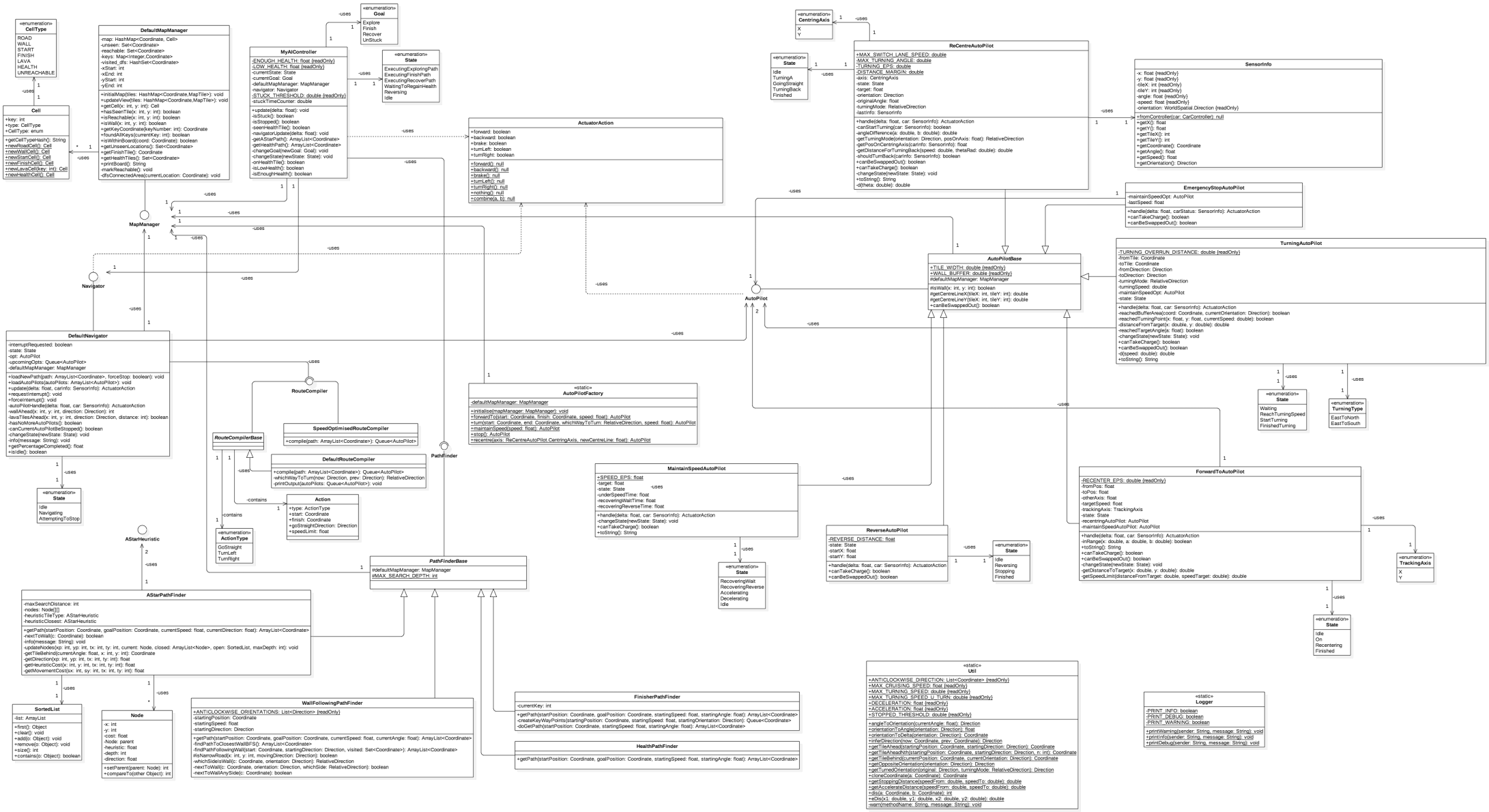
«enumeration»
CellType
ROAD
WALL
START
FINISH
LAVA
HEALTH
UNREACHABLE
Cell
+key: int
+type: CellType
+CellType: enum
+getCellTypeHash(): String
+newRoadCell(): Cell
+newWallCell(): Cell
+newStartCell(): Cell
+newFinishCell(): Cell
+newLavaCell(key: int): Cell
+newHealthCell(): Cell
DefaultMapManager
-map: HashMap<Coordinate, Cell>
-unseen: Set<Coordinate>
-reachable: Set<Coordinate>
-keys: Map<Integer,Coordinate>
-visited_dfs: HashSet<Coordinate>
-xStart: int
-xEnd: int
-yStart: int
-yEnd: int
+initialiseMap(tiles: HashMap<Coordinate,MapTile>): void
+updateView(tiles: HashMap<Coordinate,MapTile>): void
+getCell(x: int, y: int): Cell
+hasSeenTile(x: int, y: int): boolean
+isReachable(x: int, y: int): boolean
+isWall(x: int, y: int): boolean
+getKeyCoordinate(keyNumber: int): Coordinate
+foundAllKeys(currentKey: int): boolean
+isWithinBoard(coord: Coordinate): boolean
+getUnseenLocations(): Set<Coordinate>
+getFinishTile(): Coordinate
+getHealthTiles(): Set<Coordinate>
+printBoard(): String
-markReachable(): void
-dfsConnectedArea(currentLocation: Coordinate): void
«enumeration»
Goal
Explore
Finish
Recover
UnStuck
MyAIController
-ENOUGH_HEALTH: float {readOnly}
-LOW_HEALTH: float {readOnly}
-currentState: State
-currentGoal: Goal
-defaultMapManager: MapManager
-navigator: Navigator
-STUCK_THRESHOLD: double {readOnly}
-stuckTimeCounter: double
+update(delta: float): void
-isStuck(): boolean
-isStopped(): boolean
-seenHealthTile(): boolean
-navigatorUpdate(delta: float): void
-getAStarPath(): ArrayList<Coordinate>
-getHealthPath(): ArrayList<Coordinate>
-changeGoal(newGoal: Goal): void
-changeState(newState: State): void
-onHealthTile(): boolean
-isLowHealth(): boolean
-isEnoughHealth(): boolean
«enumeration»
State
ExecutingExploringPath
ExecutingFinishPath
ExecutingRecoverPath
WaitingToRegainHealth
Reversing
Idle
ActuatorAction
+forward: boolean
+backward: boolean
+brake: boolean
+turnLeft: boolean
+turnRight: boolean
+forward(): null
+backward(): null
+brake(): null
+turnLeft(): null
+turnRight(): null
+nothing(): null
+combine(a, b): null
«enumeration»
CentringAxis
X
Y
«enumeration»
State
Idle
TurningA
GoingStraight
TurningBack
Finished
ReCentreAutoPilot
+MAX_SWITCH_LANE_SPEED: double
-MAX_TURNING_ANGLE: double
-TURNING_EPS: double
-DISTANCE_MARGIN: double
-axis: CentringAxis
-state: State
-target: float
-orientation: Direction
-originalAngle: float
-turningMode: RelativeDirection
-lastInfo: SensorInfo
+handle(delta: float, car: SensorInfo): ActuatorAction
-carStartTurning(car: SensorInfo): boolean
-angleDifference(a: double, b: double): double
-getTurningMode(orientation: Direction, posOnAxis: float): RelativeDirection
-getPosOnCentringAxis(carInfo: SensorInfo): float
-getDistanceForTurningBack(speed: double, thetaRad: double): double
-shouldTurnBack(carInfo: SensorInfo): boolean
+canBeSwappedOut(): boolean
+canTakeCharge(): boolean
-changeState(newState: State): void
+toString(): String
-d(theta: double): double
SensorInfo
-x: float {readOnly}
-y: float {readOnly}
-tileX: int {readOnly}
-tileY: int {readOnly}
-angle: float {readOnly}
-speed: float {readOnly}
-orientation: WorldSpatial.Direction {readOnly}
+fromController(car: CarController): null
+getX(): float
+getY(): float
+getTileX(): int
+getTileY(): int
+getCoordinate(): Coordinate
+getAngle(): float
+getSpeed(): float
+getOrientation(): Direction
EmergencyStopAutoPilot
-maintainSpeedOpt: AutoPilot
-lastSpeed: float
+handle(delta: float, carStatus: SensorInfo): ActuatorAction
+canTakeCharge(): boolean
+canBeSwappedOut(): boolean
MapManager
Navigator
AutoPilot
AutoPilotBase
+TILE_WIDTH: double {readOnly}
+WALL_BUFFER: double {readOnly}
#defaultMapManager: MapManager
#isWall(x: int, y: int): boolean
#getCentreLineX(tileX: int, tileY: int): double
#getCentreLineY(tileX: int, tileY: int): double
+canBeSwappedOut(): boolean
TurningAutoPilot
-TURNING_OVERRUN_DISTANCE: double {readOnly}
-fromTile: Coordinate
-toTile: Coordinate
-fromDirection: Direction
-toDirection: Direction
-turningMode: RelativeDirection
-turningSpeed: double
-maintainSpeedOpt: AutoPilot
-state: State
+handle(delta: float, car: SensorInfo): ActuatorAction
-reachedBufferArea(coord: Coordinate, currentOrientation: Direction): boolean
-reachedTurningPoint(x: float, y: float, currentSpeed: double): boolean
-distanceFromTarget(x: double, y: double): double
-reachedTargetAngle(a: float): boolean
-changeState(newState: State): void
+canTakeCharge(): boolean
+canBeSwappedOut(): boolean
-d(speed: double): double
+toString(): String
«enumeration»
State
Waiting
ReachTurningSpeed
StartTurning
FinishedTurning
«enumeration»
TurningType
EastToNorth
EastToSouth
DefaultNavigator
-interruptRequested: boolean
-state: State
-opt: AutoPilot
-upcomingOpts: Queue<AutoPilot>
-defaultMapManager: MapManager
+loadNewPath(path: ArrayList<Coordinate>, forceStop: boolean): void
+loadAutoPilots(autoPilots: ArrayList<AutoPilot>): void
+update(delta: float, carInfo: SensorInfo): ActuatorAction
+requestInterrupt(): void
+forceInterrupt(): void
-autoPilotHandle(delta: float, car: SensorInfo): ActuatorAction
-wallAhead(x: int, y: int, direction: Direction): int
-lavaTilesAhead(x: int, y: int, direction: Direction, distance: int): boolean
-hasNoMoreAutoPilots(): boolean
-canCurrentAutoPilotBeStopped(): boolean
-changeState(newState: State): void
-info(message: String): void
+getPercentageCompleted(): float
+isIdle(): boolean
«enumeration»
State
Idle
Navigating
AttemptingToStop
RouteCompiler
SpeedOptimisedRouteCompiler
+compile(path: ArrayList<Coordinate>): Queue<AutoPilot>
RouteCompilerBase
DefaultRouteCompiler
+compile(path: ArrayList<Coordinate>): Queue<AutoPilot>
-whichWayToTurn(now: Direction, prev: Direction): RelativeDirection
-printOutput(autoPilots: Queue<AutoPilot>): void
Action
+type: ActionType
+start: Coordinate
+finish: Coordinate
+goStraightDirection: Direction
+speedLimit: float
«enumeration»
ActionType
GoStraight
TurnLeft
TurnRight
«static»
AutoPilotFactory
-defaultMapManager: MapManager
+initialise(mapManager: MapManager): void
+forwardTo(start: Coordinate, finish: Coordinate, speed: float): AutoPilot
+turn(start: Coordinate, end: Coordinate, whichWayToTurn: RelativeDirection, speed: float): AutoPilot
+maintainSpeed(speed: float): AutoPilot
+stop(): AutoPilot
+recentre(axis: ReCentreAutoPilot.CentringAxis, newCentreLine: float): AutoPilot
PathFinder
MaintainSpeedAutoPilot
+SPEED_EPS: float
-target: float
-state: State
-underSpeedTime: float
-recoveringWaitTime: float
-recoveringReverseTime: float
+handle(delta: float, car: SensorInfo): ActuatorAction
-changeState(newState: State): void
+canTakeCharge(): boolean
+toString(): String
«enumeration»
State
RecoveringWait
RecoveringReverse
Accelerating
Decelerating
Idle
ReverseAutoPilot
-REVERSE_DISTANCE: float
-state: State
-startX: float
-startY: float
+handle(delta: float, car: SensorInfo): ActuatorAction
+canTakeCharge(): boolean
+canBeSwappedOut(): boolean
«enumeration»
State
Idle
Reversing
Stopping
Finished
ForwardToAutoPilot
-RECENTER_EPS: double {readOnly}
-fromPos: float
-toPos: float
-otherAxis: float
-targetSpeed: float
-trackingAxis: TrackingAxis
-state: State
-recentringAutoPilot: AutoPilot
-maintainSpeedAutoPilot: AutoPilot
+handle(delta: float, car: SensorInfo): ActuatorAction
-inRange(x: double, a: double, b: double): boolean
+toString(): String
+canTakeCharge(): boolean
+canBeSwappedOut(): boolean
-changeState(newState: State): void
-getDistanceToTarget(x: double, y: double): double
-getSpeedLimit(distanceFromTarget: double, speedTarget: double): double
«enumeration»
TrackingAxis
X
Y
«enumeration»
State
Idle
On
Recentering
Finished
AStarHeuristic
AStarPathFinder
-maxSearchDistance: int
-nodes: Node[][]
-heuristicTileType: AStarHeuristic
-heuristicClosest: AStarHeuristic
+getPath(startPosition: Coordinate, goalPosition: Coordinate, currentSpeed: float, currentDirection: float): ArrayList<Coordinate>
-nextToWall(c: Coordinate): boolean
-info(message: String): void
-updateNodes(xp: int, yp: int, tx: int, ty: int, current: Node, closed: ArrayList<Node>, open: SortedList, maxDepth: int): void
-getTileBehind(currentAngle: float, x: int, y: int): Coordinate
-getDirection(xp: int, yp: int, tx: int, ty: int): float
-getHeuristicCost(x: int, y: int, tx: int, ty: int): float
-getMovementCost(sx: int, sy: int, tx: int, ty: int): float
PathFinderBase
#defaultMapManager: MapManager
#MAX_SEARCH_DEPTH: int
SortedList
-list: ArrayList
+first(): Object
+clear(): void
+add(o: Object): void
+remove(o: Object): void
+size(): int
+contains(o: Object): boolean
Node
-x: int
-y: int
-cost: float
-Node: parent
-heuristic: float
-depth: int
-direction: float
+setParent(parent: Node): int
+compareTo(other Object): int
WallFollowingPathFinder
+ANTICLOCKWISE_ORIENTATIONS: List<Direction> {readOnly}
-startingPosition: Coordinate
-startingSpeed: float
-startingDirection: Direction
+getPath(startPosition: Coordinate, goalPosition: Coordinate, currentSpeed: float, currentAngle: float): ArrayList<Coordinate>
-findPathToClosestWallBFS(): ArrayList<Coordinate>
-findPathFollowingWall(start: Coordinate, startingDirection: Direction, visited: Set<Coordinate>): ArrayList<Coordinate>
#isNarrowRoad(x: int, y: int, movingDirection: Direction): boolean
-whichSideIsWall(c: Coordinate, orientation: Direction): RelativeDirection
-nextToWall(c: Coordinate, orientation: Direction, whichSide: RelativeDirection): boolean
-nextToWallAnySide(c: Coordinate): boolean
FinisherPathFinder
-currentKey: int
+getPath(startPosition: Coordinate, goalPosition: Coordinate, startingSpeed: float, startingAngle: float): ArrayList<Coordinate>
-createKeyWayPoints(startingPosition: Coordinate, startingSpeed: float, startingOrientation: Direction): Queue<Coordinate>
-doGetPath(startPosition: Coordinate, startingSpeed: float, startingAngle: float): ArrayList<Coordinate>
HealthPathFinder
+getPath(startPosition: Coordinate, goalPosition: Coordinate, startingSpeed: float, startingAngle: float): ArrayList<Coordinate>
«static»
Util
+ANTICLOCKWISE_DIRECTION: List<Coordinate> {readOnly}
+MAX_CRUISING_SPEED: float {readOnly}
+MAX_TURNING_SPEED: double {readOnly}
+MAX_TURNING_SPEED_U_TURN: double {readOnly}
+DECELERATION: float {readOnly}
+ACCELERATION: float {readOnly}
+STOPPED_THRESHOLD: double {readOnly}
+angleToOrientation(currentAngle: float): Direction
+orientationToAngle(orientation: Direction): float
+orientationToDelta(orientation: Direction): Coordinate
+inferDirection(now: Coordinate, prev: Coordinate): Direction
+getTileAhead(startingPosition: Coordinate, startingDirection: Direction): Coordinate
+getTileAheadNth(startingPosition: Coordinate, startingDirection: Direction, n: int): Coordinate
+getTileBehind(currentPosition: Coordinate, currentOrientation: Direction): Coordinate
+getOppositeOrientation(orientation: Direction): Direction
+getTurnedOrientation(original: Direction, turningMode: RelativeDirection): Direction
+cloneCoordinate(a: Coordinate): Coordinate
+getStoppingDistance(speedFrom: double, speedTo: double): double
+getAccelerateDistance(speedFrom: double, speedTo: double): double
+dis(a: Coordinate, b: Coordinate): int
+dDis(x1: double, y1: double, x2: double, y2: double): double
-warn(methodName: String, message: String): void
«static»
Logger
-PRINT_INFO: boolean
-PRINT_DEBUG: boolean
-PRINT_WARNING: boolean
+printWarning(sender: String, message: String): void
+printInfo(sender: String, message: String): void
+printDebug(sender: String, message: String): void
-uses