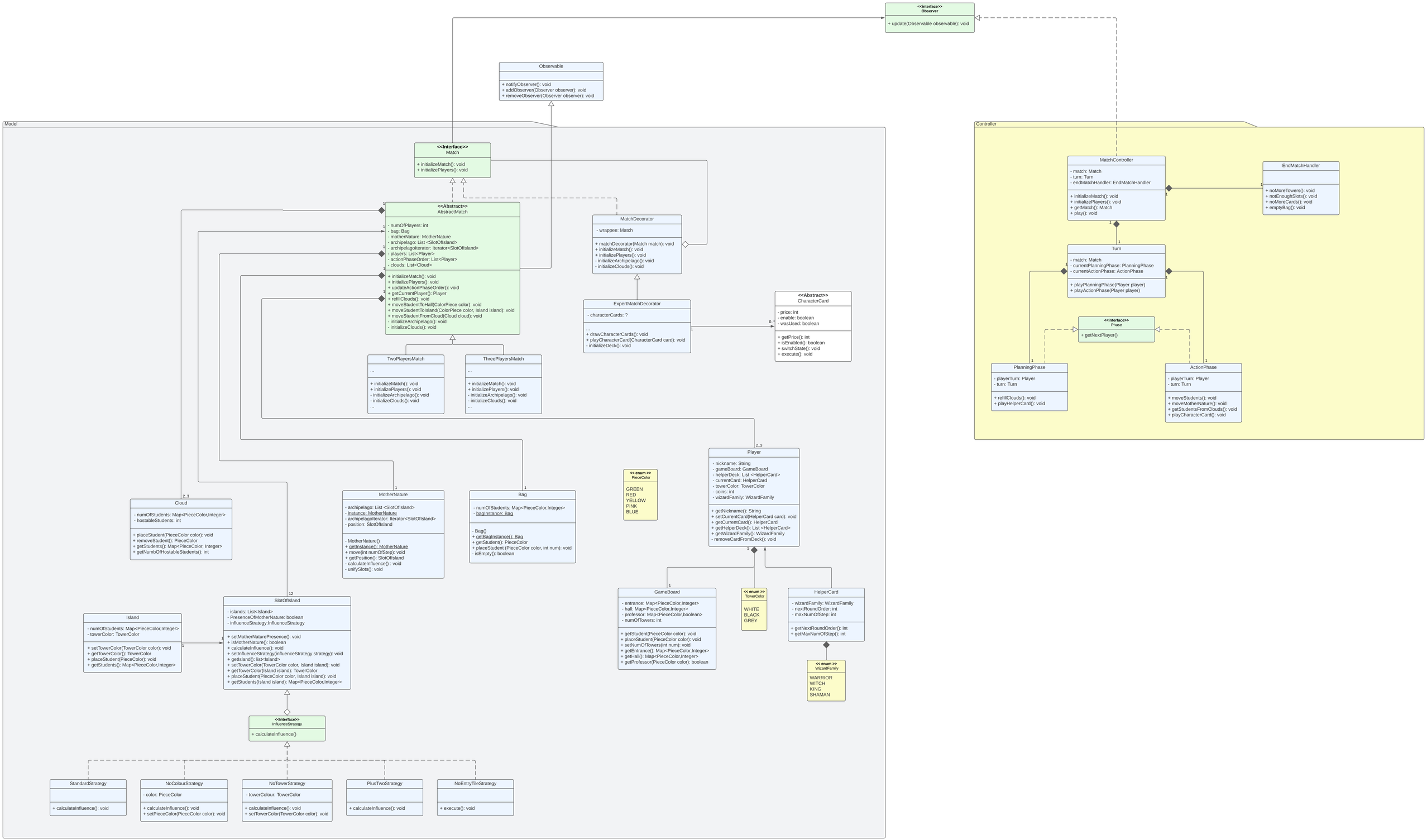
Observer
+ update(Observable observable): void
Observable
+ notifyObserver(): void
+ addObserver(Observer observer): void
+ removeObserver(Observer observer): void
Model
Controller
<<Interface>>
Match
+ initializeMatch(): void
+ initializePlayers(): void
<<Abstract>>
AbstractMatch
- numOfPlayers: int
- bag: Bag
- motherNature: MotherNature
- archipelago: List <SlotOfIsland>
- archipelagoIterator: Iterator<SlotOfIsland>
- players: List<Player>
- actionPhaseOrder: List<Player>
- clouds: List<Cloud>
+ initializeMatch(): void
+ initializePlayers(): void
+ updateActionPhaseOrder(): void
+ getCurrentPlayer(): Player
+ refillClouds(): void
+ moveStudentToHall(ColorPiece color): void
+ moveStudentToIsland(ColorPiece color, Island island): void
+ moveStudentFromCloud(Cloud cloud): void
- initializeArchipelago(): void
- initializeClouds(): void
MatchDecorator
- wrappee: Match
+ matchDecorator(Match match): void
+ initializeMatch(): void
+ initializePlayers(): void
- initializeArchipelago(): void
- initializeClouds(): void
ExpertMatchDecorator
- characterCards: ?
...
+ drawCharacterCards(): void
+ playCharacterCard(CharacterCard card): void
- initializeDeck(): void
<<Abstract>>
CharacterCard
- price: int
- enable: boolean
- wasUsed: boolean
+ getPrice(): int
+ isEnabled(): boolean
+ switchState(): void
+ execute(): void
TwoPlayersMatch
...
+ initializeMatch(): void
+ initializePlayers(): void
- initializeArchipelago(): void
- initializeClouds(): void
...
ThreePlayersMatch
...
+ initializeMatch(): void
+ initializePlayers(): void
- initializeArchipelago(): void
- initializeClouds(): void
...
Cloud
- numOfStudents: Map<PieceColor,Integer>
- hostableStudents: int
+ placeStudent(PieceColor color): void
+ removeStudent(): PieceColor
+ getStudents(): Map<PieceColor, Integer>
+ getNumbOfHostableStudents(): int
MotherNature
- archipelago: List <SlotOfIsland>
- instance: MotherNature
- archipelagoIterator: Iterator<SlotOfIsland>
- position: SlotOfIsland
- MotherNature()
+ getInstance(): MotherNature
+ move(int numOfStep): void
+ getPosition(): SlotOfIsland
- calculateInfluence() : void
- unifySlots(): void
Bag
- numOfStudents: Map<PieceColor,Integer>
- bagInstance: Bag
- Bag()
+ getBagInstance(): Bag
+ getStudent(): PieceColor
+ placeStudent (PieceColor color, int num): void
- isEmpty(): boolean
Island
- numOfStudents: Map<PieceColor,Integer>
- towerColor: TowerColor
+ setTowerColor(TowerColor color): void
+ getTowerColor(): TowerColor
+ placeStudent(PieceColor): void
+ getStudents(): Map<PieceColor,Integer>
SlotOfIsland
- islands: List<Island>
- PresenceOfMotherNature: boolean
- influenceStrategy:InfluenceStrategy
+ setMotherNaturePresence(): void
+ isMotherNature(): boolean
+ calculateInfluence(): void
+ setinfluenceStrategy(influenceStrategy strategy): void
+ getIsland(): list<Island>
+ setTowerColor(TowerColor color, Island island): void
+ getTowerColor(Island island): TowerColor
+ placeStudent(PieceColor color, Island island): void
+ getStudents(Island island): Map<PieceColor,Integer>
<<Interface>>
InfluenceStrategy
+ calculateInfluence()
StandardStrategy
+ calculateInfluence(): void
NoColourStrategy
- color: PieceColor
+ calculateInfluence(): void
+ setPieceColor(PieceColor color): void
NoTowerStrategy
- towerColour: TowerColor
+ calculateInfluence(): void
+ setTowerColor(TowerColor color): void
PlusTwoStrategy
+ calculateInfluence(): void
NoEntryTileStrategy
+ execute(): void
<< enum >>
PieceColor
GREEN
RED
YELLOW
PINK
BLUE
Player
- nickname: String
- gameBoard: GameBoard
- helperDeck: List <HelperCard>
- currentCard: HelperCard
- towerColor: TowerColor
- coins: int
- wizardFamily: WizardFamily
+ getNickname(): String
+ setCurrentCard(HelperCard card): void
+ getCurrentCard(): HelperCard
+ getHelperDeck(): List <HelperCard>
+ getWizardFamily(): WizardFamily
- removeCardFromDeck(): void
GameBoard
- entrance: Map<PieceColor,Integer>
- hall: Map<PieceColor,Integer>
- professor: Map<PieceColor,boolean>
- numOfTowers: int
+ getStudent(PieceColor color): void
+ placeStudent(PieceColor color): void
+ setNumOfTowers(int num): void
+ getEntrance(): Map<PieceColor,Integer>
+ getHall(): Map<PieceColor,Integer>
+ getProfessor(PieceColor color): boolean
<< enum >>
TowerColor
WHITE
BLACK
GREY
HelperCard
- wizardFamily: WizardFamily
- nextRoundOrder: int
- maxNumOfStep: int
+ getNextRoundOrder(): int
+ getMaxNumOfStep(): int
<< enum >>
WizardFamily
WARRIOR
WITCH
KING
SHAMAN
MatchController
- match: Match
- turn: Turn
- endMatchHandler: EndMatchHandler
+ initializeMatch(): void
+ initializePlayers(): void
+ getMatch(): Match
+ play(): void
EndMatchHandler
+ noMoreTowers(): void
+ notEnoughSlots(): void
+ noMoreCards(): void
+ emptyBag(): void
Turn
- match: Match
- currentPlanningPhase: PlanningPhase
- currentActionPhase: ActionPhase
+ playPlanningPhase(Player player)
+ playActionPhase(Player player)
<<interface>>
Phase
+ getNextPlayer()
PlanningPhase
- playerTurn: Player
- turn: Turn
+ refillClouds(): void
+ playHelperCard(): void
ActionPhase
- playerTurn: Player
- turn: Turn
+ moveStudents(): void
+ moveMotherNature(): void
+ getStudentsFromClouds(): void
+ playCharacterCard(): void