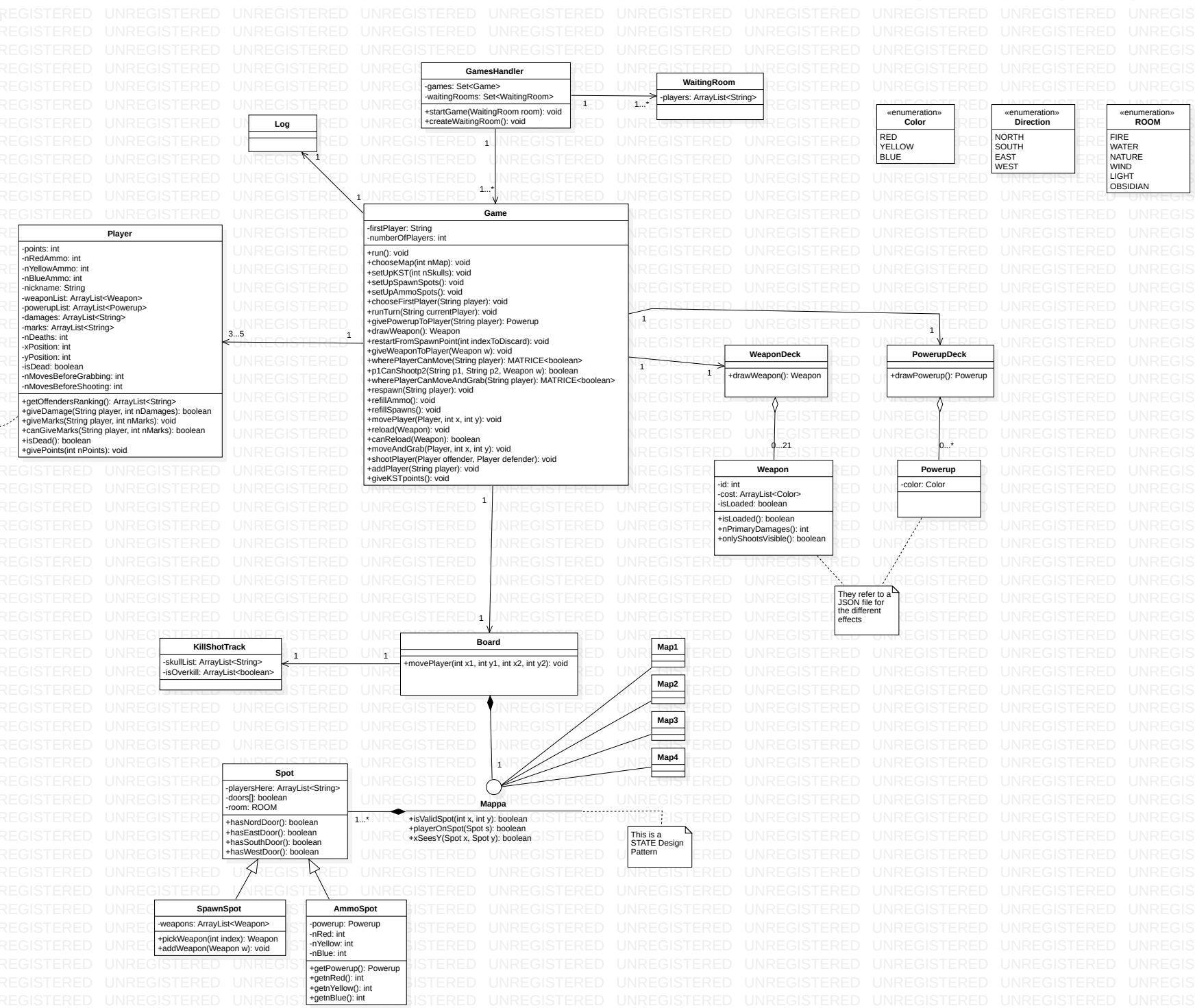
Model.Main
GamesHandler
-games: Set<Game>
-waitingRooms: Set<WaitingRoom>
+startGame(WaitingRoom room): void
+createWaitingRoom(): void
WaitingRoom
-players: ArrayList<String>
«enumeration» Color
RED
YELLOW
BLUE
«enumeration» Direction
NORTH
SOUTH
EAST
WEST
«enumeration» ROOM
FIRE
WATER
NATURE
WIND
LIGHT
OBSIDIAN
Log
Game
-firstPlayer: String
-numberOfPlayers: int
+run(): void
+chooseMap(int nMap): void
+setUpKST(int nSkulls): void
+setUpSpawnSpots(): void
+setUpAmmoSpots(): void
+chooseFirstPlayer(String player): void
+runTurn(String currentPlayer): void
+givePowerupToPlayer(String player): Powerup
+drawWeapon(): Weapon
+restartFromSpawnPoint(int indexToDiscard): void
+giveWeaponToPlayer(Weapon w): void
+wherePlayerCanMove(String player): MATRICE<boolean>
+p1CanShootp2(String p1, String p2, Weapon w): boolean
+wherePlayerCanMoveAndGrab(String player): MATRICE<boolean>
+respawn(String player): void
+refillAmmo(): void
+refillSpawns(): void
+movePlayer(Player, int x, int y): void
+reload(Weapon): void
+canReload(Weapon): boolean
+moveAndGrab(Player, int x, int y): void
+shootPlayer(Player offender, Player defender): void
+addPlayer(String player): void
+giveKSTpoints(): void
Player
-points: int
-nRedAmmo: int
-nYellowAmmo: int
-nBlueAmmo: int
-nickname: String
-weaponList: ArrayList<Weapon>
-powerupList: ArrayList<Powerup>
-damages: ArrayList<String>
-marks: ArrayList<String>
-nDeaths: int
-xPosition: int
-yPosition: int
-isDead: boolean
-nMovesBeforeGrabbing: int
-nMovesBeforeShooting: int
+getOffendersRanking(): ArrayList<String>
+giveDamage(String player, int nDamages): boolean
+giveMarks(String player, int nMarks): void
+canGiveMarks(String player, int nMarks): boolean
+isDead(): boolean
+givePoints(int nPoints): void
giveDamage & giveMarks return TRUE only if the damage kills this player
WeaponDeck
+drawWeapon(): Weapon
PowerupDeck
+drawPowerup(): Powerup
Weapon
-id: int
-cost: ArrayList<Color>
-isLoaded: boolean
+isLoaded(): boolean
+nPrimaryDamages(): int
+onlyShootsVisible(): boolean
Powerup
-color: Color
They refer to a JSON file for the different effects
KillShotTrack
-skullList: ArrayList<String>
-isOverkill: ArrayList<boolean>
Board
+movePlayer(int x1, int y1, int x2, int y2): void
Map1
Map2
Map3
Map4
Mappa
+isValidSpot(int x, int y): boolean
+playerOnSpot(Spot s): boolean
+xSeesY(Spot x, Spot y): boolean
This is a STATE Design Pattern
Spot
-playersHere: ArrayList<String>
-doors[]: boolean
-room: ROOM
+hasNordDoor(): boolean
+hasEastDoor(): boolean
+hasSouthDoor(): boolean
+hasWestDoor(): boolean
SpawnSpot
-weapons: ArrayList<Weapon>
+pickWeapon(int index): Weapon
+addWeapon(Weapon w): void
AmmoSpot
-powerup: Powerup
-nRed: int
-nYellow: int
-nBlue: int
+getPowerup(): Powerup
+getnRed(): int
+getnYellow(): int
+getnBlue(): int
1
1...*
3...5
0...21
0...*