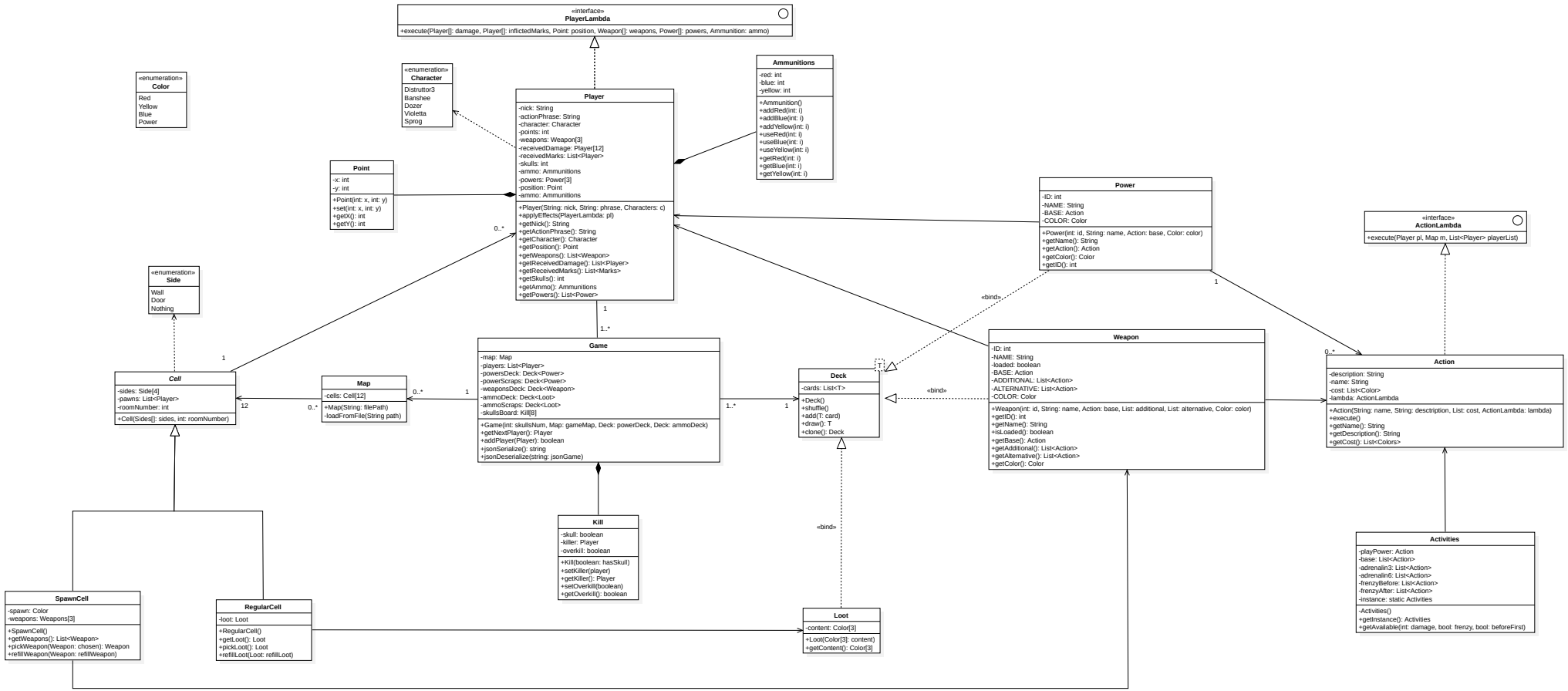
«interface»
PlayerLambda
+execute(Player[]: damage, Player[]: inflictedMarks, Point: position, Weapon[]: weapons, Power[]: powers, Ammunition: ammo)
«enumeration»
Color
Red
Yellow
Blue
Power
«enumeration»
Character
Distruttor3
Banshee
Dozer
Violetta
Sprog
Ammunitions
-red: int
-blue: int
-yellow: int
+Ammunition()
+addRed(int: i)
+addBlue(int: i)
+addYellow(int: i)
+useRed(int: i)
+useBlue(int: i)
+useYellow(int: i)
+getRed(int: i)
+getBlue(int: i)
+getYellow(int: i)
Player
-nick: String
-actionPhrase: String
-character: Character
-points: int
-weapons: Weapon[3]
-receivedDamage: Player[12]
-receivedMarks: List<Player>
-skulls: int
-ammo: Ammunitions
-powers: Power[3]
-position: Point
-ammo: Ammunitions
+Player(String: nick, String: phrase, Characters: c)
+applyEffects(PlayerLambda: pl)
+getNick(): String
+getActionPhrase(): String
+getCharacter(): Character
+getPosition(): Point
+getWeapons(): List<Weapon>
+getReceivedDamage(): List<Player>
+getReceivedMarks(): List<Marks>
+getSkulls(): int
+getAmmo(): Ammunitions
+getPowers(): List<Power>
Point
-x: int
-y: int
+Point(int: x, int: y)
+set(int: x, int: y)
+getX(): int
+getY(): int
Power
-ID: int
-NAME: String
-BASE: Action
-COLOR: Color
+Power(int: id, String: name, Action: base, Color: color)
+getName(): String
+getAction(): Action
+getColor(): Color
+getID(): int
«interface»
ActionLambda
+execute(Player pl, Map m, List<Player> playerList)
«enumeration»
Side
Wall
Door
Nothing
«bind»
Cell
-sides: Side[4]
-pawns: List<Player>
-roomNumber: int
+Cell(Sides[]: sides, int: roomNumber)
Map
-cells: Cell[12]
+Map(String: filePath)
-loadFromFile(String path)
Game
-map: Map
-players: List<Player>
-powersDeck: Deck<Power>
-powerScraps: Deck<Power>
-weaponsDeck: Deck<Weapon>
-ammoDeck: Deck<Loot>
-ammoScraps: Deck<Loot>
-skullsBoard: Kill[8]
+Game(int: skullsNum, Map: gameMap, Deck: powerDeck, Deck: ammoDeck)
+getNextPlayer(): Player
+addPlayer(Player): boolean
+jsonSerialize(): string
+jsonDeserialize(string: jsonGame)
Deck
T
-cards: List<T>
+Deck()
+shuffle()
+add(T: card)
+draw(): T
+clone(): Deck
«bind»
Weapon
-ID: int
-NAME: String
-loaded: boolean
-BASE: Action
-ADDITIONAL: List<Action>
-ALTERNATIVE: List<Action>
-COLOR: Color
+Weapon(int: id, String: name, Action: base, List: additional, List: alternative, Color: color)
+getID(): int
+getName(): String
+isLoaded(): boolean
+getBase(): Action
+getAdditional(): List<Action>
+getAlternative(): List<Action>
+getColor(): Color
Action
-description: String
-name: String
-cost: List<Color>
-lambda: ActionLambda
+Action(String: name, String: descriptiion, List: cost, ActionLambda: lambda)
+execute()
+getName(): String
+getDescription(): String
+getCost(): List<Colors>
Kill
-skull: boolean
-killer: Player
-overkill: boolean
+Kill(boolean: hasSkull)
+setKiller(player)
+getKiller(): Player
+setOverkill(boolean)
+getOverkill(): boolean
«bind»
Activities
-playPower: Action
-base: List<Action>
-adrenalin3: List<Action>
-adrenalin6: List<Action>
-frenzyBefore: List<Action>
-frenzyAfter: List<Action>
-instance: static Activities
-Activities()
+getInstance(): Activities
+getAvailable(int: damage, bool: frenzy, bool: beforeFirst)
SpawnCell
-spawn: Color
-weapons: Weapons[3]
+SpawnCell()
+getWeapons(): List<Weapon>
+pickWeapon(Weapon: chosen): Weapon
+refillWeapon(Weapon: refillWeapon)
RegularCell
-loot: Loot
+RegularCell()
+getLoot(): Loot
+pickLoot(): Loot
+refillLoot(Loot: refillLoot)
Loot
-content: Color[3]
+Loot(Color[3]: content)
+getContent(): Color[3]
0..*
1
1..*
1
12
0..*
0..*
1
1..*
1
1
0..*