<<interface>>
Effect
+ getId(): int
+ getCost(): int

CentaurEffect
+ getId(): int
+ getCost(): int

FarmerEffect
+ getId(): int
+ getCost(): int

PrincessEffect
- students: Map<HouseColor, Integer>
- takeStudent(HouseColor): void
+ hashCode(): int
+ getStudents(): Map<HouseColor, Integer>
+ effect(HouseColor, HouseColor): void
+ getId(): int
+ addStudent(HouseColor): void
+ equals(Object): boolean
+ getCost(): int

MessengerEffect
+ getCost(): int
+ getId(): int

MinstrelEffect
+ getId(): int
+ getCost(): int

JesterEffect
- students: Map<HouseColor, Integer>
+ getStudents(): Map<HouseColor, Integer>
- takeStudent(HouseColor): void
+ getId(): int
+ hashCode(): int
+ addStudent(HouseColor): void
+ getCost(): int
+ effect(HouseColor, HouseColor): void
+ equals(Object): boolean

HeraldEffect
+ getCost(): int
+ getId(): int

MonkEffect
- students: Map<HouseColor, Integer>
- takeStudent(HouseColor): void
+ hashCode(): int
+ equals(Object): boolean
+ getCost(): int
+ addStudent(HouseColor): void
+ getId(): int
+ effect(HouseColor, HouseColor): void
+ getStudents(): Map<HouseColor, Integer>

HerbalistEffect
- availableBans: int
+ getAvailableBans(): int
+ equals(Object): boolean
+ effect(Action): void
+ getCost(): int
+ getId(): int
+ putBackBan(): void
+ hashCode(): int
+ takeBan(): void

KnightEffect
+ getId(): int
+ getCost(): int

ThiefEffect
+ getCost(): int
+ getId(): int

MushroomerEffect
+ getId(): int
+ getCost(): int

SchoolBoard
- entrance: Map<HouseColor, Integer>
- diningRoom: Map<HouseColor, Integer>
- towersNumber: int
- towerType: TowerType
+ removeFromEntrance(HouseColor): void
+ getTowerType(): TowerType
+ addToDiningRoom(HouseColor): void
+ removeFromDiningRoom(HouseColor): void
+ getDiningRoom(): Map<HouseColor, Integer>
+ removeTowers(int): void
+ addToEntrance(Map<HouseColor, Integer>): void
+ getEntrance(): Map<HouseColor, Integer>
+ getStudentsNumberOf(HouseColor): int
+ getTowersNumber(): int
+ equals(Object): boolean
+ addTowers(int): void
+ hashCode(): int

Assistant
- maxDistance: int
- bonus: boolean
- id: int
+ hasBonus(): boolean
+ getId(): int
+ setBonus(): void
+ getMaxDistance(): int
+ hashCode(): int
+ equals(Object): boolean

SpecialCharacter
- id: int
- alreadyPaid: boolean
- usesNumber: int
- assignedEffect: Effect
- effectCost: int
- paidInTurn: boolean
- isActive: boolean
+ getUsesNumber(): Integer
+ getEffectCost(): int
+ equals(Object): boolean
+ isActive(): boolean
+ changedTurn(): void
+ getEffectBy(int, Map<HouseColor, Integer>, int): Effect
+ isPaidInTurn(): boolean
+ hashCode(): int
+ isAlreadyPaid(): boolean
+ getId(): int
+ activateEffect(): void
+ getEffect(): Effect
+ cleanEffect(): void
+ decreaseUsesNumber(): void

Island
- id: int
- ban: boolean
- tower: TowerType
- size: int
- students: Map<HouseColor, Integer>
+ hashCode(): int
+ getStudents(): Map<HouseColor, Integer>
+ removeBan(): void
+ setBan(): void
+ getId(): int
+ getSize(): int
+ setSize(int): void
+ getTower(): TowerType
+ addStudent(HouseColor): void
+ isBanned(): boolean
+ setTower(TowerType): void
+ equals(Object): boolean

Cloud
- id: int
- students: Map<HouseColor, Integer>
+ getStudents(): Map<HouseColor, Integer>
+ isFull(): boolean
+ flush(): Map<HouseColor, Integer>
+ hashCode(): int
+ equals(Object): boolean
+ getId(): int
+ refill(Map<HouseColor, Integer>): void

Bag
- listStudents: Deque<HouseColor>
+ push(HouseColor): void
+ boardSetUp(): List<HouseColor>
+ getStatus(): Map<HouseColor, Integer>
+ pop(): HouseColor
+ isEmpty(): boolean
+ equals(Object): boolean
+ hashCode(): int
+ randomize(): void

Player
- name: String
- wizardType: WizardType
- schoolBoard: SchoolBoard
- coins: int
- assistants: List<Assistant>
+ getCoins(): int
+ hashCode(): int
+ paySpecialCharacter(SpecialCharacter): void
+ equals(Object): boolean
+ getWizard(): WizardType
+ takeCoins(int): void
+ getSchoolBoard(): SchoolBoard
+ addCoins(): void
+ playAssistant(int): Assistant
+ getAssistants(): List<Assistant>
+ getName(): String

GameBoard
- clouds: List<Cloud>
- tieWinner: Player
- motherNatureIsland: Island
- characters: List<SpecialCharacter>
- ignoreColor: HouseColor
- professors: Map<HouseColor, Player>
- playedAssistants: Map<Player, Assistant>
- influenceBonus: Player
- bag: Bag
- islands: List<Island>
+ getInfluenceBonus(): Player
+ getAssistant(Player): Assistant
+ removeEffects(): void
+ getInfluence(Island): Map<Player, Integer>
+ addPlayedAssistant(Player, Assistant): void
- initializeClouds(int): void
+ equals(Object): boolean
- merge(Island, Island): void
+ hashCode(): int
+ setInfluenceBonus(Player): void
+ getPlayedAssistants(): Map<Player, Assistant>
+ getCharacters(): List<SpecialCharacter>
+ getInfluenceByTowers(Map<Player, Integer>, Island): void
- initializeCharacters(): void
+ getClouds(): List<Cloud>
+ setIgnoreColor(HouseColor): void
+ getMotherNatureIsland(): Island
+ getIgnoreColor(): HouseColor
+ getIslands(): List<Island>
+ moveMotherNature(Island, Assistant): void
+ setTieWinner(Player): void
+ getIslandById(int): Island
+ getTieWinner(): Player
+ getBag(): Bag
+ setTowerOnIsland(Island, TowerType): void
+ getProfessors(): Map<HouseColor, Player>
+ setProfessor(HouseColor, Player): void
+ flushAssistantsList(): void

GamePlatform
- gameBoard: GameBoard
- players: Map<String, Player>
- clockwiseOrder: List<Player>
- turnOrder: List<Player>
- currentPlayer: String
- expert: boolean
- roundWinner: String
- playersNumber: int
+ getTurnOrder(): List<Player>
- getWizardType(): WizardType
+ getPlayerByName(String): Player
- getNewOrder(String, Map<Player, Assistant>): void
+ getPlayers(): List<Player>
+ getTowerType(): TowerType
+ getGameBoard(): GameBoard
- getTowersNumber(): int
- updateTurnOrder(): void
- reorderPlayedCards(String, Map<Player, Assistant>): List<Pair<Player, Assistant>>
+ getCurrentPlayer(): Player
+ addPlayer(String): void
+ nextTurn(): void
+ isExpert(): boolean
+ nextRound(): void

«create»
assignedEffect
assistants
playedAssistants
Text
schoolBoard
characters
islands
clouds
bag
clockwiseOrder
professors
gameBoard