User
Main
Logger
main(map.json, corporations.json, scenario.json, 0)
Accumulator()
accu: Accumulator
accu
MapJSONParser(accu)
m: MapJSONParser
m
parseMap(map.json)
createTileObjects(map.json)
objects: List<JSONObject>
validateTiles(objects)
loop
[for tileObject in objects]
validateTile(tileObject)
True
Tiles(id, coord, shipTraversable)
t:Tiles
t
addTile(id, t)
addTileByCoordinat(coord, t)
True
createMap()
Map(accu.tiles.getValues(), accu.tilesbyCoord)
s: Sea
s
s
setMap(s)
logInitializationInfoSuccess(maps.json)
True
CorporationJSONParser(accu)
c: CorporartionJSONParser
c
parseShips(corporations.json)
createShipObjects(corporations.json)
objects: List<JSONObject>
validateShips(objects)
loop
[for shipObject in objects]
ValidateShip(shipObject)
True
createShip(shipObject)
Ship(id, name, tile, maxVelocity, acceleration, fuelCapacity, fuelConsumption, visibilityRange, garbageType, capacity)
s:Ship
s
s
addShip(id, s)
addShipToCorp(corporationId, id)
True
True
True
parseCorporations(corporations.json)
createCorporationObjectst(corporations.json)
objects: List<JSONObject>
validateCorporations(objects)
loop
[for corporationObject in objects]
ValidateCorporation(corporationObject)
loop
[for id in homeHarbors]
getTilebyId(id)
t
loop
[for id in ships]
getShipById(id)
s
getShipsOfCorp(id)
shouldBeShips: List<Int>
Corporation(id, name, ships, homeHarbors, garbage)
corporation: Corporation
corp
addCorporation(id, corp)
True
True
logInitializationInfoSuccess(corporations.json)
True
ScenarioJSONParser(accu)
Sc: ScenarioJSONParser
Sc
parseGarbage(Scenario.json)
createGarbageObjects(Scenario.json)
objects: List<JSONObject>
validateGarbages(objects)
loop
[for garbageObject in objects]
validateGarbage(garbageObject)
getTilebyId(locationId)
t
True
createGarbage(garbageObject)
Garbage(id, amount, type, t)
g: Garbage
g
g
addGarbage(id, g)
True
True
True
parseEvents(Scenario.json)
createEventObjects(Scenario.json)
objects: List<JSONObject>
validateEvents(objects)
loop
[eventObject in objects]
validateEvent(eventObject)
createEvent(eventObject)
Event(id, type, tick, duration, location, radius, speed, amount, direction, shipID)
event: Event
event
event
addEvent(id, event)
True
True
True
parseRewards(Scenario.json)
validateRewards(rewards)
loop
[rewardObject in rewards]
validateReward(rewardObject)
createReward(rewardObject)
Reward(id)
reward: Reward
reward
reward
addReward(id, reward)
True
True
True
parseTasks(Scenario.json)
validateTasks(tasks)
loop
[taskObject in tasks]
validateTask(taskObject)
getRewardById(rewardId)
reward
getShipById(taskShipId)
taskShip
getShipById(rewardShipId)
rewardShip
ship1.getOwner().equals(ship2.getOwner())
createTask(taskObject)
Task(id, tick, taskShip, reward, rewardShip, ship1.getOwner())
task: Task
task
task
addEvent(id, task)
True
True
True
loop
[for g in garbage]
m.getTilebyId(g.getLocationId()).addGarbage(g)
Simulation(corporations, events, 0, 0, m, tasks)
s: Simulation
s
User
Main
accu: Accumulator
m: MapJSONParser
t:Tiles
s: Sea
Logger
c: CorporartionJSONParser
s:Ship
corporation: Corporation
Sc: ScenarioJSONParser
g: Garbage
event: Event
reward: Reward
task: Task
s: Simulation