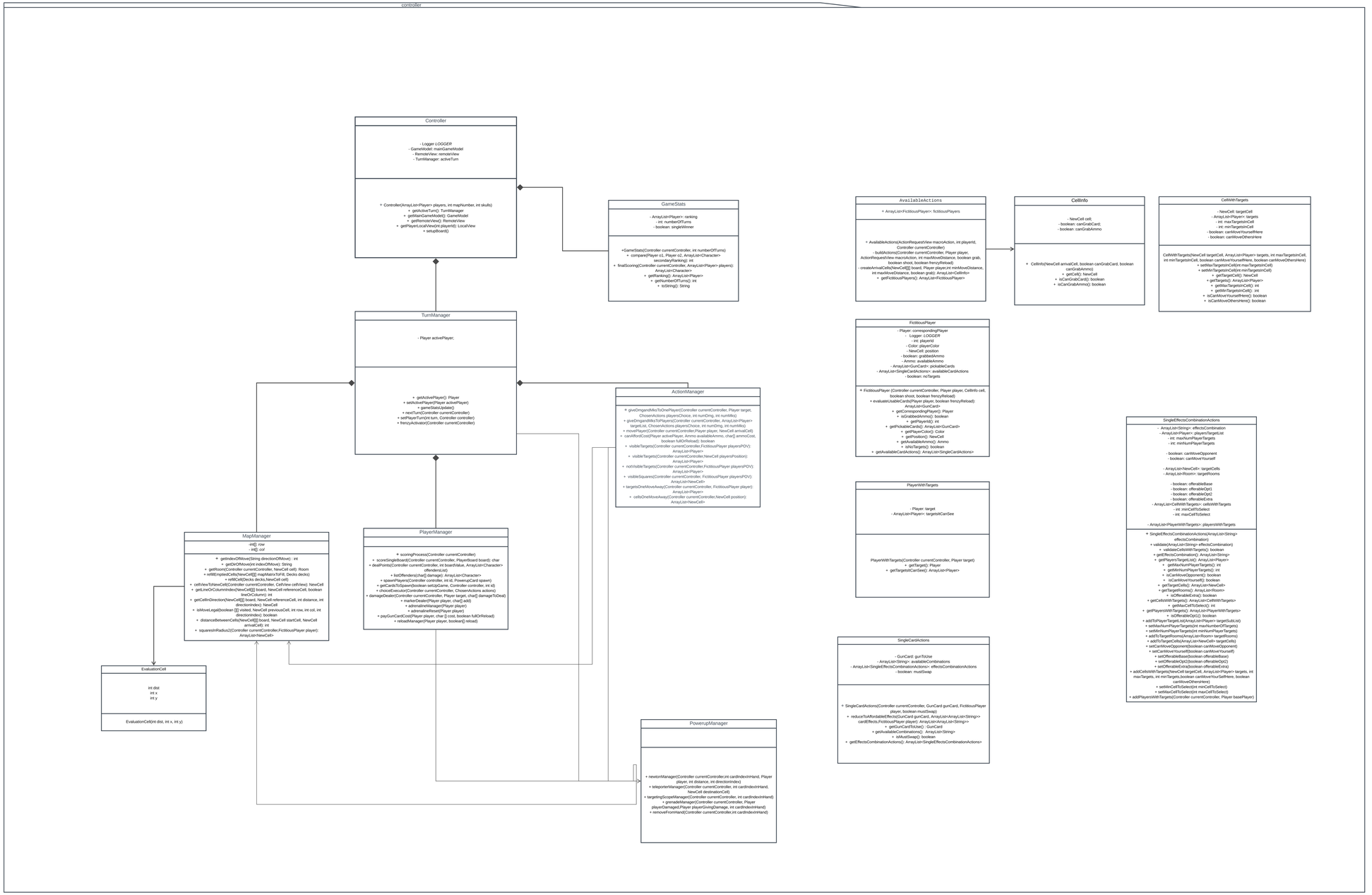
controller
Controller
- Logger LOGGER
- GameModel: mainGameModel
- RemoteView: remoteView
- TurnManager: activeTurn
+ Controller(ArrayList<Player> players, int mapNumber, int skulls)
+ getActiveTurn(): TurnManager
+ getMainGameMode(): GameModel
+ getRemoteView(): RemoteView
+ getPlayerLocalView(int playerId): LocalView
+ setupBoard()
GameStats
- ArrayList<Player>: ranking
- int: numberOfTurns
- boolean: singleWinner
+GameStats(Controller currentController, int numberOfTurns)
+ compare(Player o1, Player o2, ArrayList<Character> secondaryRanking): int
+ finalScoring(Controller currentController, ArrayList<Player> players): ArrayList<Character>
+ getRanking(): ArrayList<Player>
+ getNumberOfTurns(): int
+ toString(): String
TurnManager
- Player activePlayer;
+ getActivePlayer(): Player
+ setActivePlayer(Player activePlayer)
+ gameStateUpdate()
+ nextTurn(Controller currentController)
+ setPlayerTurn(int turn, Controller controller)
+ frenzyActivator(Controller currentController)
ActionManager
+ giveDmgandMksToOnePlayer(Controller currentController, Player target, ChosenActions playersChoice, int numDmg, int numMks)
+ giveDmgandMksToPlayers(Controller currentController, ArrayList<Player> targetList, ChosenActions playersChoice, int numDmg, int numMks)
+ movePlayer(Controller currentController,Player player, NewCell arrivalCell)
+ canAffordCost(Player activePlayer, Ammo availableAmmo, char[] ammoCost, boolean fullOrReload): boolean
+ visibleTargets(Controller currentController,FictitiousPlayer playersPOV): ArrayList<Player>
+ visibleTargets(Controller currentController,NewCell playersPosition): ArrayList<Player>
+ notVisibleTargets(Controller currentController,FictitiousPlayer playersPOV): ArrayList<Player>
+ visibleSquares(Controller currentController, FictitiousPlayer playersPOV): ArrayList<NewCell>
+ targetsOneMoveAway(Controller currentController, FictitiousPlayer player): ArrayList<Player>
+ cellsOneMoveAway(Controller currentController,NewCell position): ArrayList<NewCell>
MapManager
-int[] row
- int[] col
+ getIndexOfMove(String directionOfMove) : int
+ getDirOfMove(int indexOfMove): String
+ getRoom(Controller currentController, NewCell cell): Room
+ refillEmptiedCells(NewCell[][] mapMatrixToFill, Decks decks)
+ refillCell(Decks decks,NewCell cell)
+ cellViewToNewCell(Controller currentController, CellView cellView): NewCell
+ getLineOrColumnIndex(NewCell[][] board, NewCell referenceCell, boolean lineOrColumn): int
+ getCellInDirection(NewCell[][] board, NewCell referenceCell, int distance, int directionIndex): NewCell
+ isMoveLegal(boolean [][] visited, NewCell previousCell, int row, int col, int directionIndex): boolean
+ distancelBetweenCells(NewCell[][] board, NewCell startCell, NewCell arrivalCell): int
+ squaresInRadius2(Controller currentController,FictitiousPlayer player): ArrayList<NewCell>
PlayerManager
+ scoringProcess(Controller currentController)
+ scoreSingleBoard(Controller currentController, PlayerBoard board): char
+ dealPoints(Controller currentController, int boardValue, ArrayList<Character> dfRendersList)
+ listOfRenders(char[] damage): ArrayList<Character>
+ spawnPlayers(Controller controller, int id, PowerupCard spawn)
+ getCardsToSpawn(boolean setUpGame, Controller controller, int id)
+ choiceExecutor(Controller currentController, ChosenActions actions)
+ damageDealer(Controller currentController, Player target, char[] damageToDeal)
+ markerDealer(Player player, char[] add)
+ adrenalineManager(Player player)
+ adrenalineReload(Player player)
+ payGunCardCost(Player player, char [] cost, boolean fullOrReload)
+ reloadManager(Player player, boolean[] reload)
EvaluationCell
int dist
int x
int y
EvaluationCell(int dist, int x, int y)
PowerupManager
+ newtonManager(Controller currentController,int cardIndexInHand, Player player, int distance, int directionIndex)
+ teleporterManager(Controller currentController, int cardIndexInHand, NewCell destinationCell)
+ targetingScopeManager(Controller currentController, int cardIndexInHand)
+ grenadeManager(Controller currentController, Player playerDamaged,Player playerGivingDamage, int cardIndexInHand)
+ removeFromHand(Controller currentController,int cardIndexInHand)
AvailableActions
+ ArrayList<FictitiousPlayer>: fictitiousPlayers
+ AvailableActions(ActionRequestView macroAction, int playerId, Controller currentController)
- buildActions(Controller currentController, Player player, ActionRequestView macroAction, int maxMoveDistance, boolean grab, boolean shoot, boolean frenzyReload)
- createArrivalCells(NewCell[][] board, Player player,int minMoveDistance, int maxMoveDistance, boolean grab): ArrayList<CellInfo>
+ getFictitiousPlayers(): ArrayList<FictitiousPlayer>
CellInfo
- NewCell cell;
- boolean: canGrabCard;
- boolean :canGrabAmmo
+ CellInfo(NewCell arrivalCell, boolean canGrabCard, boolean canGrabAmmo)
+ getCell(): NewCell
+ isCanGrabCard(): boolean
+ isCanGrabAmmo(): boolean
CellWithTargets
- NewCell: targetCell
- ArrayList<Player>: targets
- int: maxTargetsInCell
- int: minTargetsInCell
- boolean: canMoveYourselfHere
- boolean: canMoveOthersHere
CellWithTargets(NewCell targetCell, ArrayList<Player> targets, int maxTargetsInCell, int minTargetsInCell, boolean canMoveYourselfHere, boolean canMoveOthersHere)
+ setMaxTargetsInCell(int maxTargetsInCell)
+ setMinTargetsInCell(int minTargetsInCell)
+ getTargetCell(): NewCell
+ getTargets(): ArrayList<Player>
+ getMaxTargetsInCell(): int
+ getMinTargetsInCell(): int
+ isCanMoveYourselfHere(): boolean
+ isCanMoveOthersHere(): boolean
FictitiousPlayer
- Player: correspondingPlayer
- Logger: LOGGER
- int: playerId
- Color: playerColor
- NewCell: position
- boolean: grabbedAmmo
- Ammo: availableAmmo
ArrayList<GunCard>: pickableCards
- ArrayList<SingleCardActions>: availableCardsActions
- boolean: noTargets
+ FictitiousPlayer (Controller currentController, Player player, CellInfo cell, boolean shoot, boolean frenzyReload)
+ evaluateUsableCards(Player player, boolean frenzyReload): ArrayList<GunCard>
+ getCorrespondingPlayer(): Player
+ isGrabbedAmmo(): boolean
+ getPlayerId(): int
+ getPickableCards(): ArrayList<GunCard>
+ getPlayerColor(): Color
+ getPosition(): NewCell
+ getAvailableAmmo(): Ammo
+ isNoTargets(): boolean
+ getAvailableCardActions(): ArrayList<SingleCardActions>
PlayerWithTargets
- Player: target
- ArrayList<Player>: targetsItCanSee
PlayerWithTargets(Controller currentController, Player target)
+ getTarget(): Player
+ getTargetsItCanSee(): ArrayList<Player>
SingleCardActions
- GunCard: gunToUse
- ArrayList<String>: availableCombinations
- ArrayList<SingleEffectsCombinationActions>: effectsCombinationActions
- boolean: mustSwap
+ SingleCardActions(Controller currentController, GunCard gunCard, FictitiousPlayer player, boolean mustSwap)
+ reduceToAffordableEffects(GunCard gunCard, ArrayList<ArrayList<String>> cardEffects,FictitiousPlayer player): ArrayList<ArrayList<String>>
+ getGunCardToUse(): GunCard
+ getAvailableCombinations(): ArrayList<String>
+ isMustSwap(): boolean
+ getEffectsCombinationActions(): ArrayList<SingleEffectsCombinationActions>
SingleEffectsCombinationActions
- ArrayList<String>: effectsCombination
- ArrayList<Player>: playersTargetList
- int: maxNumPlayerTargets
- int: minNumPlayerTargets
- boolean: canMoveOpponent
- boolean: canMoveYourself
- ArrayList<NewCell>: targetCells
- ArrayList<Room>: targetRooms
- boolean: offerableBase
- boolean: offerableOpt1
- boolean: offerableOpt2
- boolean: offerableExtra
- ArrayList<CellWithTargets>: cellsWithTargets
- int: minCellToSelect
- int: maxCellToSelect
- ArrayList<PlayerWithTargets>: playersWithTargets
+ SingleEffectsCombinationActions(ArrayList<String> effectsCombination)
+ validate(ArrayList<String> effectsCombination)
+ validateCellsWithTargets(): boolean
+ getEffectsCombination(): ArrayList<String>
+ getPlayersTargetsList(): ArrayList<Player>
+ getMaxNumPlayerTargets(): int
+ getMinNumPlayerTargets(): int
+ isCanMoveOpponent(): boolean
+ isCanMoveYourself(): boolean
+ getTargetCells(): ArrayList<NewCell>
+ getTargetRooms(): ArrayList<Room>
+ isOfferableExtra(): boolean
+ getCellsWithTargets(): ArrayList<CellWithTargets>
+ getMaxCellToSelect(): int
+ getPlayersWithTargets(): ArrayList<PlayerWithTargets>
+ isOfferableOpt1(): boolean
+ addToPlayerTarget_list(ArrayList<Player> targetSubList)
+ setMaxNumPlayerTargets(int maxNumberOfTargets)
+ setMinNumPlayerTargets(int minNumPlayerTargets)
+ addToTargetRooms(ArrayList<Room> targetRooms)
+ addToTargetCells(ArrayList<NewCell> targetCells)
+ setCanMoveOpponent(boolean canMoveOpponent)
+ setCanMoveYourself(boolean canMoveYourself)
+ setOfferableBase(boolean offerableBase)
+ setOfferableOpt2(boolean offerableOpt2)
+ setOfferableExtra(boolean offerableExtra)
+ addCellsWithTargets(NewCell targetCell, ArrayList<Player> targets, int maxTargets, int minTargets,boolean canMoveYourSelfHere, boolean canMoveOthersHere)
+ setMinCellToSelect(int minCellToSelect)
+ setMaxCellToSelect(int maxCellToSelect)
+ addPlayersWithTargets(Controller currentController, Player basePlayer)