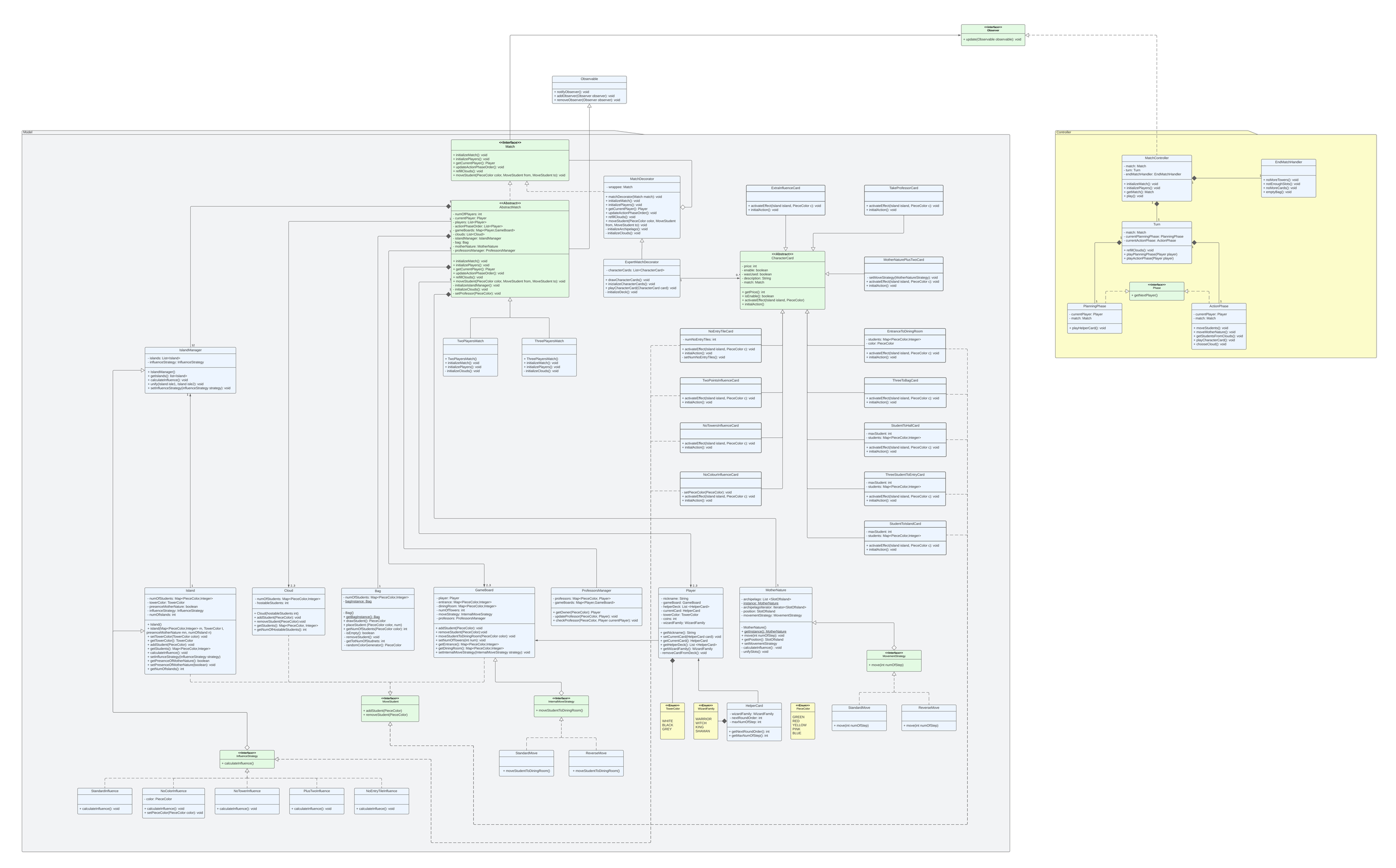
«interface»
Observer
+ update(Observable observable): void
Observable
+ notifyObserver(): void
+ addObserver(Observer observer): void
+ removeObserver(Observer observer): void
Model
«interface»
Match
+ initializeMatch(): void
+ initializePlayers(): void
+ getCurrentPlayer(): Player
+ updateActionPhaseOrder(): void
+ refillClouds(): void
+ moveStudent(PieceColor color, MoveStudent from, MoveStudent to): void
MatchDecorator
- wrappee: Match
+ matchDecorator(Match match): void
+ initializeMatch(): void
+ initializePlayers(): void
+ getCurrentPlayer(): Player
+ updateActionPhaseOrder(): void
+ refillClouds(): void
+ moveStudent(PieceColor color, MoveStudent from, MoveStudent to): void
- initializeArchipelago(): void
- initializeClouds(): void
«<Abstract>>
AbstractMatch
- numOfPlayers: int
- currentPlayer: Player
- players: List<Player>
- actionPhaseOrder: List<Player>
- gameBoards: Map<Player,GameBoard>
- clouds: List<Cloud>
- islandManager: IslandManager
- bag: Bag
- motherNature: MotherNature
- professorsManager: ProfessorsManager
+ initializeMatch(): void
+ initializePlayers(): void
+ getCurrentPlayer(): Player
+ updateActionPhaseOrder(): void
+ refillClouds(): void
+ moveStudent(PieceColor color, MoveStudent from, MoveStudent to): void
- initializeIslandManager(): void
- initializeClouds(): void
- setProfessor(PieceColor): void
ExpertMatchDecorator
- characterCards: List<CharacterCard>
+ drawCharacterCards(): void
+ inizializeCharacterCards(): void
+ playCharacterCard(CharacterCard card): void
- initializeDeck(): void
TwoPlayersMatch
+ TwoPlayersMatch()
+ initializeMatch(): void
+ initializePlayers(): void
+ initializeClouds(): void
ThreePlayersMatch
+ ThreePlayersMatch()
+ initializeMatch(): void
+ initializePlayers(): void
+ initializeClouds(): void
ExtraInfluenceCard
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
TakeProfessorCard
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
«<Abstract>>
CharacterCard
- price: int
- enable: boolean
- wasUsed: boolean
- description: String
- match: Match
+ getPrice(): int
+ isEnable(): boolean
+ activateEffect(Island island, PieceColor)
+ initialAction()
MotherNaturePlusTwoCard
- setMoveStrategy(MotherNatureStrategy): void
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
NoEntryTileCard
- numNoEntryTiles: int
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
- setNumNoEntryTiles(): void
EntranceToDiningRoom
- students: Map<PieceColor,Integer>
- color: PieceColor
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
TwoPointsInfluenceCard
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
ThreeToBagCard
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
NoTowersInfluenceCard
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
StudentToHallCard
- maxStudent: int
- students: Map<PieceColor,Integer>
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
NoColourInfluenceCard
- setPieceColor(PieceColor): void
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
ThreeStudentToEntryCard
- maxStudent: int
- students: Map<PieceColor,Integer>
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
StudentToIslandCard
- maxStudent: int
- students: Map<PieceColor,Integer>
+ activateEffect(Island island, PieceColor c): void
+ initialAction(): void
IslandManager
- islands: List<Island>
- influenceStrategy: InfluenceStrategy
+ IslandManager()
+ getIslands(): list<Island>
+ calculateInfluence(): void
+ unify(Island isle1, Island isle2): void
+ setInfluenceStrategy(influenceStrategy strategy): void
Island
- numOfStudents: Map<PieceColor,Integer>
- towerColor: TowerColor
- presenceOfMotherNature: boolean
- influenceStrategy: InfluenceStrategy
- numOfIslands: int
+ Island()
+ Island(Map<PieceColor,Integer> m, TowerColor t, presenceOfMotherNature mn, numOfIsland n)
+ setTowerColor(TowerColor color): void
+ getTowerColor(): TowerColor
+ addStudent(PieceColor): void
+ getStudents(): Map<PieceColor,Integer>
+ calculateInfluence(): void
+ setInfluenceStrategy(InfluenceStrategy strategy)
+ getPresenceOfMotherNature(): boolean
+ setPresenceOfMotherNature(boolean): void
+ getNumOfIslands(): int
Cloud
- numOfStudents: Map<PieceColor,Integer>
- hostableStudents: int
+ Cloud(hostableStudents int)
+ addStudent(PieceColor): void
+ removeStudent(PieceColor):void
+ getStudents(): Map<PieceColor, Integer>
+ getNumOfHostableStudents(): int
Bag
- numOfStudents: Map<PieceColor,Integer>
- bagInstance: Bag
- Bag()
+ getBagInstance(): Bag
+ drawStudent(): PieceColor
+ placeStudent (PieceColor color, num)
+ getNumOfStudents(PieceColor color): int
- isEmpty(): boolean
- removeStudent(): void
- getTotNumOfStudents: int
- randomColorGenerator(): PieceColor
GameBoard
- player: Player
- entrance: Map<PieceColor,Integer>
- diningRoom: Map<PieceColor,Integer>
- numOfTowers: int
- moveStrategy: InternalMoveSrategy
- professors: ProfessorsManager
+ addStudent(PieceColor): void
+ removeStudent(PieceColor):void
+ moveStudentToDiningRoom(PieceColor color): void
+ setNumOfTowers(int num): void
+ getEntrance(): Map<PieceColor,Integer>
+ getDiningRoom(): Map<PieceColor,Integer>
+ setInternalMoveStrategy(InternalMoveStrategy strategy): void
ProfessorsManager
- professors: Map<PieceColor, Player>
- gameBoards: Map<Player,GameBoard>
+ getOwner(PieceColor): Player
- updateProfessor(PieceColor, Player): void
+ checkProfessor(PieceColor, Player currentPlayer): void
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
MotherNature
- archipelago: List <SlotOfIsland>
- instance: MotherNature
- archipelagoIterator: Iterator<SlotOfIsland>
- position: SlotOfIsland
- movementStrategy: MovementStrategy
- MotherNature()
+ getInstance(): MotherNature
+ move(int numOfStep): void
+ getPosition(): SlotOfIsland
+ setMovementStrategy
- calculateInfluence() : void
- unifyISlots(): void
«interface»
MoveStudent
+ addStudent(PieceColor)
+ removeStudent(PieceColor)
«interface»
InternalMoveStrategy
+ moveStudentToDiningRoom()
«interface»
MovementStrategy
+ move(int numOfStep)
«interface»
InfluenceStrategy
+ calculateInfluence()
«Enum»
TowerColor
WHITE
BLACK
GREY
«Enum»
WizardFamily
WARRIOR
WITCH
KING
SHAMAN
HelperCard
- wizardFamily: WizardFamily
- nextRoundOrder: int
- maxNumOfStep: int
+ getNextRoundOrder(): int
+ getMaxNumOfStep(): int
«Enum»
PieceColor
GREEN
RED
YELLOW
PINK
BLUE
StandardMove
+ move(int numOfStep)
ReverseMove
+ move(int numOfStep)
StandardMove
+ moveStudentToDiningRoom()
ReverseMove
+ moveStudentToDiningRoom()
StandardInfluence
+ calculateInfluence(): void
NoColorInfluence
- color: PieceColor
+ calculateInfluence(): void
+ setPieceColor(PieceColor color): void
NoTowerInfluence
+ calculateInfluence(): void
PlusTwoInfluence
+ calculateInfluence(): void
NoEntryTileInfluence
+ calculateInfluence(): void
Controller
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
+ refillClouds(): void
+ playPlanningPhase(Player player)
+ playActionPhase(Player player)
«interface»
Phase
+ getNextPlayer()
PlanningPhase
- currentPlayer: Player
- match: Match
+ playHelperCard(): void
ActionPhase
- currentPlayer: Player
- match: Match
+ moveStudents(): void
+ moveMotherNature(): void
+ getStudentsFromClouds(): void
+ playCharacterCard(): void
+ chooseCloud(): void