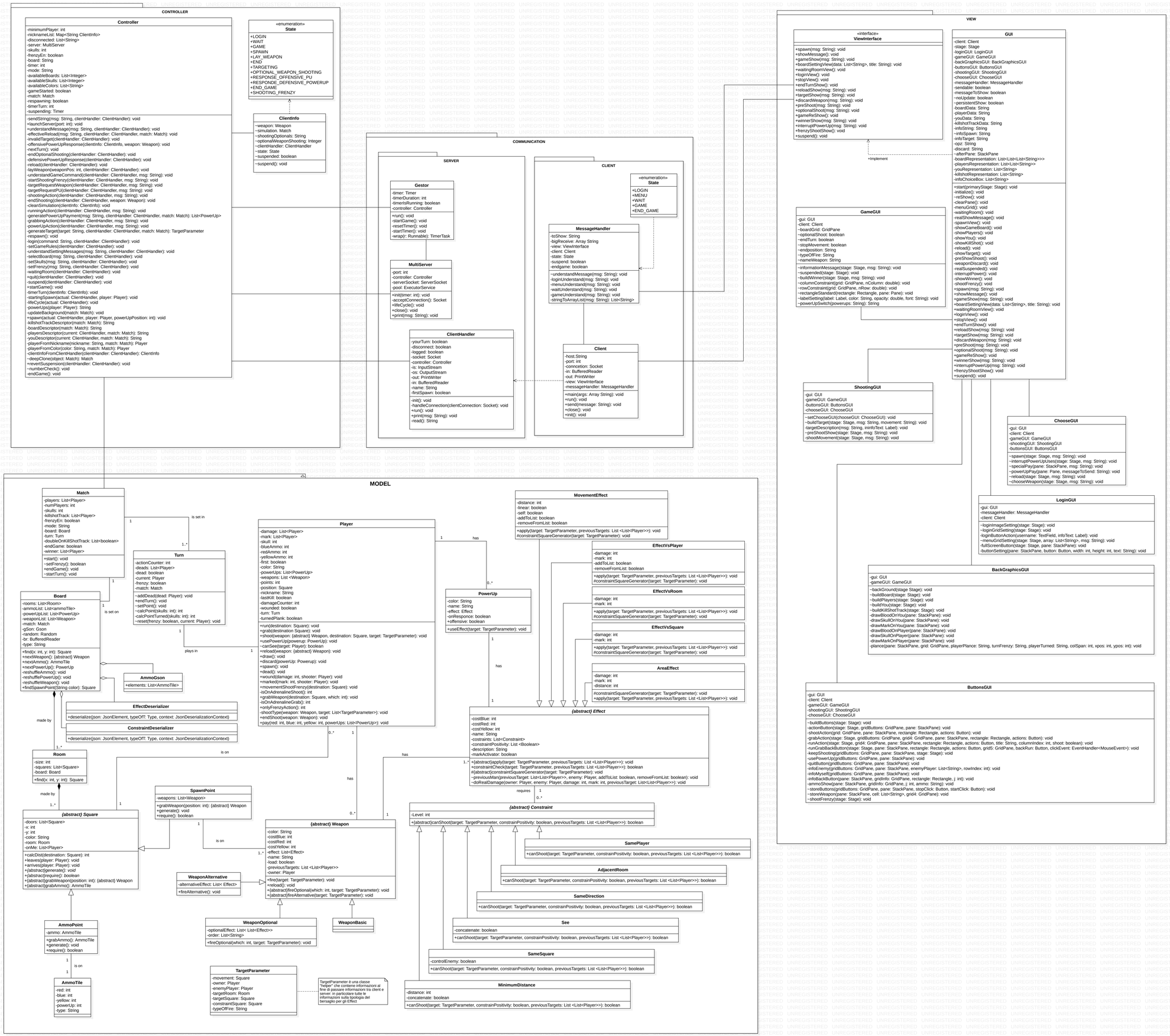
CONTROLLER
Controller
-minimumPlayer: int
-nicknameList: Map<String ClientInfo>
-disconnected: List<String>
-server: MultiServer
-skulls: int
-frenzyEn: boolean
-board: String
-timer: int
-mode: String
-availableBoards: List<Integer>
-availableSkulls: List<Integer>
-availableColors: List<String>
-gameStarted: boolean
-match: Match
-respawning: boolean
-timerTurn: int
-suspending: Timer
-sendString(msg: String, clientHandler: ClientHandler): void
+launchServer(port: int): void
+understandMessage(msg: String, clientHandler: ClientHandler): void
-effectiveReload(msg: String, clientHandler: ClientHandler, match: Match): void
-invalidTarget(clientHandler: ClientHandler): void
-offensivePowerUpResponse(clientInfo: ClientInfo, weapon: Weapon): void
-nextTurn(): void
-endOptionalShooting(clientHandler: ClientHandler): void
-defensivePowerUpResponse(clientHandler: ClientHandler): void
-reload(clientHandler: ClientHandler): void
-layWeapon(weaponPos: int, clientHandler: ClientHandler): void
-understandGameCommand(clientHandler: ClientHandler, msg: String): void
-startShootingFrenzy(clientHandler: ClientHandler, msg: String): void
-targetRequestWeapon(clientHandler: ClientHandler, msg: String): void
-targetRequestPU(clientHandler: ClientHandler, msg: String): void
-shootingAction(clientHandler: ClientHandler, msg: String): void
-endShooting(clientHandler: ClientHandler, weapon: Weapon): void
-clearSimulation(clientInfo: ClientInfo): void
-runningAction(clientHandler: ClientHandler, msg: String): void
-generatePowerUpPayment(msg: String, clientHandler: ClientHandler, match: Match): List<PowerUp>
-grabbingAction(clientHandler: ClientHandler, msg: String): void
-powerUpAction(clientHandler: ClientHandler, msg: String): void
-generateTarget(target: String, clientHandler: ClientHandler, match: Match): TargetParameter
-respawn(): void
-login(command: String, clientHandler: ClientHandler): void
-setGameRules(clientHandler: ClientHandler): void
-understandSettingMessages(msg: String, clientHandler: ClientHandler): void
-selectBoard(msg: String, clientHandler: ClientHandler): void
-setSkulls(msg: String, clientHandler: ClientHandler): void
-setFrenzy(msg: String, clientHandler: ClientHandler): void
-waitingRoom(clientHandler: ClientHandler): void
+quit(clientHandler: ClientHandler): void
-suspend(clientHandler: ClientHandler): void
+startGame(): void
-timerTurn(clientInfo: ClientInfo): void
-startingSpawn(actual: ClientHandler, player: Player): void
-liteCycle(actual: ClientHandler): void
-powerUps(player: Player): String
-updateBackground(match: Match): void
+spawn(actual: ClientHandler, player: Player, powerUpPosition: int): void
-killshotTrackDescriptor(match: Match): String
-boardDescriptor(match: Match): String
-playersDescriptor(current: ClientHandler, match: Match): String
-youDescriptor(current: ClientHandler, match: Match): String
-playerFromNickname(nickname: String, match: Match): Player
-playerFromColor(color: String, match: Match): Player
-clientInfoFromClientHandler(clientHandler: ClientHandler): ClientInfo
~deepClone(object: Match): Match
+revertSuspension(clientHandler: ClientHandler): void
~numberCheck(): void
-endGame(): void
«enumeration»
State
+LOGIN
+WAIT
+GAME
+SPAWN
+LAY_WEAPON
+END
+TARGETING
+OPTIONAL_WEAPON_SHOOTING
+RESPONSE_OFFENSIVE_PU
+RESPONDE_DEFENSIVE_POWERUP
+END_GAME
+SHOOTING_FRENZY
ClientInfo
~weapon: Weapon
~simulation. Match
~shootingOptionals: String
~optionalWeaponShooting: Integer
~clientHandler: ClientHandler
~state: State
~suspended: boolean
~suspend(): void
COMMUNICATION
SERVER
Gestor
-timer: Timer
-timerDuration: int
-timerIsRunning: boolean
-controller: Controller
+run(): void
-startGame(): void
-resetTimer(): void
-startTimer(): void
-wrap(r: Runnable): TimerTask
MultiServer
-port: int
-controller: Controller
-serverSocket: ServerSocket
-pool: ExecutorService
+init(timer: int): void
-acceptConnection(): Socket
+liteCycle(): void
+close(): void
+print(msg: String): void
ClientHandler
-yourTurn: boolean
-disconnect: boolean
-logged: boolean
-socket: Socket
-controller: Controller
-is: InputStream
-os: OutputStream
-out: PrintWriter
-in: BufferedReader
-name: String
-firstSpawn: boolean
-init(): void
-handleConnection(clientConnection: Socket): void
+run(): void
+print(msg: String): void
-read(): String
CLIENT
«enumeration»
State
+LOGIN
+MENU
+WAIT
+GAME
+END_GAME
MessageHandler
-toShow: String
-bigReceive: Array String
-view: ViewInterface
-client: Client
-state: State
-suspend: boolean
-endgame: boolean
~understandMessage(msg: String): void
-loginUnderstand(msg: String): void
-menuUnderstand(msg: String): void
-waitUnderstand(msg: String): void
-gameUnderstand(msg: String): void
-stringToArrayList(msg: String): List<String>
Client
-host.String
-port: int
-connection: Socket
-in: BufferedReader
-out: PrintWriter
-view: ViewInterface
-messageHandler: MessageHandler
+main(args: Array String): void
+run(): void
+send(message: String): void
+close(): void
+init(): void
VIEW
«interface»
ViewInterface
+spawn(msg: String): void
+showMessage(): void
+gameShow(msg: String): void
+boardSettingView(data: List<String>, title: String): void
+waitingRoomView(): void
+loginView(): void
+stopView(): void
+endTurnShow(): void
+reloadShow(msg: String): void
+targetShow(msg: String): void
+discardWeapon(msg: String): void
+preShoot(msg: String): void
+optionalShoot(msg: String): void
+gameReShow(): void
+winnerShow(msg: String): void
+interruptPowerUp(msg: String): void
+frenzyShootShow(): void
+suspend(): void
*Implement
GUI
-client: Client
-stage: Stage
-loginGUI: LoginGUI
-gameGUI: GameGUI
-backGraphicsGUI: BackGraphicsGUI
-buttonsGUI: ButtonsGUI
-shootingGUI: ShootingGUI
-chooseGUI: ChooseGUI
-messageHandler: MessageHandler
-sendable: boolean
-messageToShow: boolean
-noUpdate: boolean
~persistentShow: boolean
-boardData: String
-playerData: String
-youData: String
-killshotTrackData: String
-infoString: String
~infoSpawn: String
-infoTarget: String
-opz: String
-discard: String
~afterPane: StackPane
-boardRepresentation: List<List<List<String>>>
-playersRepresentation: List<List<String>>
-youRepresentation: List<String>
-killshotRepresentation: List<String>
-infoChoiceBox: List<String>
+start(primaryStage: Stage): void
-initialize(): void
-reShow(): void
-clearPane(): void
-menuGrid(): void
-waitingRoom(): void
-realShowMessage(): void
-spawnView(): void
-showGameBoard(): void
-showPlayers(): void
-showYou(): void
-showKillShot(): void
-reload(): void
-showTarget(): void
-preShowShoot(): void
-weaponDiscard(): void
-realSuspended(): void
-interruptPower(): void
-showWinner(): void
-shootFrenzy(): void
+spawn(msg: String): void
+showMessage(): void
+gameShow(msg: String): void
+boardSettingView(data: List<String>, title: String): void
+waitingRoomView(): void
+loginView(): void
+stopView(): void
+endTurnShow(): void
+reloadShow(msg: String): void
+targetShow(msg: String): void
+discardWeapon(msg: String): void
+preShoot(msg: String): void
+optionalShoot(msg: String): void
+gameReShow(): void
+winnerShow(msg: String): void
+interruptPowerUp(msg: String): void
+frenzyShootShow(): void
+suspend(): void
GameGUI
-gui: GUI
-client: Client
~boardGrid: GridPane
~optionalShoot: boolean
~endTurn: boolean
~stopMovement: boolean
~endposition: String
~typeOfFire: String
~nameWeapon: String
~informationMessage(stage: Stage, msg: String): void
~suspended(stage: Stage): void
~buildWinner(stage: Stage, msg: String): void
~columnConstraint(grid: GridPane, nColumn: double): void
~rowConstraint(grid: GridPane, nRow: double): void
~rectangleStandard(rectangle: Rectangle, pane: Pane): void
~labelSetting(label: Label, color: String, opacity: double, font: String): void
~powerUpSwitch(powerups: String): String
ShootingGUI
-gui: GUI
-gameGUI: GameGUI
-buttonsGUI: ButtonsGUI
-chooseGUI: ChooseGUI
~setChooseGUI(chooseGUI: ChooseGUI): void
~buildTarget(stage: Stage, msg: String, movement: String): void
-targetDescription(msg: String,ininfoText: Label): void
~preShootShow(stage: Stage, msg: String): void
-shootMovement(stage: Stage, msg: String): void
ChooseGUI
-gui: GUI
-client: Client
-gameGUI: GameGUI
-shootingGUI: ShootingGUI
-buttonsGUI: ButtonsGUI
~spawn(stage: Stage, msg: String): void
~interruptPowerUpUses(stage: Stage, msg: String): void
~specialPay(pane: StackPane, msg: String): void
~powerUpPay(pane: Pane, messageToSend: String): void
~reload(stage: Stage, msg: String): void
~chooseWeapon(stage: Stage, msg: String): void
LoginGUI
-gui: GUI
-messageHandler: MessageHandler
-client: Client
~loginImageSetting(stage: Stage): void
~loginGridSetting(stage: Stage): void
-loginButtonAction(username: TextField, infoText: Label): void
~menuGridSetting(stage: Stage, array: List<String>, msg: String): void
-fullScreenButton(stage: Stage, pane: StackPane): void
-buttonSetting(pane: StackPane, button: Button, width: int, height: int, text: String): void
BackGraphicsGUI
-gui: GUI
-gameGUI: GameGUI
~backGround(stage Stage): void
~buildBoard(stage: Stage): void
~buildPlayers(stage: Stage): void
~buildYou(stage: Stage): void
~buildKillShotTrack(stage: Stage): void
-drawBloodOnYou(pane: StackPane): void
-drawSkullOnYou(pane: StackPane): void
-drawMarkOnYou(pane: StackPane): void
-drawBloodOnPlayer(pane: StackPane): void
-drawSkullOnPlayer(pane: StackPane): void
-drawMarkOnPlayer(pane: StackPane): void
-plance(pane: StackPane, grid: GridPane, playerPlance: String, turnFrenzy: String, playerTurned: String, colSpan: int, xpos: int, ypos: int): void
ButtonsGUI
-gui: GUI
-client: Client
-gameGUI: GameGUI
-shootingGUI: ShootingGUI
-chooseGUI: ChooseGUI
~buildButtons(stage: Stage): void
-actionButton(stage: Stage, gridButtons: GridPane, pane: StackPane): void
-shootAction(grid: GridPane, pane: StackPane, rectangle: Rectangle, actions: Button): void
-grabAction(stage: Stage, gridButtons: GridPane, grid4: GridPane, pane: StackPane, rectangle: Rectangle, actions: Button): void
-runAction(stage: Stage, grid4: GridPane, pane: StackPane, rectangle: Rectangle, actions: Button, title: String, columnIndex: int, shoot: boolean): void
-runGrabBackButton(stage: Stage, pane: StackPane, rectangle: Rectangle, actions: Button, grid5: GridPane, backRun: Button, clickEvent: EventHandler<MouseEvent>): void
-keepShooting(gridButtons: GridPane, pane: StackPane, stage: Stage): void
-usePowerUp(gridButtons: GridPane, pane: StackPane): void
-quitButton(gridButtons: GridPane, pane: StackPane): void
-infoEnemy(gridButtons: GridPane, pane: StackPane, enemyPlayer: List<String>, rowIndex: int): void
-infoMyself(gridButtons: GridPane, pane: StackPane): void
-infoBackButton(pane: StackPane, gridInfo: GridPane, rectangle: Rectangle, j: int): void
-ammoShow(pane: StackPane, gridInfo: GridPane, j: int, ammo: String): void
~storeButtons(gridButtons: GridPane, pane: StackPane, stopClick: Button, startClick: Button): void
~storeWeapon(pane: StackPane, cell: List<String>, grid4: GridPane): void
~shootFrenzy(stage: Stage): void
MODEL
Match
-players: List<Player>
-numPlayers: int
-skulls: int
-killshotTrack: List<Player>
-frenzyEn: boolean
-mode: String
-board: Board
-turn: Turn
-doubleOnKillShotTrack: List<boolean>
-endGame: boolean
-winner: List<Player>
+start(): void
~setFrenzy(): boolean
+endGame(): void
~startTurn(): void
is set in
1
1..*
Turn
-actionCounter: int
-deads: List<Player>
-dead: boolean
-current: Player
-frenzy: boolean
-match: Match
~addDead(dead: Player): void
+endTurn(): void
~setPoint(): void
~calcPoint(skulls: int): int
-calcPointTurned(skulls: int): int
~reset(frenzy: boolean, current: Player): void
Board
-rooms: List<Room>
-ammoList: List<AmmoTile>
-powerUpList: List<PowerUp>
-weaponList: List<Weapon>
-match: Match
-gSon: Gson
-random: Random
-br: BufferedReader
-type: String
+find(x: int, y: int): Square
+nextWeapon(): {abstract} Weapon
+nextAmmo(): AmmoTile
+nextPowerUp(): PowerUp
-reshuffleAmmo(): void
-reshufflePowerUp(): void
-reshuffleWeapon(): void
+findSpawnPoint(String color): Square
is set on
plays in
AmmoGson
+elements: List<AmmoTile>
EffectDeserializer
+deserialize(json: JsonElement, typeOfT: Type, context: JsonDeserializationContext)
ConstraintDeserializer
+deserialize(json: JsonElement, typeOfT: Type, context: JsonDeserializationContext)
made by
Room
-size: int
-squares: List<Square>
-board: Board
+find(x: int, y: int): Square
made by
{abstract} Square
-doors: List<Square>
-x: int
-y: int
-color: String
-room: Room
-onMe: List<Player>
+calcDist(destination: Square): int
+leaves(player: Player): void
+arrives(player: Player): void
+{abstract}generate(): void
+{abstract}require(): boolean
+{abstract}grabWeapon(position: int): {abstract} Weapon
+{abstract}grabAmmo(): AmmoTile
is on
SpawnPoint
-weapons: List<Weapon>
+grabWeapon(position: int): {abstract} Weapon
+generate(): void
+require(): boolean
is on
Player
-damage: List<Player>
-mark: List<Player>
-skull: int
-blueAmmo: int
-redAmmo: int
-yellowAmmo: int
-first: boolean
-color: String
-powerUps: List<PowerUp>
-weapons: List <Weapon>
-points: int
-position: Square
-nickname: String
-lastKill: boolean
-damageCounter: int
-wounded: boolean
-turn: Turn
-turnedPlank: boolean
+run(destination: Square): void
+grab(destination Square): void
+shoot(weapon: {abstract} Weapon, destination: Square, target: TargetParameter): void
+usePowerUp(powerup: PowerUp): void
~canSee(target: Player): boolean
+reload(weapon: {abstract} Weapon): void
+draw(): void
+discard(powerUp: Powerup): void
+spawn(): void
+dead(): void
+wound(damage: int, shooter: Player): void
+marked(mark: int, shooter: Player): void
+movementShootFrenzy(destination: Square): void
-isOnAdrenalineShoot(): int
+grabWeapon(destination: Square, which: int): void
-isOnAdrenalineGrab(): int
+onlyFrenzyAction(): int
-shootType(weapon: Weapon, target: List<TargetParameter>): void
+endShoot(weapon: Weapon): void
+pay(red: int, blue: int, yellow: int, powerUps: List<PowerUp>): void
0..*
1
has
0..*
PowerUp
-color: String
-name: String
-effect: Effect
-onResponce: boolean
+offensive: boolean
+useEffect(target: TargetParameter): void
1
has
1
MovementEffect
-distance: int
-linear: boolean
-self: boolean
-addToList: boolean
-removeFromList: boolean
+apply(target: TargetParameter, previousTargets: List <List<Player>>): void
#constraintSquareGenerator(target: TargetParameter): void
EffectVsPlayer
-damage: int
-mark: int
-addToList: boolean
-removeFromList: boolean
+apply(target: TargetParameter, previousTargets: List <List<Player>>): void
#constraintSquareGenerator(target: TargetParameter): void
EffectVsRoom
-damage: int
-mark: int
+apply(target: TargetParameter, previousTargets: List <List<Player>>): void
#constraintSquareGenerator(target: TargetParameter): void
EffectVsSquare
-damage: int
-mark: int
+apply(target: TargetParameter, previousTargets: List <List<Player>>): void
#constraintSquareGenerator(target: TargetParameter): void
AreaEffect
-damage: int
-mark: int
-distance: int
#constraintSquareGenerator(target: TargetParameter): void
+apply(target: TargetParameter, previousTargets: List <List<Player>>): void
{abstract} Effect
-costBlue: int
-costRed: int
-costYellow: int
-name: String
-costraints: List<Constraint>
-constraintPositivity: List <Boolean>
-description: String
-markActivator: boolean
+{abstract}apply(target: TargetParameter, previousTargets: List <List<Player>>): void
+constraintCheck(target: TargetParameter, previousTargets: List <List<Player>>): boolean
#{abstract}constraintSquareGenerator(target: TargetParameter): void
~previousMan(previousTarget: List<List<Player>>, enemy: Player, addToList: boolean, removeFromList: boolean): void
~doRealDamage(owner: Player, enemy: Player, damage: int, mark: int, previousTarget: List<List<Player>>): void
1..*
requires
1
0..*
{abstract} Constraint
-Level: int
+{abstract}canShoot(target: TargetParameter, constrainPositivity: boolean, previousTargets: List <List<Player>>): boolean
SamePlayer
+canShoot(target: TargetParameter, constrainPositivity: boolean, previousTargets: List <List<Player>>): boolean
AdjacentRoom
+canShoot(target: TargetParameter, constrainPositivity: boolean, previousTargets: List <List<Player>>): boolean
SameDirection
+canShoot(target: TargetParameter, constrainPositivity: boolean, previousTargets: List <List<Player>>): boolean
See
-concatenate: boolean
+canShoot(target: TargetParameter, constrainPositivity: boolean, previousTargets: List <List<Player>>): boolean
SameSquare
-controlEnemy: boolean
+canShoot(target: TargetParameter, constrainPositivity: boolean, previousTargets: List <List<Player>>): boolean
MinimumDistance
-distance: int
-concatenate: boolean
+canShoot(target: TargetParameter, constrainPositivity: boolean, previousTargets: List <List<Player>>): boolean
{abstract} Weapon
-color: String
-costBlue: int
-costRed: int
-costYellow: int
-effect: List<Effect>
-name: String
-load: boolean
-previousTargets: List <List<Player>>
-owner: Player
+fire(target: TargetParameter): void
+reload(): void
+{abstract}fireOptional(which: int, target: TargetParameter): void
+{abstract}fireAlternative(target: TargetParameter): void
1..*
WeaponAlternative
-alternativeEffect: List< Effect>
+fireAlternative(): void
WeaponOptional
-optionalEffect: List< List<Effect>>
-order: List<String>
+fireOptional(which: int, target: TargetParameter): void
WeaponBasic
AmmoPoint
-ammo: AmmoTile
+grabAmmo(): AmmoTile
+generate(): void
+require(): boolean
1
is on
1
AmmoTile
-red: int
-blue: int
-yellow: int
-powerUp: int
-type: String
TargetParameter
-movement: Square
-owner: Player
-enemyPlayer: Player
-targetRoom: Room
-targetSquare: Square
-constraintSquare: Square
-typeOfFire: String
TargetParameter è una classe "helper" che contiene informazioni al fine di passare informazioni tra client e server: in particolare tutte le informazioni sulla tipologia del bersaglio per gli Effect