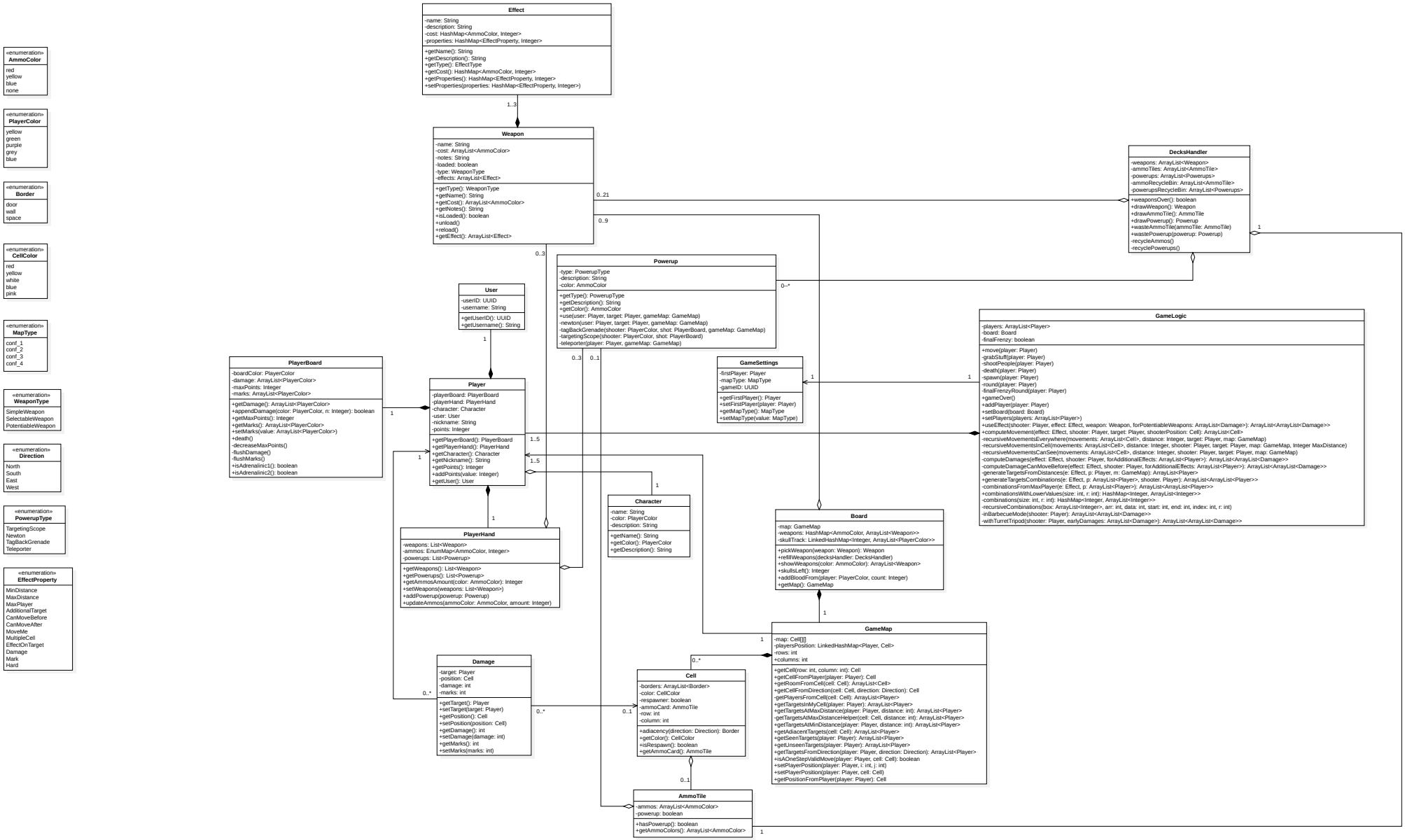
«enumeration»
AmmoColor
red
yellow
blue
none
«enumeration»
PlayerColor
yellow
green
purple
grey
blue
«enumeration»
Border
door
wall
space
«enumeration»
CellColor
red
yellow
white
blue
pink
«enumeration»
MapType
conf_1
conf_2
conf_3
conf_4
«enumeration»
WeaponType
SimpleWeapon
SelectableWeapon
PotentiableWeapon
«enumeration»
Direction
North
South
East
West
«enumeration»
PowerupType
TargetingScope
Newton
TagBackGrenade
Teleporter
«enumeration»
EffectProperty
MinDistance
MaxDistance
MaxPlayer
AdditionalTarget
CanMoveBefore
CanMoveAfter
MoveMe
MutipleCell
EffectOnTarget
Damage
Mark
Hard
Effect
-name: String
-description: String
-cost: HashMap<AmmoColor, Integer>
-properties: HashMap<EffectProperty, Integer>
+getName(): String
+getDescription(): String
+getType(): EffectType
+getCost(): HashMap<AmmoColor, Integer>
+getProperties(): HashMap<EffectProperty, Integer>
+setProperties(properties: HashMap<EffectProperty, Integer>)
1..3
Weapon
-name: String
-cost: ArrayList<AmmoColor>
-notes: String
-loaded: boolean
-type: WeaponType
-effects: ArrayList<Effect>
+getType(): WeaponType
+getName(): String
+getCost(): ArrayList<AmmoColor>
+getNotes(): String
+isLoaded(): boolean
+unload()
+reload()
+getEffect(): ArrayList<Effect>
0..21
0..9
0..3
DecksHandler
-weapons: ArrayList<Weapon>
-ammoTiles: ArrayList<AmmoTile>
-powerups: ArrayList<Powerups>
-ammoRecycleBin: ArrayList<AmmoTile>
-poweruposRecycleBin: ArrayList<Powerups>
+weaponsOver(): boolean
+drawWeapon(): Weapon
+drawAmmoTile(): AmmoTile
+drawPowerup(): Powerup
+wasteAmmoTile(ammoTile: AmmoTile)
+wastePowerup(powerup: Powerup)
-recycleAmmos()
-recyclePowerups()
1
Powerup
-type: PowerupType
-description: String
-color: AmmoColor
+getType(): PowerupType
+getDescription(): String
+getColor(): AmmoColor
+use(user: Player, target: Player, gameMap: GameMap)
-newton(user: Player, target: Player, gameMap: GameMap)
-tagBackGrenade(shooter: PlayerColor, shot: PlayerBoard, gameMap: GameMap)
-targetingScope(shooter: PlayerColor, shot: PlayerBoard)
-teleporter(player: Player, gameMap: GameMap)
0..*
User
-userID: UUID
-username: String
+getUserID(): UUID
+getUsername(): String
1
GameLogic
-players: ArrayList<Player>
-board: Board
-finalFrenzy: boolean
+move(player: Player)
-grabStuff(player: Player)
-shootPeople(player: Player)
-death(player: Player)
-spawn(player: Player)
-round(player: Player)
-finalFrenzyRound(player: Player)
+gameOver()
+addPlayer(player: Player)
+setBoard(board: Board)
+setPlayers(players: ArrayList<Player>)
+useEffect(shooter: Player, effect: Effect, weapon: Weapon, forPotentiableWeapons: ArrayList<Damage>): ArrayList<ArrayList<Damage>>
+computeMovement(effect: Effect, shooter: Player, target: Player, shooterPosition: Cell): ArrayList<Cell>
-recursiveMovementsEverywhere(movements: ArrayList<Cell>, distance: Integer, target: Player, map: GameMap)
-recursiveMovementsInCell(movements: ArrayList<Cell>, distance: Integer, shooter: Player, target: Player, map: GameMap, Integer MaxDistance)
-recursiveMovementsCanSee(movements: ArrayList<Cell>, distance: Integer, shooter: Player, target: Player, map: GameMap)
-computeDamages(effect: Effect, shooter: Player, forAdditionalEffects: ArrayList<Player>): ArrayList<ArrayList<Damage>>
-computeDamageCanMoveBefore(effect: Effect, shooter: Player, forAdditionalEffects: ArrayList<Player>): ArrayList<ArrayList<Damage>>
-generateTargetsFromDistances(e: Effect, p: Player, m: GameMap): ArrayList<Player>
+generateTargetsCombinations(e: Effect, p: ArrayList<Player>, shooter. Player): ArrayList<ArrayList<Player>>
-combinationsFromMaxPlayer(e: Effect, p: ArrayList<Player>): ArrayList<ArrayList<Player>>
+combinationsWithLowerValues(size: int, r: int): HashMap<Integer, ArrayList<Integer>>
-combinations(size: int, r: int): HashMap<Integer, ArrayList<Integer>>
-recursiveCombinations(box: ArrayList<Integer>, arr: int, data: int, start: int, end: int, index: int, r: int)
-inBarbecueMode(shooter: Player): ArrayList<ArrayList<Damage>>
-withTurretTripod(shooter: Player, earlyDamages: ArrayList<Damage>): ArrayList<ArrayList<Damage>>
1
1
GameSettings
-firstPlayer: Player
-mapType: MapType
-gameID: UUID
+getFirstPlayer(): Player
+setFirstPlayer(player: Player)
+getMapType(): MapType
+setMapType(value: MapType)
PlayerBoard
-boardColor: PlayerColor
-damage: ArrayList<PlayerColor>
-maxPoints: Integer
-marks: ArrayList<PlayerColor>
+getDamage(): ArrayList<PlayerColor>
+appendDamage(color: PlayerColor, n: Integer): boolean
+getMaxPoints(): Integer
+getMarks(): ArrayList<PlayerColor>
+setMarks(value: ArrayList<PlayerColor>)
+death()
-decreaseMaxPoints()
-flushDamage()
-flushMarks()
+isAdrenalinic1(): boolean
+isAdrenalinic2(): boolean
1
Player
-playerBoard: PlayerBoard
-playerHand: PlayerHand
-character: Character
-user: User
-nickname: String
-points: Integer
+getPlayerBoard(): PlayerBoard
+getPlayerHand(): PlayerHand
+getCharacter(): Character
+getNickname(): String
+getPoints(): Integer
+addPoints(value: Integer)
+getUser(): User
1..5
1..5
1
1
Character
-name: String
-color: PlayerColor
-description: String
+getName(): String
+getColor(): PlayerColor
+getDescription(): String
1
PlayerHand
-weapons: List<Weapon>
-ammos: EnumMap<AmmoColor, Integer>
-powerups: List<Powerup>
+getWeapons(): List<Weapon>
+getPowerups(): List<Powerup>
+getAmmosAmount(color: AmmoColor): Integer
+setWeapons(weapons: List<Weapon>)
+addPowerup(powerup: Powerup)
+updateAmmos(ammoColor: AmmoColor, amount: Integer)
Board
-map: GameMap
-weapons: HashMap<AmmoColor, ArrayList<Weapon>>
-skullTrack: LinkedHashMap<Integer, ArrayList<PlayerColor>>
+pickWeapon(weapon: Weapon): Weapon
+refillWeapons(decksHandler: DecksHandler)
+showWeapons(color: AmmoColor): ArrayList<Weapon>
+skullsLeft(): Integer
+addBloodFrom(player: PlayerColor, count: Integer)
+getMap(): GameMap
1
GameMap
-map: Cell[][]
-playersPosition: LinkedHashMap<Player, Cell>
-rows: int
+columns: int
+getCell(row: int, column: int): Cell
+getCellFromPlayer(player: Player): Cell
+getRoomFromCell(cell: Cell): ArrayList<Cell>
+getCellFromDirection(cell: Cell, direction: Direction): Cell
-getPlayersFromCell(cell: Cell): ArrayList<Player>
+getTargetsInMyCell(player: Player): ArrayList<Player>
+getTargetsAtMaxDistance(player: Player, distance: int): ArrayList<Player>
-getTargetsAtMaxDistanceHelper(cell: Cell, distance: int): ArrayList<Player>
+getTargetsAtMinDistance(player: Player, distance: int): ArrayList<Player>
+getAdiacentTargets(cell: Cell): ArrayList<Player>
+getSeenTargets(player: Player): ArrayList<Player>
+getUnseenTargets(player: Player): ArrayList<Player>
+getTargetsFromDirection(player: Player, direction: Direction): ArrayList<Player>
+isAOneStepValidMove(player: Player, cell: Cell): boolean
+setPlayerPosition(player: Player, i: int, j: int)
+setPlayerPosition(player: Player, cell: Cell)
+getPositionFromPlayer(player: Player): Cell
1
0..*
Damage
-target: Player
-position: Cell
-damage: int
-marks: int
+getTarget(): Player
+setTarget(target: Player)
+getPosition(): Cell
+setPosition(position: Cell)
+getDamage(): int
+setDamage(damage: int)
+getMarks(): int
+setMarks(marks: int)
0..*
0..*
Cell
-borders: ArrayList<Border>
-color: CellColor
-respawner: boolean
-ammoCard: AmmoTile
-row: int
-column: int
+adiacency(direction: Direction): Border
+getColor(): CellColor
+isRespawn(): boolean
+getAmmoCard(): AmmoTile
0..1
0..1
AmmoTile
-ammos: ArrayList<AmmoColor>
-powerup: boolean
+hasPowerup(): boolean
+getAmmoColors(): ArrayList<AmmoColor>
1