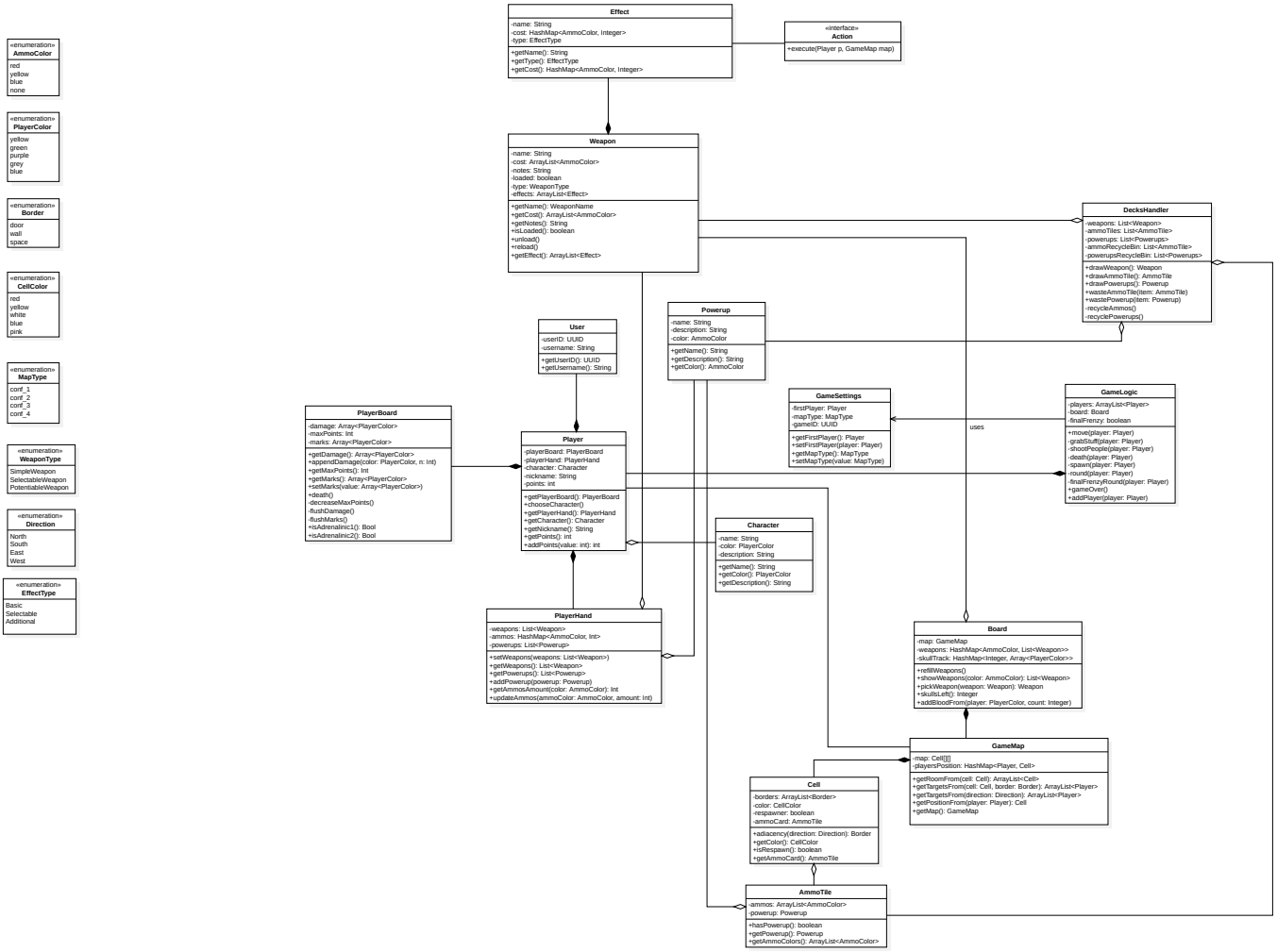
Model::Main


«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»
EffectType
Basic
Selectable
Additional
Effect
-name: String
-cost: HashMap<AmmoColor, Integer>
-type: EffectType
+getName(): String
+getType(): EffectType
+getCost(): HashMap<AmmoColor, Integer>
«interface»
Action
+execute(Player p, GameMap map)
Weapon
-name: String
-cost: ArrayList<AmmoColor>
-notes: String
-loaded: boolean
-type: WeaponType
-effects: ArrayList<Effect>
+getName(): WeaponName
+getCost(): ArrayList<AmmoColor>
+getNotes(): String
+isLoaded(): boolean
+unload()
+reload()
+getEffect(): ArrayList<Effect>
DecksHandler
-weapons: List<Weapon>
-ammoTiles: List<AmmoTile>
-powerups: List<Powerups>
-ammoRecycleBin: List<AmmoTile>
-powerupsRecycleBin: List<Powerups>
+drawWeapon(): Weapon
+drawAmmoTile(): AmmoTile
+drawPowerups(): Powerup
+wasteAmmoTile(item: AmmoTile)
+wastePowerup(item: Powerup)
-recycleAmmos()
-recyclePowerups()
User
-userID: UUID
-username: String
+getUserID(): UUID
+getUsername(): String
Powerup
-name: String
-description: String
-color: AmmoColor
+getName(): String
+getDescription(): String
+getColor(): AmmoColor
GameSettings
-firstPlayer: Player
-mapType: MapType
-gameID: UUID
+getFirstPlayer(): Player
+setFirstPlayer(player: Player)
+getMapType(): MapType
+setMapType(value: MapType)
uses
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)
PlayerBoard
-damage: Array<PlayerColor>
-maxPoints: Int
-marks: Array<PlayerColor>
+getDamage(): Array<PlayerColor>
+appendDamage(color: PlayerColor, n: Int)
+getMaxPoints(): Int
+getMarks(): Array<PlayerColor>
+setMarks(value: Array<PlayerColor>)
+death()
-decreaseMaxPoints()
-flushDamage()
-flushMarks()
+isAdrenalinic1(): Bool
+isAdrenalinic2(): Bool
Player
-playerBoard: PlayerBoard
-playerHand: PlayerHand
-character: Character
-nickname: String
-points: int
+getPlayerBoard(): PlayerBoard
+chooseCharacter()
+getPlayerHand(): PlayerHand
+getCharacter(): Character
+getNickname(): String
+getPoints(): int
+addPoints(value: int): int
Character
-name: String
-color: PlayerColor
-description: String
+getName(): String
+getColor(): PlayerColor
+getDescription(): String
PlayerHand
-weapons: List<Weapon>
-ammos: HashMap<AmmoColor, Int>
-powerups: List<Powerup>
+setWeapons(weapons: List<Weapon>)
+getWeapons(): List<Weapon>
+getPowerups(): List<Powerup>
+addPowerup(powerup: Powerup)
+getAmmosAmount(color: AmmoColor): Int
+updateAmmos(ammoColor: AmmoColor, amount: Int)
Board
-map: GameMap
-weapons: HashMap<AmmoColor, List<Weapon>>
-skullTrack: HashMap<Integer, Array<PlayerColor>>
+refillWeapons()
+showWeapons(color: AmmoColor): List<Weapon>
+pickWeapon(weapon: Weapon): Weapon
+skullsLeft(): Integer
+addBloodFrom(player: PlayerColor, count: Integer)
GameMap
-map: Cell[][]
-playersPostion: HashMap<Player, Cell>
+getRoomFrom(cell: Cell): ArrayLisi<Cell>
+getTargetsFrom(cell: Cell, border: Border): ArrayList<Player>
+getTargetsFrom(direction: Direction): ArrayList<Player>
+getPositionFrom(player: Player): Cell
+getMap(): GameMap
Cell
-borders: ArrayList<Border>
-color: CellColor
-respawner: boolean
-ammoCard: AmmoTile
+adiacency(direction: Direction): Border
+getColor(): CellColor
+isRespawn(): boolean
+getAmmoCard(): AmmoTile
AmmoTile
-ammos: ArrayLisi<AmmoColor>
-powerup: Powerup
+hasPowerup(): boolean
+getPowerup(): Powerup
+getAmmoColors(): ArrayList<AmmoColor>