GameBoard

bag: Bag
standing: Map<String, Integer>
modelListener: Listener
clouds: Cloud[]
professors: Map<Color, Tower>
motherNature: int
currentPlayer: int
players: PlayerBoard[]
clients: List<ClientHandler>
islands: Island[]
numRound: int
initialIslandNumber: int
playerNumber: int

+ professorsUpdateTieEffect(): void
+ getNumRound(): int
+ setListenerAndClients(Listener, List<ClientHandler>): void
+ useAssistantCard(Tower, int): boolean
+ getCurrentPlayer(): Tower
+ getSetOfSpecialCardNames(): Set<SpecialCardName>
+ setCurrentPlayer(Tower): boolean
+ getArrayOfSpecialCard(): SpecialCard[]
+ disableColorInfluence(Color): void
+ resetAllTurnFlags(): void
+ removeStudent(StudentCounter, Color, int): boolean
# transferIslandPossessionsIfSameTower(Island, Island): boolean
+ addStudent(StudentCounter, Color): boolean
+ drawFromBag(): Color
+ updateProfessorOwnershipIfTie(): void
+ notifyArchipelago(): void
+ isDiningRoomOccupied(): boolean
+ removeStudent(StudentCounter, Color): boolean
+ getCloudLimit(): int
+ getSchoolEntranceOccupation(Tower): int
+ increaseInfluence(): void
+ disableInfluence(int): boolean
+ moveMotherNature(int): boolean
+ getPlayerPosition(Tower): int
- leaderBoardUpdate(): void
- notifyClouds(): void
+ getLastAssistantCard(Tower): int
+ getUsableClouds(): Set<Integer>
+ isEndOfMatch(): EndOfMatchCondition
+ getIslandNumber(): int
+ getDiningRoomOccupation(Tower, Color): int
+ updateProfessorOwnership(): void
+ initializeBag(): void
+ computeInfluence(int): Tower
+ isBagEmpty(): boolean
+ fillClouds(): void
+ increaseRound(): void
+ paySpecialCard(int): boolean
+ getUsableAssistantCard(Tower): Map<Integer, Integer>
+ disableTowerInfluence(): void
+ getSchoolEntranceOccupation(Tower, Color): int
+ getSpecialCardCost(SpecialCardName): Integer
+ addStudent(StudentCounter, Color, int): boolean
+ getPlayerTower(String): Tower
+ notifyEndOfMatchLeaderBoard(): void
+ increaseMovementMotherNature(): void
+ getDiningRoomOccupation(Tower): int
# groupIslands(int): void

ExpertGameBoard

- noInfluenceByColor: Color
- specialCardsNum: int
- motherNatureIncreasedMove: boolean
- moreInfluenceQuantity: int
- increasedMovement: int
specialCards: SpecialCard[]
- professorsUpdateTieEffect: boolean
- towerInfluence: boolean

+ increaseMovementMotherNature(): void
+ setListenerAndClients(Listener, List<ClientHandler>): void
+ computeInfluence(int): Tower
+ getMapOfSpecialCardNames(): Map<SpecialCardName, Integer>
+ updateProfessorOwnershipIfTie(): void
+ getArrayOfSpecialCard(): SpecialCard[]
+ disableInfluence(int): boolean
+ moveMotherNature(int): boolean
+ getSetOfSpecialCardNames(): Set<SpecialCardName>
+ resetAllTurnFlags(): void
+ paySpecialCard(int): boolean
+ addStudent(StudentCounter, Color, int): boolean
+ disableColorInfluence(Color): void
+ removeStudent(StudentCounter, Color, int): boolean
+ updateProfessorOwnership(): void
+ increaseInfluence(): void
+ professorsUpdateTieEffect(): void
+ disableTowerInfluence(): void
+ getSpecialCardCost(SpecialCardName): Integer

EasyGameBoard

+ disableTowerInfluence(): void
+ getSpecialCardCost(SpecialCardName): Integer
+ getSetOfSpecialCardNames(): Set<SpecialCardName>
+ updateProfessorOwnershipIfTie(): void
+ getArrayOfSpecialCard(): SpecialCard[]
+ increaseMovementMotherNature(): void
+ disableInfluence(int): boolean
+ increaseInfluence(): void
+ professorsUpdateTieEffect(): void
+ computeInfluence(int): Tower
+ resetAllTurnFlags(): void
+ disableColorInfluence(Color): void
+ paySpecialCard(int): boolean

SpecialCard

name: SpecialCardName
modelListener: Listener
cost: int
clients: List<ClientHandler>
firstUse: boolean

+ returnTile(): boolean
+ increaseCostIfFirstUse(): void
+ removeTile(): boolean
+ getCostCoin(): int
+ countStudents(Color): int
+ setListenerAndClients(Listener, List<ClientHandler>): void
+ countStudents(): int
+ removeStudent(Color): boolean
+ getNumberOfEntryTiles(): int
+ getGuestsChangeLimit(): int
+ getName(): SpecialCardName
+ use(): void
+ addStudent(Color): boolean
+ getGuestsLimit(): int
+ isEmptyOfTiles(): boolean

SpecialCardWithStudent

- guests: HashMap<Color, Integer>
- guestsLimit: int
- guestsChangeLimit: int

+ countStudents(Color): int
- notifyGuestsChange(): void
+ getGuestsChangeLimit(): int
+ countStudents(): int
+ removeStudent(Color): boolean
+ getGuestsLimit(): int
+ addStudent(Color): boolean
+ setListenerAndClients(Listener, List<ClientHandler>): void

Herbalist

- noEntryTiles: int
- noEntryTilesLimit: int

+ setListenerAndClients(Listener, List<ClientHandler>): void
+ isEmptyOfTiles(): boolean
- notifyTilesChange(): void
+ getNumberOfEntryTiles(): int
+ removeTile(): boolean
+ returnTile(): boolean

Room

~ guestLimit: int
~ guests: HashMap<Color, Integer>

+ removeStudent(Color): boolean
+ addStudent(Color): boolean
+ countStudents(): int
+ countStudents(Color): int

DiningRoom

SchoolEntrance

+ addStudent(Color): boolean

SchoolBoard

- house: DiningRoom
- entrance: SchoolEntrance

+ addStudentDiningRoom(Color): boolean
+ getDiningRoomOccupancy(): Map<Color, Integer>
+ addStudentEntrance(Color): boolean
+ countStudentsDiningRoom(): int
+ countStudentsEntrance(): int
+ countStudentsEntrance(Color): int
+ getSchoolEntranceOccupancy(): Map<Color, Integer>
+ removeStudentEntrance(Color): boolean
+ countStudentsDiningRoom(Color): int
+ removeStudentDiningRoom(Color): boolean

PlayerBoard

- modelListener: Listener
- clients: List<ClientHandler>
- nickName: String
- lastUsed: Integer
- schoolBoard: SchoolBoard
- coin: int
- towerColor: Tower
- towers: int
- numTowersLimit: int
- deck: List<AssistantCard>

+ countStudentsEntrance(): int
+ countStudentsDiningRoom(Color): int
+ increaseCoinBudget(): void
+ setListenerAndClients(Listener, List<ClientHandler>): void
+ getTowerColor(): Tower
+ showUsableCards(): ArrayList<Integer>
+ addStudentEntrance(Color): boolean
+ getLastCard(): int
+ countStudentsDiningRoom(): int
+ useAssistantCard(int): void
+ removeStudentEntrance(Color): boolean
- notifyLastCard(): void
+ pay(int): boolean
+ addTower(int): boolean
- notifyUsablesCards(): void
+ removeStudentDiningRoom(Color): boolean
+ removeTower(int): boolean
- notifyPlayerBoard(): void
+ getCoin(): int
+ addStudentDiningRoom(Color): boolean
+ getAvailableTowers(): int
+ getAvailableCards(): Map<Integer, Integer>
+ getNickName(): String
+ countStudentsEntrance(Color): int
+ getLastCardSteps(): int

AssistantCard

~ priority: int
~ motherNatureSteps: int
- used: boolean

+ getSteps(): int
+ use(): void
+ getPriority(): int
+ isUsed(): boolean

Cloud

- counter: Map<Color, Integer>
- studentLimit: int
- hasBeenUsed: boolean

+ emptyCloud(): boolean
+ getStudents(): Map<Color, Integer>
+ isHasBeenUsed(): boolean
+ removeStudent(Color): boolean
+ countStudents(): int
+ addStudent(Color): boolean
+ isEmpty(): boolean
+ getStudentLimit(): int
+ setHasBeenUsed(boolean): void
+ isFull(): boolean
+ getLimit(): int
+ setStudents(Map<Color, Integer>): void

Island

- modelListener: Listener
- clients: List<ClientHandler>
influence: HashMap<Color, Integer>
towerColor: Tower
influenceIsEnabled: boolean
position: int
towerNumber: int

+ addStudent(Map<Color, Integer>): boolean
+ setPosition(int): void
+ studentNumber(Color): int
- notifySomethingHasChanged(): void
+ isTaken(): boolean
+ getStudentMap(): Map<Color, Integer>
+ addStudent(Color): boolean
+ setTowerNumber(int): void
+ getTowerNumber(): int
+ enableInfluence(): void
+ getPosition(): int
+ setListenerAndClients(Listener, List<ClientHandler>): void
+ disableInfluence(): void
+ getTower(): Tower
+ isInfluenceEnabled(): boolean
+ setTower(Tower): void

<<enumeration>>
SpecialCardName

+ KNIGHT:
+ BARD:
+ HERBALIST:
+ JUGGLER:
+ PRIEST:
+ PRINCESS:
+ GAMBLER:
+ POSTMAN:
+ HERALD:
+ CHEESEMAKER:
+ COOKER:
+ ARCHER:

+ convertFromStringToEnum(String): SpecialCardName?
+ values(): SpecialCardName[]
+ getSpecialCardsWithStudents(): List<SpecialCardName>
+ valueOf(String): SpecialCardName

<<enumeration>>
EndOfMatchCondition

+ NoEndOfMatch:
+ InstantEndOfMatch:
+ DelayedEndOfMatch:

+ values(): EndOfMatchCondition[]

<<enumeration>>
StudentCounter

+ DININGROOM:
+ ISLAND:
+ PLAYER:
+ CARD:
+ SCHOOLENTRANCE:
+ BAG:
+ CLOUD:

+ values(): StudentCounter[]

<<enumeration>>
Tower

+ BLACK:
+ WHITE:
+ GRAY:

+ values(): Tower[]
+ valueOf(String): Tower

<<enumeration>>
Color

+ YELLOW:
+ PINK:
+ GREEN:
+ BLUE:
+ RED:

+ toColor(String): Color?
+ getAbbreviation(Color): String
+ valueOf(String): Color

Bag

- counter: HashMap<Color, Integer>
- initStudentsPerColor: int
- studentNumber: int

+ addStudent(Color): boolean
+ countStudentByColor(Color): int
+ drawStudent(): Color
+ isEmpty(): boolean
+ initialise(): void
+ getStudentNumber(): int