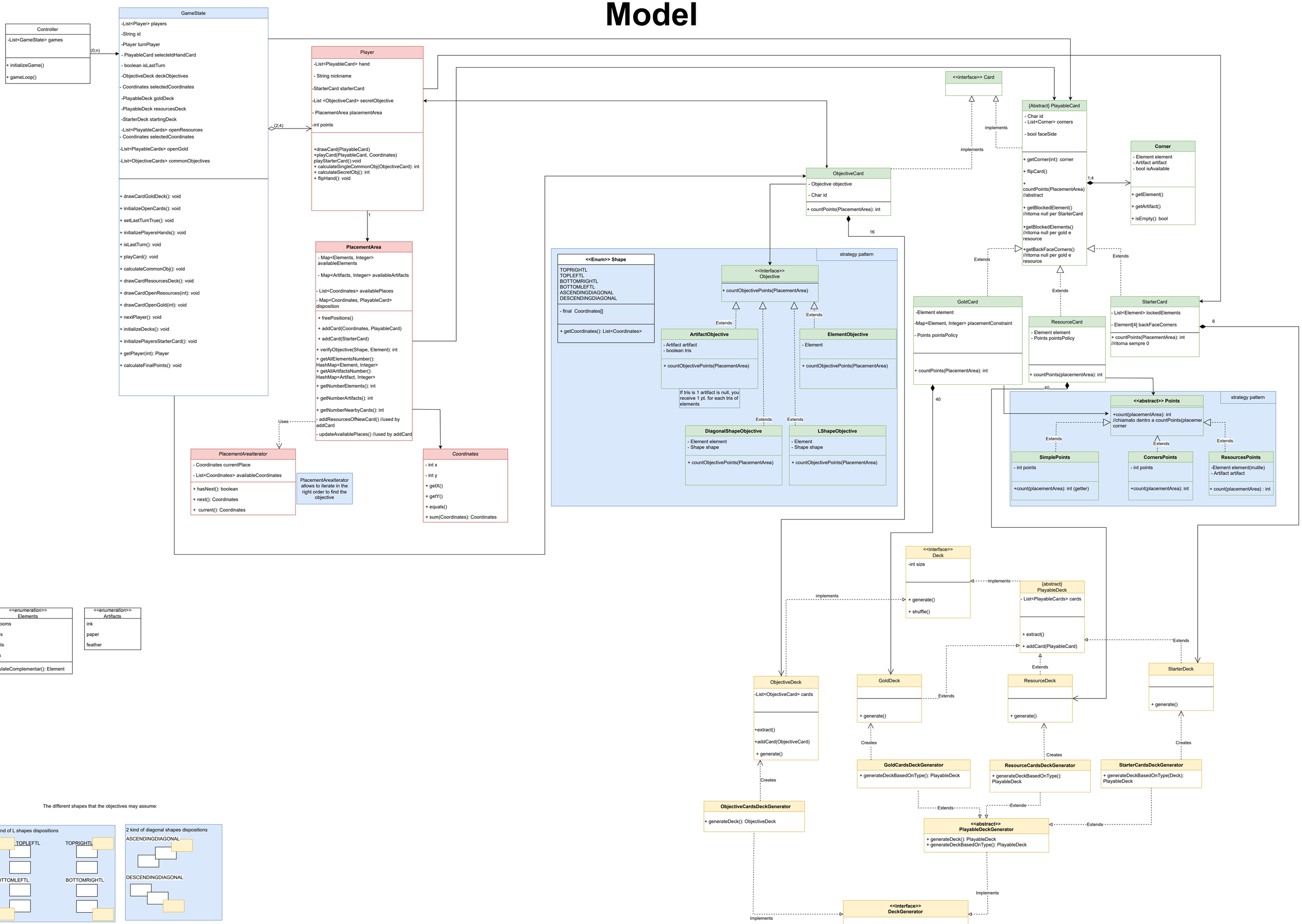
# Model


Controller
-List<GameState> games
+ initializeGame()
+ gameLoop()
(0;n)
GameState
-List<Player> players
-String id
-Player turnPlayer
- PlayableCard selectetdHandCard
- boolean isLastTurn
-ObjectiveDeck deckObjectives
- Coordinates selectedCoordinates
-PlayableDeck goldDeck
-PlayableDeck resourcesDeck
-StarterDeck startingDeck
-List<PlayableCards> openResources
- Coordinates selectedCoordinates
-List<PlayableCards> openGold
-List<ObjectiveCards> commonObjectives
+ drawCardGoldDeck(): void
+ initializeOpenCards(): void
+ setLastTurnTrue(): void
+ initializePlayersHands(): void
+ isLastTurn(): void
+ playCard(): void
+ calculateCommonObj(): void
+ drawCardResourcesDeck(): void
+ drawCardOpenResources(int): void
+ drawCardOpenGold(int): void
+ nextPlayer(): void
+ initializeDecks(): void
+ initializePlayersStarterCard(): void
+ getPlayer(int): Player
+ calculateFinalPoints(): void
(2;4)
Player
-List<PlayableCard> hand
- String nickname
-StarterCard starterCard
-List <ObjectiveCard> secretObjective
- PlacementArea placementArea
-int points
+drawCard(PlayableCard)
+playCard(PlayableCard, Coordinates)
playStarterCard():void
+ calculateSingleCommonObj(ObjectiveCard): int
+ calculateSecretObj(): int
+ flipHand(): void
1
PlacementArea
- Map<Elements, Integer> availableElements
- Map<Artifacts, Integer> availableArtifacts
- List<Coordinates> availablePlaces
- Map<Coordinates, PlayableCard> disposition
+ freePositions()
+ addCard(Coordinates, PlayableCard)
+ addCard(StarterCard)
+ verifyObjective(Shape, Element): int
+ getAllElementsNumber(): HashMap<Element, Integer>
+ getAllArtifactsNumber(): HashMap<Artifact, Integer>
+ getNumberElements(): int
+ getNumberArtifacts(): int
+ getNumberNearbyCards(): int
- addResourcesOfNewCard() //used by addCard
- updateAvailablePlaces() //used by addCard
Uses
PlacementAreaIterator
- Coordinates currentPlace
- List<Coordinates> availableCoordinates
+ hasNext(): boolean
+ next(): Coordinates
+ current(): Coordinates
PlacementAreaIterator allows to iterate in the right order to find the objective
Coordinates
- int x
- int y
+ getX()
+ getY()
+ equals()
+ sum(Coordinates): Coordinates
<<interface>> Card
implements
{Abstract} PlayableCard
- Char id
- List<Corner> corners
- bool faceSide
+ getCorner(int): corner
+ flipCard()
+ countPoints(PlacementArea) //abstract
+ getBlockedElement() //ritorna null per StarterCard
+getBlockedElements() //ritorna null per gold e resource
+getBackFaceCorners() //ritorna null per gold e resource
1:4
Corner
- Element element
- Artifact artifact
- bool isAvailable
+ getElement()
+ getArtifact()
+ isEmpty(): bool
ObjectiveCard
- Objective objective
- Char id
+ countPoints(PlacementArea): int
16
strategy pattern
<<Enum>> Shape
TOPRIGHTL
TOPLEFTL
BOTTOMRIGHTL
BOTTOMLEFTL
ASCENDINGDIAGONAL
DESCENDINGDIAGONAL
- final Coordinates[]
+ getCoordinates(): List<Coordinates>
<<Interface>> Objective
+ countObjectivePoints(PlacementArea)
Extends
ArtifactObjective
- Artifact artifact
- boolean tris
+ countObjectivePoints(PlacementArea)
If tris is 1 artifact is null, you receive 1 pt. for each tris of elements
ElementObjective
- Element
+ countObjectivePoints(PlacementArea)
DiagonalShapeObjective
- Element element
- Shape shape
+ countObjectivePoints(PlacementArea)
LShapeObjective
- Element
- Shape shape
+ countObjectivePoints(PlacementArea)
Extends
GoldCard
-Element element
-Map<Element, Integer> placementConstraint
- Points pointsPolicy
+ countPoints(PlacementArea): int
ResourceCard
- Element element
- Points pointsPolicy
+ countPoints(placementArea): int
StarterCard
- List<Element> lockedElements
- Element[4] backFaceCorners
+ countPoints(PlacementArea): int //ritorna sempre 0
6
40
40
strategy pattern
<<abstract>> Points
+count(placementArea): int //chiamato dentro a countPoints(placemer corner
SimplePoints
- int points
+count(placementArea): int (getter)
CornersPoints
- int points
+count(placementArea): int
ResourcesPoints
-Element element(inutile)
- Artifact artifact
+ count(placementArea) : int
<<enumeration>> Elements
mushrooms
animals
vegetals
insects
+ calculateComplementar(): Element
<<enumeration>> Artifacts
ink
paper
feather
The different shapes that the objectives may assume:
4 kind of L shapes dispositions
TOPLEFTL
TOPRIGHTL
BOTTOMLEFTL
BOTTOMRIGHTL
2 kind of diagonal shapes dispositions
ASCENDINGDIAGONAL
DESCENDINGDIAGONAL
<<interface>> Deck
-int size
+ generate()
+ shuffle()
{abstract} PlayableDeck
- List<PlayableCards> cards
+ extract()
+ addCard(PlayableCard)
ObjectiveDeck
-List<ObjectiveCard> cards
+extract()
+addCard(ObjectiveCard)
+ generate()
GoldDeck
+ generate()
ResourceDeck
+ generate()
StarterDeck
+ generate()
Creates
GoldCardsDeckGenerator
+ generateDeckBasedOnType(): PlayableDeck
ResourceCardsDeckGenerator
+ generateDeckBasedOnType(): PlayableDeck
StarterCardsDeckGenerator
+ generateDeckBasedOnType(Deck): PlayableDeck
ObjectiveCardsDeckGenerator
+ generateDeck(): ObjectiveDeck
<<abstract>> PlayableDeckGenerator
+ generateDeck(): PlayableDeck
+ generateDeckBasedOnType(): PlayableDeck
Implements
<<interface>> DeckGenerator