MODEL Main

Model

«enumeration» TileColor
green
red
yellow
blue
white
purple

Tile *(italic)*
-color: TileColor
-doors: boolean[4]
+grabObjects(player: Player)
+getColor(): TileColor
+getDoors(): boolean[4]

Board
-scheme: Tile[ ][ ]
-width: int
-height: int
+getScheme(): Tile
+getTileFromPosition(pos: Position): Tile
+fromJSON(json: String): Board
+getWidth(): int
+getHeight(): int

SpawnTile
-weapons: Weapon[3]
+addWeapon(value: Weapon)

NormalTile
-ammoCard: AmmoCard
+setAmmoCard(value: AmmoCard)

AmmoCard
-ammoGain: AmmoValue
-drawPowerUp: boolean
+getDrawPowerUp(): boolean
+getAmmoGain(): AmmoValue
+returnDeckFromJson(json: String)

Cloneable

Weapon *(italic)*
-name: String
-loaded: boolean
-grabCost: AmmoValue
-reloadCost: AmmoValue
+shoot(shooter: Player)
+getName(): String
+getGrabCost(): AmmoValue
+getReloadCost(): AmmoValue
+getTargets(shooter: Player, board: Board): ArrayList<Player>[0..5]
+isLoaded(): boolean

AmmoValue
-red: int
-yellow: int
-blue: int
+MAX_AMMO: int
+add(value: AmmoValue)
+subtract(value: AmmoValue)
+clampValue(value: int)
+getRed(): int
+getYellow(): int
+getBlue(): int
+isValueValid(value: int): boolean

«Exception» NotEnoughAmmoException

Player
-ammo: AmmoValue
-powerUps: PowerUpCard[4]
-weapons: Weapon[3]
-color: PlayerColor
-deathsNum: int
-damageTaken: PlayerColor[12]
-marks: EnumMap<PlayerColor,int>
-score: int
-pos: Position
-name: String
-isDead: boolean
+MAX_MARKS: int
+addScore(value: int): void
+getAmmo(): AmmoValue
+setAmmo(value: AmmoValue)
+getColor(): PlayerColor
+getDeathsNum(): int
+incrementDeaths(): void
+getWeapons(): Weapon[3]
+addWeapon(value: Weapon)
+getPowerUps(): PowerUpCard
+addPowerUp(value: PowerUpCard, isRespawn: boolean)
+getScore(): int
+getPos(): Position
+move(value: Position)
+getName(): String
+initializeMarkMap(): void
+sufferedDamage(attackingPlayer: PlayerColor, damage: int)
+getDamageTaken(): PlayerColor[12]
+getMarks(): EnumMap<PlayerColor,int>
+isDead(): boolean
+addPowerUp(card: PowerUpCard): void
+discard(card: PowerUpCard): void

MachineGun

One class for
each weapon in
the game

Damage
-damage: int
-marksNum: int
+getDamage(): int
+getMarksNum(): int

«Exception» FullHandException

Game
-board: Board
-players: ArrayList<Player>
-powerUpDeck: Deck<PowerUpCard>
-weapons: Deck<Weapon>
-ammoCardDeck: Deck<AmmoCard>
-turnNumber: int
-skullNum: int
-deaths: ArrayList<PlayerColor>
-finalFrenzy: boolean
-finalFrenzyStart: Integer
-activePlayer: PlayerColor
+KILLS_VALUE: int[]
+getVisiblePlayers(player: Player): ArrayList<Player>
+startNextTurn()
+isGameOver(): boolean
+getTurnNumber(): int
+getAmmoCardDeck(): Deck<AmmoCard>
+getWeapons(): Deck<Weapon>
+getPlayers(): ArrayList<Player>
+getPowerUpDeck(): Deck<PowerUpCard>
+getBoard(): Board
+getSkullNum(): int
+getPlayerFromColor(color: PlayerColor): Player
+addDeath(killer: PlayerColor)
+isFinalFrenzy(): boolean
+getFinalFrenzyTurnStart(): Integer
+setFinalFrenzyStatus()
+distributeKillScore()

PowerUpCard
-name: String
-ammoValue: AmmoValue
-behaviour: PowerUpBehaviour
+getColor(): TileColor
+getName(): String
+activate(player: Player)

«enumeration» PlayerColor
yellow
green
purple
grey
blue

Position
-x: int
-y: int
+getY(): int
+getX(): int

«interface» PowerUpBehaviour
+activate(player: Player)

Deck<T>
-deck: ArrayList<T>
-cards: ArrayList<T>
+drawCard(): T
+shuffle()
+getDeck(): ArrayList<T>
+getCards(): ArrayList<T>

TeleportBehaviour

VenomGranadeBehaviour

NewtonBehaviour

TargetScopeBehaviour

Strategy pattern

CompositeAction
-action: ArrayList<Action>
+perform(game: Game)
+getAction(): ArrayList<Action>
+isValid(game: Game): boolean

Action *(italic)*
-player: PlayerColor
+perform(game: Game)
+isValid(game: Game): boolean
+getPlayerColor(): PlayerColor

MoveAction
-pos: Position
+getPos(): Position

GrabAction

ReloadAction
-weaponIndex: int
+getWeaponIndex(): int

PowerUpAction

ShootAction
«?»=weaponIndex or effect

Work in progress, need to add many messages from model to view and link
them to correct classes

PlayerMovedMessage
-player: Player
+getPlayer(): Player

PlayerDamagedMessage

ObjectGrabbedMessage

View

CLIView
-scanner: Scanner
+parseCommand(command: String)

View *(italic)*
-playerViews: PlayerView[3..5]
-boardView: BoardView
-infoView: InfoView
+showMessage(message: String)
+reportError(error: String)
+updateBoard()
+updatePlayers()
+interact()
+getActivePlayer(): PlayerColor

GraphicView
«Finestre»-panels: Panel[1..*]
+showPanel(panel: int)

VirtualView
+notifyController()

PlayerView
-player: Player
+getPlayer(): Player

BoardView
-board: Board
+getBoard(): Board

InfoView
-activePlayer: PlayerColor
-turnNumber: int
-deaths: ArrayList<PlayerColor>
+getTurnNumber(): int
+getActivePlayer(): PlayerColor
+getDeaths(): ArrayList<PlayerColor>

WeaponView
-weapon: Weapon
+getWeapon(): Weapon

Notified from model when
player take damage

VenomGranadeView

ActionMessage
-actionList: CompositeAction
-view: View
+getActions(): CompositeAction
+getView(): View

Message from
VirtualView to Controller

Util

ResourceHandler
+PATH_TO_RESOURCE_FOLDER: String
-resources: HashMap<String, Resource>
+registerResource(loader: Function<String, Resource>, path: String)
+registerResource(loader: Function<String, Resource>, path: String, customName: String)
+get(resourceKey: String): Object

Resource *(italic)*
+path: String
+generateName(): String
+get(): Object

«Exception» NonExistentResourceException

Observable

Observer

JsonResource
+contents: String
+loadFromPath(path: String): Resource

Controller

Controller
-model: Game
+performPlayerAction(action: Action)
+requestPlayerAction()

Actually not used but
stashed just in case

PerformTurnMessage
-board: Board
-players: Player[3..5]
-activePlayer: PlayerColor
+getPlayers(): Player
+getActivePlayer(): PlayerColor
+getBoard(): Board

Network

ServerStrings
+serverURL: String
+port: String

Server
-socket: ServerSocket
-input: DataInputStream
-out: DataOutputStream
-games: ArrayList<GameThread>[0..*]

ClientSocket
+socket: Socket
+out: DataOutputStream
+input: DataInputStream

ClientConnection

GameThread
-model: Game
-controller: Controller
-virtualViews: View[0..*]

Work in progress