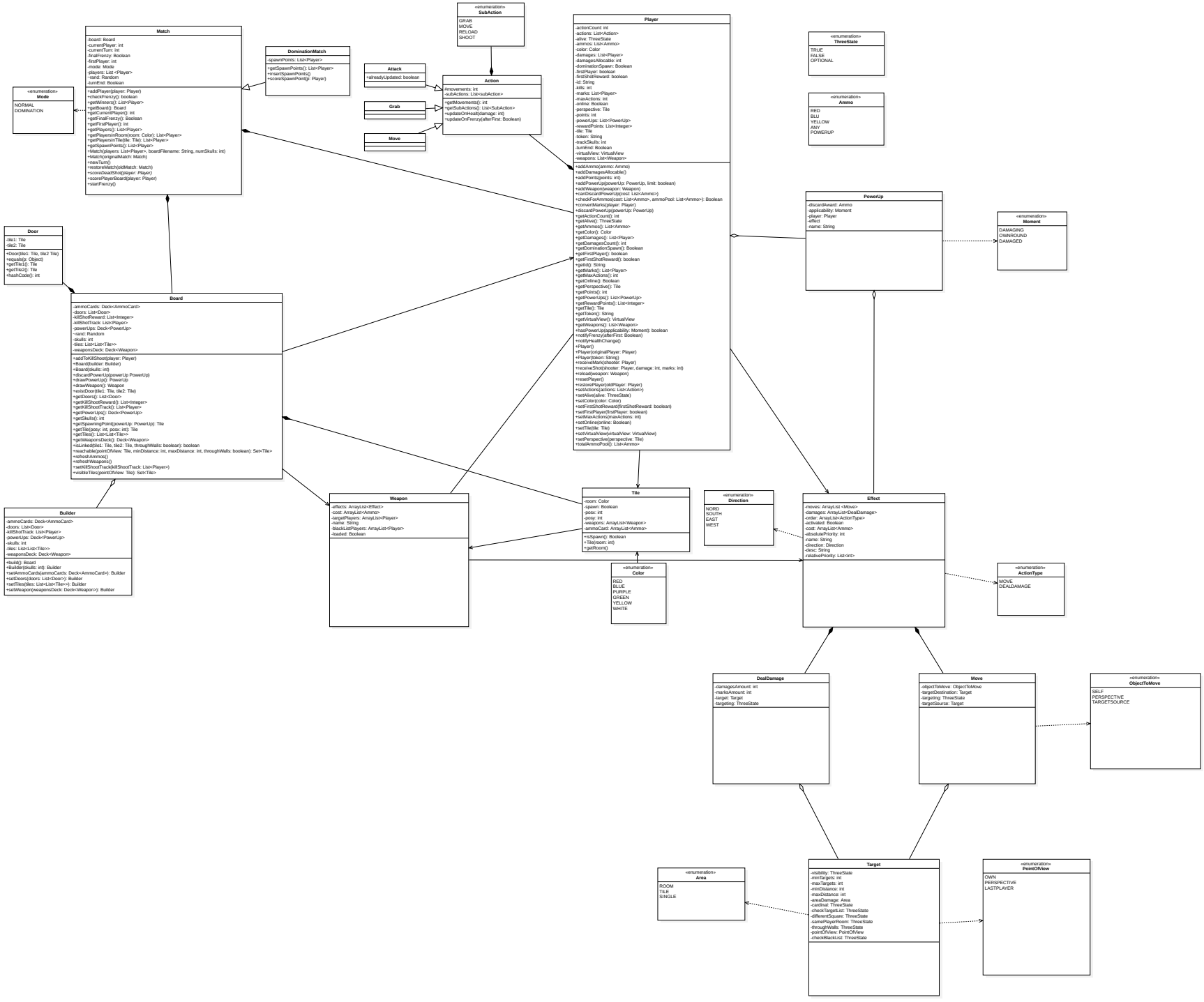
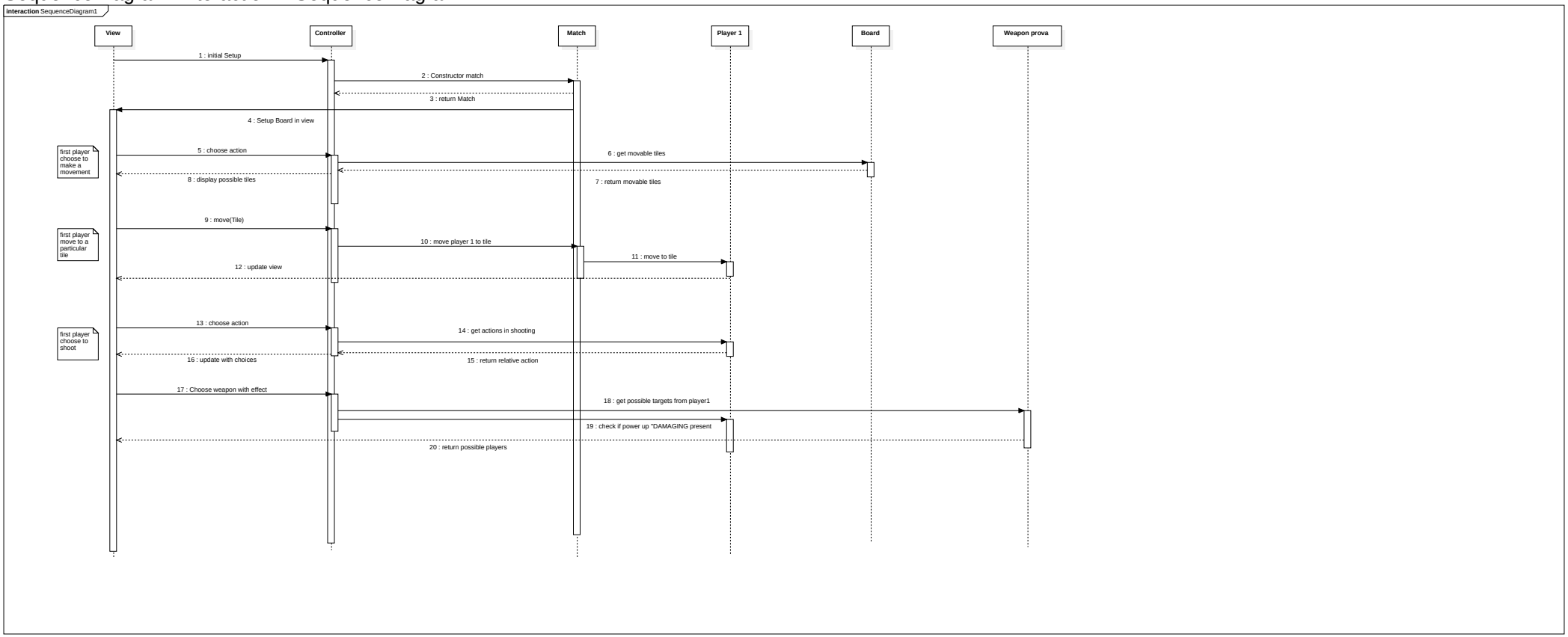
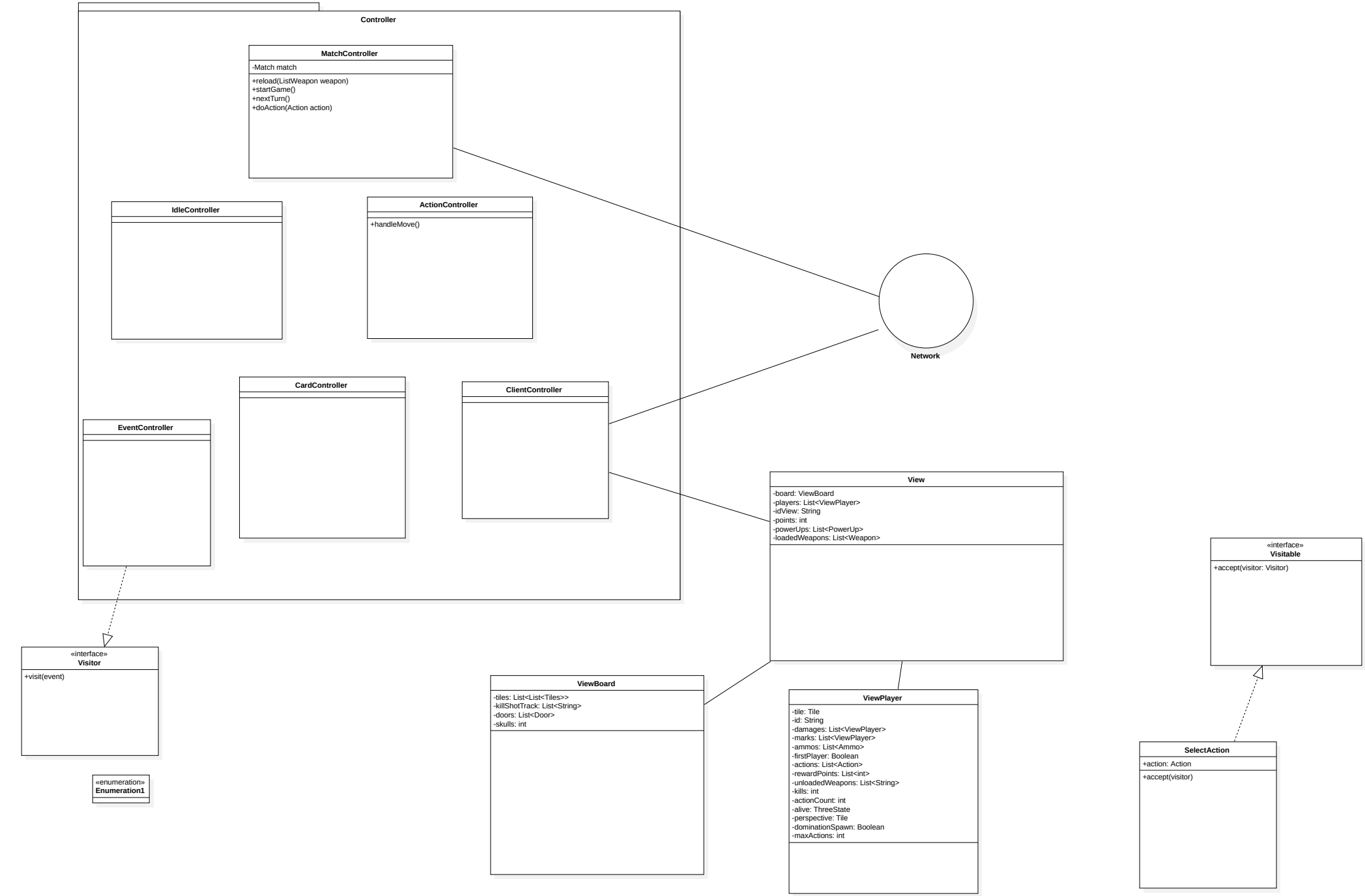
model::Model


«enumeration»
SubAction
GRAB
MOVE
RELOAD
SHOOT
Match
-board: Board
-currentPlayer: int
-currentTurn: int
-finalFrenzy: Boolean
-firstPlayer: int
-mode: Mode
-players: List <Player>
-rand: Random
-turnEnd: Boolean
+addPlayer(player: Player)
+checkFrenzy(): boolean
+getWinners(): List<Player>
+getBoard(): Board
+getCurrentPlayer(): int
+getFinalFrenzy(): Boolean
+getFirstPlayer(): int
+getPlayers(): List<Player>
+getPlayersInRoom(room: Color): List<Player>
+getPlayersInTile(tile: Tile): List<Player>
+getSpawnPoints(): List<Player>
+Match(players: List<Player>, boardFilename: String, numSkulls: int)
+Match(originalMatch: Match)
+newTurn()
+restoreMatch(oldMatch: Match)
+scoreDeadShot(player: Player)
+scorePlayerBoard(player: Player)
+startFrenzy()
DominationMatch
-spawnPoints: List<Player>
+getSpawnPoints(): List<Player>
+insertSpawnPoints()
+scoreSpawnPoint(p: Player)
Attack
+alreadyUpdated: boolean
Grab
Move
Action
#movements: int
-subActions: List<subAction>
+getMovements(): int
+getSubActions(): List<SubAction>
+updateOnHealt(damage: int)
+updateOnFrenzy(afterFirst: Boolean)
«enumeration»
Mode
NORMAL
DOMINATION
Player
-actionCount: int
-actions: List<Action>
-alive: ThreeState
-ammos: List<Ammo>
-color: Color
-damages: List<Player>
-damagesAllocable: int
-dominationSpawn: Boolean
-firstPlayer: boolean
-firstShotReward: boolean
-id: String
-kills: int
-marks: List<Player>
-maxActions: int
-online: Boolean
-perspective: Tile
-points: int
-powerUps: List<PowerUp>
-rewardPoints: List<Integer>
-tile: Tile
-token: String
-trackSkulls: int
-turnEnd: Boolean
-virtualView: VirtualView
-weapons: List<Weapon>
+addAmmo(ammo: Ammo)
+addDamagesAllocable()
+addPoints(points: int)
+addPowerUp(powerUp: PowerUp, limit: boolean)
+addWeapon(weapon: Weapon)
+canDiscardPowerUp(cost: List<Ammo>)
+checkForAmmos(cost: List<Ammo>, ammoPool: List<Ammo>): Boolean
+convertMarks(player: Player)
+discardPowerUp(powerUp: PowerUp)
+getActionCount(): int
+getAlive(): ThreeState
+getAmmos(): List<Ammo>
+getColor(): Color
+getDamages(): List<Player>
+getDamagesCount(): int
+getDominationSpawn(): Boolean
+getFirstPlayer(): boolean
+getFirstShotReward(): boolean
+getId(): String
+getMarks(): List<Player>
+getMaxActions(): int
+getOnline(): Boolean
+getPerspective(): Tile
+getPoints(): int
+getPowerUps(): List<PowerUp>
+getRewardPoints(): List<Integer>
+getTile(): Tile
+getToken(): String
+getVirtualView(): VirtualView
+getWeapons(): List<Weapon>
+hasPower(pUpgradeability: Moment): boolean
+notifyFrenzy(afterFirst: Boolean)
+notifyHealthChange()
+Player()
+Player(orginalPlayer: Player)
+Player(token: String)
+receiveMark(shooter: Player)
+receiveShot(shooter: Player, damage: int, marks: int)
+reload(weapon: Weapon)
+resetPlayer()
+restorePlayer(oldPlayer: Player)
+setActions(actions: List<Action>)
+setAlive(alive: ThreeState)
+setColor(color: Color)
+setFirstShotReward(firstShotReward: boolean)
+setFirstPlayer(firstPlayer: boolean)
+setMaxActions(maxActions: int)
+setOnline(online: Boolean)
+setTile(tile: Tile)
+setVirtualView(virtualView: VirtualView)
+setPerspective(perspective: Tile)
+totalAmmoPool(): List<Ammo>
«enumeration»
ThreeState
TRUE
FALSE
OPTIONAL
«enumeration»
Ammo
RED
BLU
YELLOW
ANY
POWERUP
PowerUp
-discardAward: Ammo
-applicability: Moment
-player: Player
-effect
-name: String
«enumeration»
Moment
DAMAGING
OWNROUND
DAMAGED
Door
-tile1: Tile
-tile2: Tile
+Door(tile1: Tile, tile2 Tile)
+equals(p: Object)
+getTile1(): Tile
+getTile2(): Tile
+hashCode(): int
Board
-ammoCards: Deck<AmmoCard>
-doors: List<Door>
-killShotReward: List<Integer>
-killShotTrack: List<Player>
-powerUps: Deck<PowerUp>
-rand: Random
-skulls: int
-tiles: List<List<Tile>>
-weaponsDeck: Deck<Weapon>
+addToKillShoot(player: Player)
+Board(builder: Builder)
+Board(skulls: int)
+discardPowerUp(powerUp PowerUp)
+drawPowerUp(): PowerUp
+drawWeapon(): Weapon
+existDoor(tile1: Tile, tile2: Tile)
+getDoors(): List<Door>
+getKillShootReward(): List<Integer>
+getKillShootTrack(): List<Player>
+getPowerUps(): Deck<PowerUp>
+getSkulls(): int
+getSpawningPoint(powerUp: PowerUp): Tile
+getTile(xray: int, posx: int): Tile
+getTiles(): List<List<Tile>>
+getWeaponsDeck(): Deck<Weapon>
+isLinked(tile1: Tile, tile2: Tile, throughWalls: boolean): boolean
+isReachable(pointOfView: Tile, minDistance: int, maxDistance: int, throughWalls: boolean): Set<Tile>
+refreshAmmos()
+refreshWeapons()
+setKillShootTrack(killShootTrack: List<Player>)
+visibleTiles(pointOfView: Tile): Set<Tile>
Builder
-ammoCards: Deck<AmmoCard>
-doors: List<Door>
-killShotTrack: List<Player>
-powerUps: Deck<PowerUp>
-skulls: int
-tiles: List<List<Tile>>
-weaponsDeck: Deck<Weapon>
+build(): Board
+Builder(skulls: int): Builder
+setAmmoCards(ammoCards: Deck<AmmoCard>): Builder
+setDoors(doors: List<Door>): Builder
+setTiles(tiles: List<List<Tile>>): Builder
+setWeapon(weaponsDeck: Deck<Weapon>): Builder
Weapon
-effects: ArrayList<Effect>
-cost: ArrayList<Ammo>
-targetPlayers: ArrayList<Player>
-name: String
-blackListPlayers: ArrayList<Player>
-loaded: Boolean
Tile
-room: Color
-spawn: Boolean
-posx: int
-posy: int
-weapons: ArrayList<Weapon>
-ammoCard: ArrayList<Ammo>
+isSpawn(): Boolean
+Tile(room: int)
+getRoom()
«enumeration»
Direction
NORD
SOUTH
EAST
WEST
«enumeration»
Color
RED
BLUE
PURPLE
GREEN
YELLOW
WHITE
Effect
-moves: ArrayList <Move>
-damages: ArrayList<DealDamage>
-order: ArrayList<ActionType>
-activated: Boolean
-cost: ArrayList<Ammo>
-absolutePriority: int
-name: String
-direction: Drection
-desc: String
-relativePriority: List<int>
«enumeration»
ActionType
MOVE
DEALDAMAGE
DealDamage
-damagesAmount: int
-marksAmount: int
-target: Target
-targeting: ThreeState
Move
-objectToMove: ObjectToMove
-targetDestination: Target
-targeting: ThreeState
-targetSource: Target
«enumeration»
ObjectToMove
SELF
PERSPECTIVE
TARGETSOURCE
«enumeration»
Area
ROOM
TILE
SINGLE
Target
-visibility: ThreeState
-minTargets: int
-maxTargets: int
-minDistance: int
-maxDistance: int
-areaDamage: Area
-cardinal: ThreeState
-checkTargetList: ThreeState
-differentSquare: ThreeState
-sameRoom: ThreeState
-throughWalls: ThreeState
-pointOfView: PointOfView
-checkBlackList: ThreeState
«enumeration»
PointOfView
OWN
PERSPECTIVE
LASTPLAYER

SequenceDiagram::Interaction1::SequenceDiagram1



controller::Controller