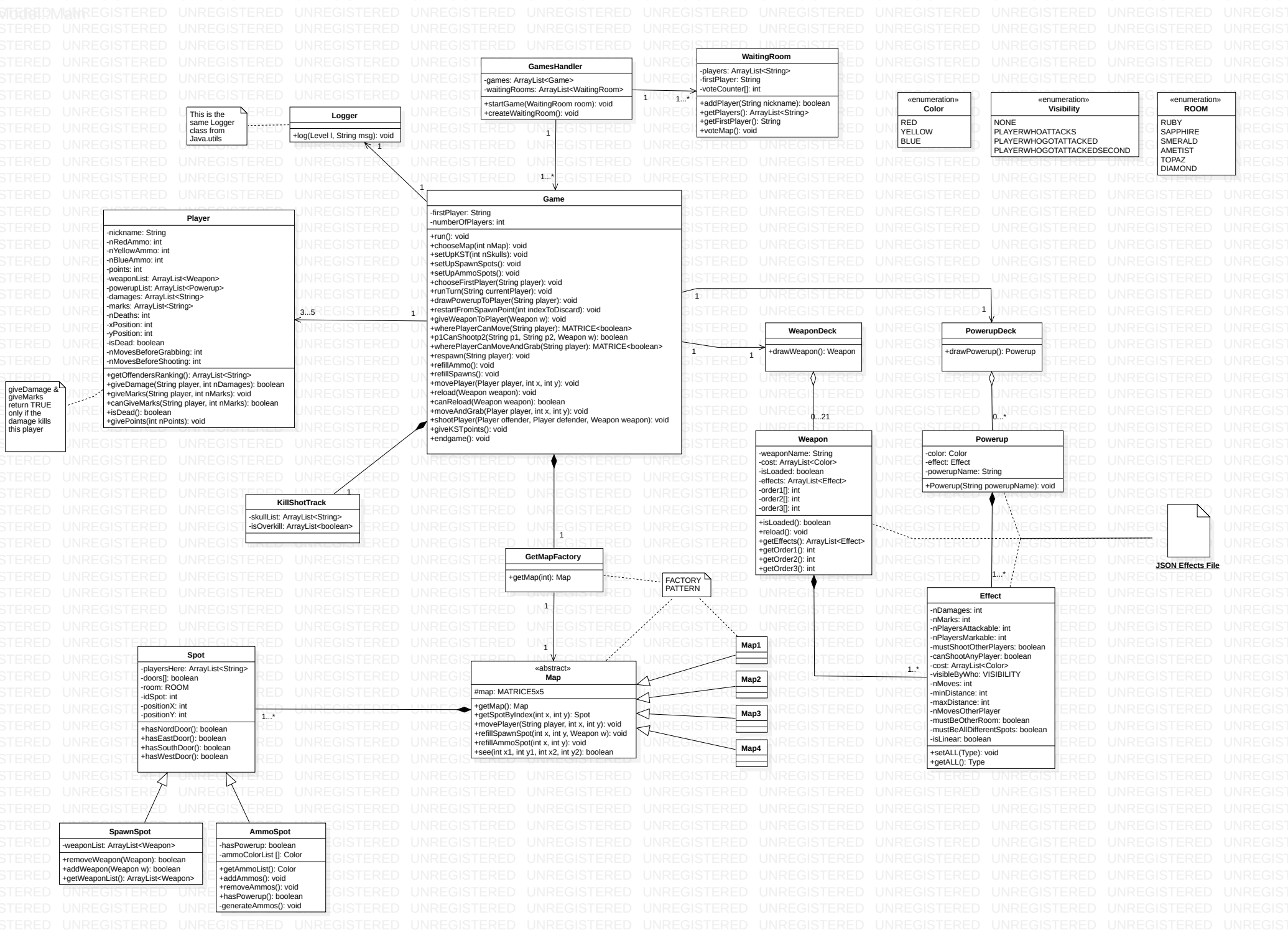
Model.main
GamesHandler
-games: ArrayList<Game>
-waitingRooms: ArrayList<WaitingRoom>
+startGame(WaitingRoom room): void
+createWaitingRoom(): void
WaitingRoom
-players: ArrayList<String>
-firstPlayer: String
-voteCounter[]: int
+addPlayer(String nickname): boolean
+getPlayers(): ArrayList<String>
+getFirstPlayer(): String
+voteMap(): void
«enumeration»
Color
RED
YELLOW
BLUE
«enumeration»
Visibility
NONE
PLAYERWHOATTACKS
PLAYERWHOGOTATTACKED
PLAYERWHOGOTATTACKEDSECOND
«enumeration»
ROOM
RUBY
SAPPHIRE
SMERALD
AMETIST
TOPAZ
DIAMOND
This is the same Logger class from Java.utils
Logger
+log(Level l, String msg): void
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
+drawPowerupToPlayer(String player): void
+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 player, int x, int y): void
+reload(Weapon weapon): void
+canReload(Weapon weapon): boolean
+moveAndGrab(Player player, int x, int y): void
+shootPlayer(Player offender, Player defender, Weapon weapon): void
+giveKSTpoints(): void
+endgame(): void
Player
-nickname: String
-nRedAmmo: int
-nYellowAmmo: int
-nBlueAmmo: int
-points: int
-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
-weaponName: String
-cost: ArrayList<Color>
-isLoaded: boolean
-effects: ArrayList<Effect>
-order1[]: int
-order2[]: int
-order3[]: int
+isLoaded(): boolean
+reload(): void
+getEffects(): ArrayList<Effect>
+getOrder1(): int
+getOrder2(): int
+getOrder3(): int
Powerup
-color: Color
-effect: Effect
-powerupName: String
+Powerup(String powerupName): void
JSON Effects File
KillShotTrack
-skullList: ArrayList<String>
-isOverkill: ArrayList<boolean>
GetMapFactory
+getMap(int): Map
FACTORY PATTERN
Effect
-nDamages: int
-nMarks: int
-nPlayersAttackable: int
-nPlayersMarkable: int
-mustShootOtherPlayers: boolean
-canShootAnyPlayer: boolean
-cost: ArrayList<Color>
-visibleByWho: VISIBILITY
-nMoves: int
-minDistance: int
-maxDistance: int
-nMovesOtherPlayer
-mustBeOtherRoom: boolean
-mustBeAllDifferentSpots: boolean
-isLinear: boolean
+setALL(Type): void
+getALL(): Type
Spot
-playersHere: ArrayList<String>
-doors[]: boolean
-room: ROOM
-idSpot: int
-positionX: int
-positionY: int
+hasNordDoor(): boolean
+hasEastDoor(): boolean
+hasSouthDoor(): boolean
+hasWestDoor(): boolean
«abstract»
Map
#map: MATRICE5x5
+getMap(): Map
+getSpotByIndex(int x, int y): Spot
+movePlayer(String player, int x, int y): void
+refillSpawnSpot(int x, int y, Weapon w): void
+refillAmmoSpot(int x, int y): void
+see(int x1, int y1, int x2, int y2): boolean
Map1
Map2
Map3
Map4
SpawnSpot
-weaponList: ArrayList<Weapon>
+removeWeapon(Weapon): boolean
+addWeapon(Weapon w): boolean
+getWeaponList(): ArrayList<Weapon>
AmmoSpot
-hasPowerup: boolean
-ammoColorList []: Color
+getAmmoList(): Color
+addAmmos(): void
+removeAmmos(): void
+hasPowerup(): boolean
-generateAmmos(): void
1
1...*
3...5
0...21
0...*
1..*