Logger

Logger
- writer: PrintWriter
- logInitializationInfoSuccess(filename:String)
- logInitializationInfoFail(filename:String)
- logSimulationStarted()
- logSimulationEnded()
- logSimulationTick(tick:Int)
- logCorporationStartMoveShips(corporationId:Int)
- logShipMovement(shipId:Int,speed:Int,tileId:Int)
- logCorporationStartCollectGarbage(corporateId)
- logGarbageCollectionByShip(shipId:Int,garbageType:GarbageType,garbageId:Int)
- logCorporationCooperationStart(corporationId:Int)
- logCooperationBetweenCorporations(corporationId:Int, otherCorporationId: Int, shipId:Int, cooperatedShipId: Int)
- logCorporationRefueling(corporationId:Int)
- logRefuelingShip(shipId:Int, tileId: Int)
- logUnloadShip(shipId:Int, amount:Int, garbageType:GarbageType, tileId:Int)
- logCorporationFinishedActions(corporationId:Int)
- logCurrentDriftGarbage(garbageType:GarbageType, garbageId:Int, amount:Int, startTileId:Int, endTileId:Int)
- logCurrentDriftShip(shipId:Int,startTileId:Int, endTileId:Int)
- logEventStart(eventId:Int,eventType:EventType)
- logSimulationStatisticsCalculated()
- logSimulationStatistics()

Tile

Vec2D
- posX: Int,
- posY: Int

Harbor
- owner: List<Corporation>,
- pos: Tile,

Current
- speed: Int,
- direction: Direction,
- intensity: Int,
- getSpeed(): Int
- getIntensity(): Int
- getDirection(): Direction

Garbage
- id: Int,
- amount: Int,
- type: GarbageType,
- static maxId: Int,
- drift(Tile)
- static getNextId(): Int
- canDriftAtOnce(Int): Boolean
- removeAmount(Int): void

Land

Shore
- harbor: Harbor?
- getHarbor(): Harbor?

ShallowOcean

DeepOcean
- current: Current?
- getCurrent(): Current?

Tile (abstract)
- id: Int,
- pos: Vec2D,
- adjacentTiles: Tile[6],
- shipTraversable: Boolean,
- garbage: List<Garbage>
- Tile(id: Int, pos: Vec2D, shipTraversable): Tile
- isSpaceAvailable(Int): Boolean
- driftGarbage(GarbageType, Int): void
- getTileInDirection(distance: Int, dir: Direction): Tile
- addGarbage(g: Garbage): void
- isOilPresent(): Boolean
- isPlasticPresent(): Boolean
- areChemicalsPresent(): Boolean

Ship

CollectingShip
- garbageType: GarbageType
- garbageCapacity: Int,
- garbageLoad: Int,
- giveGarbage()

CoordinatingShip

ScoutingShip

Ship (abstract)
- id: Int,
- name: String,
- owner: Corporation,
- pos: Tile,
- dir: Direction,
- maxVelocity: Int,
- acceleration: Int,
- fuelCapacity: Int,
- fuelConsumption: Int,
- visibleRange: Int,
- destinationTile: Tile,
- navigateTo(Tile)
- refuel()
- giveGarbage()
- getTilesWithGarbageInFoV(): List<Tile>
- drift()
- updateTile(Tile)

GeneralEnums

GarbageType
OIL,
PLASTIC,
CHEMICALS

Direction
D0,
D60,
D120,
D180,
D240,
D300

Coordinator
- main(mapFile : String, corporationsFile: String, scenarioFile: String, max_ticks: Int)
- parse(): Accumulator

Map
- tiles: List<Tile>
- tileIndex: Map<Vec2D, Tile>
- getTileById(Int): Tile
- getTiles(): List<Tile>
- addTile(Tile): void
- getTileByPosition(Vec2D): Tile
- getTilesByInstance(instanceOf(Tile))

Dijkstra
- execute(Tile): Map<Tile,Int>

Corporation
- id:Int,
- partner:Corporation
- name:String,
- acceptedGarbageTypes: List<GarbageType>
- ownedHarbors: List<Harbor>
- ownedShips: List<Ship>
- moveShips(),
- collectGarbage(),
- refuelShips()
- unloadShips()
- cooperateWith(c1:Corporation)
- run()

Simulation
- corporations: List<Corporation>
- allEvents: List<Event>
- tick: Int
- maxTick: Int
- map: Map
- start()
- tick()
- runCorporations()
- processEvents()

Event

OilSpill
- amount: Int,

Storm
- speed: Int,
- direction: Direction

Restriction
- duration: Int,

Local_Event (abstract)
- location: Tile,
- radius: Int,

Pirate_Attack
- ship: Ship,

Event (abstract)
- id: Int,
- act(Simulation)

Parser

ParsedObject
- properties:Map<String,String>

TileCategory
LAND,
SHORE,
SHALLOW_OCEAN,
DEEP_OCEAN

Accumulator
- parsedTiles: List<ParsedObject>
- parsedCorporations: List<ParsedObject>
- parsedShips: List<ParsedObject>
- parsedGarbages: List<ParsedObject>
- parsedEvents: List<ParsedObject>
- currentValidatingObject: ParsedObject
- map: Map
- tiles: Map<Int, Tile>
- tilesByCoordinat: Map<Vec2D, Tile>
- corporations: Map<Int, Corporation>
- ships: Map<Int, Ship>
- events: Map<Int, Event>
- garbage: Map<Int, Garbage>
- mapCorporationToHarbor: Map<Int, List<Tile>>
- mapCorporationToShips: Map<Int, List<Int>>
- Getters()
- addTile(id: Int, t: Tile): Unit
- addTileByCoordinat(coord: Vec2D, t: Tile): Unit
- addShip(shipId: Int, ship: Ship)
- addShipToCorp(corporationId: Int, shipId: Int)
- getShipsOfCorp(corpId: Int): List<Int>
- addEvent(eventId: Int, event: Event)
- addCorporation(corpId: Int, corporation: Corporation)
- addGarbage(garbageId: Int, garbage: Garbage)

MapJSONParser
- parseMap(jsonFile:String): Boolean
- createTileObjects(filePath: String): List<ParsedObject>
- validateTiles(objects: List<ParsedObject>): Boolean
- validateTile(tile: ParsedObject): Boolean
- validateUniqueId(id: Int): Boolean
- validateCoordinates(coord: Vec2D): Boolean
- validateRequirements(tileObject: ParsedObject): Boolean

JSONParser (interface)
- logger: Logger
- accumulator: Accumulator
- validateObjects(object:List<ParsedObject>)

CorporationJSONParser
- CorporationJSONParser(accumulator:Accumulator)
- parseCorporations(filePath: String): Boolean
- parseShips(filePath: String): Boolean
- createCorporationObjects(file: String): List<ParsedObject>
- validateCorporations(corpObjects: List<ParsedObject>): Boolean
- validateCorporation(corporationObject: ParsedObject): Boolean
- validateCorporationUniqueId(id: Int): Boolean
- validateCorporationUniqueName(name: String): Boolean
- validateHarborTiles(harbors: List<Int>): Boolean
- validateCorporationGarbageTypes(g: List<GarbageType>): Boolean
- validateCorporationShip(ships: List<Int>): Boolean
- createShipObjects(file: String): List<ParsedObject>
- validateShips(ships: List<ParsedObject>):Boolean
- validateShip(ship:ParsedObject): Boolean
- validateShipUniqueId(id:Int): Boolean
- validateShipUniqueName(name:String): Boolean
- validateShipLocation(location: Int): Boolean
- validateShipAttributes(ship:ParsedObject): Boolean

ScenarioJSONParser
- ScenarioJSONParser(accumulator:Accumulator)
- parseEvents(json: String)
- parseGarbage(garbage: String): Garbage
- validateGarbages(garbages:List<ParsedObject>)
- validateGarbageUniqueId(): Boolean
- validateGarbageType(): Boolean
- validateGarbageLocationExists(): Boolean
- validateGarbageAmount(): Boolean
- createGarbage(garbage: ParsedObject)
- validateEvents(events:List<ParsedObject>)
- validateEvent(event: ParsedObject)
- validateEventUniqueId(): Boolean
- validateEventType(): Boolean
- validateEventTick(): Boolean
- validateEventLocation(): Boolean
- validateEventRadius(): Boolean
- validateEventSpeed(): Boolean
- validateEventAmountIfExists(): Boolean
- validateEventDirectionIfExists(): Boolean
- validateEventShipIdIfExists(): Boolean
- createEvent(event: ParsedObject)