MODEL UML

Game
- componentDeck :ArrayList<Component>
- eventCardDeck :ArrayList<EventCard>
- players :ArrayList<Player>
- level :int
- phase :GamePhase
- board :Board
- id :int
- maxNumPlayers :int
- visibleComponentDeck :ArrayList<Component>
- activeEventCard :EventCard
- + saveGame () :void
- + pickEventCard () :EventCard
- + addPlayer (in player :Player) :void
- + loadComponents () :ArrayList<Component>
- + loadEvents () :ArrayList<EventCard>
- + getPhase () :GamePhase
- + setPhase (in :GamePhase :enum)
- + getLevel () :int
- + getPlayers () :ArrayList<Player>
- + getBoard () :Board
- + tryAddPlayer (in name :String) :boolean
- + elaborateSizeBoardFromLv () :int
- + getId () :int
- + getPlayersSize () :int
- + pickHiddenComponent (in player :Player) :Component
- + setActiveEventCard (in eventCard :EventCard)
- + pickVisibleComponent (in player :Player, in indexComponent :int)
- + getMaxNumPlayers () :int
- + endGame () :ArrayList<Player>

GamePhase «enumeration»
- Init
- Building
- Travel
- Event
- EndGame

Battlezone
- couples :ArrayList<ConditionPenalty>
- + getCouples () :ArrayList<ConditionPenalty>
- + penaltyDays (in board :Board, in player :Player, in penaltyDays :int) :void
- + chooseDiscardedCrew (in component :StorageComponent) :boolean
- + checkShields (in player :Player, in shot :Projectile) :boolean
- + penaltyShot (in game :Game, in player :Player, in shot :Projectile, in position :int) :boolean
- + chooseDiscardedBox (in player :Player, in component :BoxStorageComponent, in boxIdx :int) :boolean
- + chooseDiscardedBattery (in component :StorageComponent) :boolean

Condition
- type :ConditionType
- greater :boolean
- fewer :boolean
- + getType () :ConditionType
- + getGrater () :boolean
- + getfewer () :boolean

ConditionType «enumeration»
- CrewRequirement
- FirePowerRequirement
- EnginePowerRequirement

ConditionPenalty
- condition :Condition
- penalty :Penalty
- + getCondition (in condition :Condition)
- + getPenalty (in penalty :Penalty)

LostShip
- penaltyCrew :int
- rewardCredits :int
- penaltyDays :int
- + getPenaltyCrew () :int
- + getRewardCredits () :int
- + getPenaltyDays () :int
- + chooseDiscardedCrew (in component :StorageComponent) :boolean
- + rewardPenalty (in board :Board, in player :Player) :void

Penalty
- type :PenaltyType
- neededAmount :int
- shots :ArrayList<Projectile>
- + getType () :PenaltyType
- + getNeededAmount () :int
- + getMeteors () :ArrayList<Projectile>

PenaltyType «enumeration»
- PenaltyDays
- PenaltyCrew
- PenaltyShots
- PenaltyBoxes

Planets
- planetsTaken[] :boolean
- rewardsForPlanets :ArrayList<ArrayList<Box>>
- landedPlayers :Player[]
- penaltyDays :int
- + getPlanetsTaken () :boolean[]
- + getRewardsForPlanets () :ArrayList<ArrayList<Box>>
- + getPenaltyDays () :int
- + choosePlanet (in player :Player, in planetIdx :int) :boolean
- + chooseRewardsBox (in component :BoxStorageComponent, in boxIdx :int, in box :Box) :boolean
- + penalty (in board :Board) :void

Stardust
- penaltyDays :int
- + getPenaltyDays () :int
- + penalty (in board :Board, in player :Player) :void

Pirates
- firePowerRequired :int
- penaltyShots :ArrayList<Projectile>
- penaltyDays :int
- rewardCredits :int
- + getFirePowerRequired () :int
- + getPenaltyShots () :arrayOf(Projectile)
- + getPenaltyDays () :int
- + getRewardCredits () :int
- + checkShields (in player :Player, in shot :Projectile) :boolean
- + penaltyShot (in game :Game, in player :Player, in shot :Projectile, in position :int) :boolean
- + rewardPenalty (in board :Board, in player :Player) :void
- + battleResult (in player :Player, in firePower :int) :int
- + chooseDiscardedBattery (in component :StorageComponent) :boolean

EventCard
- type :CardType
- imgSrc :String
- + getType () :enum
- + setType (in type :enum)
- + getImgScr () :String

CardType «enumeration»
- OpenSpace
- MeteorsRain
- Slavers
- Smugglers
- LostStation
- Sabotage
- Epidemic
- Battlezone
- Pirates
- Stardust
- Planets
- LostShip

Board
- track :Player[]
- imgSrc :String
- activeTravelers :ArrayList<Player>
- + movePlayerByDistance (in player :Player, in distance :int) :void
- + getTrack () :Player[]
- + getImgSrc () :String
- + leaveTravel (in player :Player)
- + addTraveler (in player :Player)

BuildingBoard
- spaceshipMatrix :Component[][]
- booked :ArrayList<Component>
- imgSrc :String
- boardMask :int[][]
- boardComponent :Component
- spaceship :Spaceship
- destroyComponent (in row :int, in column :int) :void
- insertComponent (in row :int, in column :int) :boolean
- setAsBooked (in component :Component) :boolean
- getspaceshipMatrix () :Component[][]
- getBooked () :ArrayList<Component>
- getSpaceship () :Spaceship
- chooseRemainingPart () :void
- fixConnections () :void
- getImgSrc () :String
- getBoardMask () :int[][]
- loadBoardMask () :int[][]
- createSpaceshipMatrix () :Component[][]
- areConnected (in c1 :Component, c2 :Component) :boolean
- placeCentralUnit (in levelShip :int, in color :int)
- checkShipValidity ()
- dfsValidity (in x :int, in y :int, in visited :boolean[][], in numComponentChecked :AtomicInteger)
- checkCannonValidity (in cannon :Component, in x :int, in y :int)

Player
- name :String
- credits :int
- color :int
- position :int
- spaceship :Spaceship
- getName () :String
- getColor () :int
- getCredits () :int
- getPositions () :int
- getSpaceship () :Spaceship
- setPosition (in newPosition :int) :void
- addCredits (in creditsToAdd :int) :void
- rollDice () :int

Component
- type :ComponentType
- connections :int[4]
- rotation :int
- placed :boolean
- imgSrc :String
- x_coordinate :int
- y_coordinate :int
- + rotate () :void
- + setRotation () :void
- + setHidden () :void
- + setPlaced () :void
- + getType () :ComponentType
- + getConnections () :int[4]
- + getRotation () :int
- + getImgSrc () :String
- + isPlaced () :boolean
- + getPlaced () :boolean
- + getXCoordinate () :int
- + getYCoordinate () :int
- + setXCoordinate (in x :int) :void
- + setYCoordinate (in y :int) :void
- + printBoard () :void

ComponentType «enumeration»
- Cannon
- DoubleCannon
- Engine
- DoubleEngine
- Shield
- HousingUnit
- OrangeHousingUnit
- PurpleHousingUnit
- CentralUnit
- StructuralUnit
- BatteryStorage
- RedBoxStorage
- BoxStorage

Spaceship
- destroyedCount :int
- crewCount :int
- batteriesCount :int
- exposedConnectorsCount :int
- alienPurple :boolean
- alienOrange :boolean
- normalShootingPower :float
- normalEnginePower :float
- shieldCounts :int[4]
- buildingBoard :BuildingBoard
- doubleCannonCount :int
- doubleEngineCount :int
- boxCounts :int[4]
- componentShipCount :int
- + getDestroyedCount () :int
- + getCrewCount () :int
- + getBoxesValue () :int
- + getBatteriesCount () :int
- + getExposedConnectorsCount () :int
- + getAlienPurple () :boolean
- + getAlienOrange () :boolean
- + getNormalShootingPower () :float
- + getDoubleCannonCount () :int
- + getDoubleEngineCount () :int
- + getNormalEnginePower () :float
- + getIsShield (in index :int) :int
- + getBoxCounts () :int[]
- + getBuildingBoard () :BuildingBoard
- + setExposedConnectorsCount (in count :int)
- + setAlienPurple (in alienPurple :boolean) :void
- + setAlienOrange (in alienPresence :boolean)
- + addDestroyedCount (in countToAdd :int)
- + addCrewCount (in countToAdd :int)
- + addBatteriesCount (in countToAdd :int)
- + addNormalShootingPower (in powerToAdd :float)
- + addDoubleCannonCount (in countToAdd :int)
- + addNormalEnginePower (in powerToAdd :float)
- + addDoubleEngineCount (in countToAdd :int)
- + addUpRightShieldCount (in countToAdd :int)
- + addRightDownShieldCount (in countToAdd :int)
- + addDownLeftShieldCount (in countToAdd :int)
- + addLeftUpShieldCount (in countToAdd :int)
- + addRedBoxCount (in countToAdd :int)
- + addYellowBoxCount (in countToAdd :int)
- + addGreenBoxCount (in countToAdd :int)
- + addBlueBoxCount (in countToAdd :int)
- + getComponentShipCount () :int
- + addComponentShipCount (in countToAdd :int)

MeteorsRain
- meteors :ArrayList<Projectile>
- + getMeteors () :ArrayList<Projectile>
- + checkImpactComponent (in game :Game, in player :Player, in shot :Projectile, in position :int) :Component
- + checkShields (in player :Player, in shot :Projectile) :boolean
- + chooseDiscardedBattery (in component :StorageComponent) :boolean

OpenSpace
- + moveAhead (in board :Board, in player :Player, in enginePower :int) :void
- + chooseDiscardedBattery (in component :StorageComponent) :boolean

Slavers
- firePowerRequired :int
- penaltyCrew :int
- penaltyDays :int
- rewardCredits :int
- + getFirePowerRequired () :int
- + getPenaltyCrew () :int
- + getPenaltyDays () :int
- + getRewardCredits () :int
- + chosenDiscardedCrew (in component :StorageComponent) :boolean
- + rewardPenalty (in board :Board, in player :Player) :void
- + battleResult (in player :Player, in firePower :int) :int

Smugglers
- firePowerRequired :int
- penaltyBoxes :int
- penaltyDays :int
- rewardBoxes :ArrayList<Box>
- + getFirePowerRequired () :int
- + getPenaltyBoxes () :int
- + getPenaltyDays () :int
- + getRewardBoxes () :ArrayList<Box>
- + chooseRewardBox (in component :BoxStorageComponent, in boxIdx :int, in box :Box) :boolean
- + chooseDiscardedBox (in player :Player, in component :BoxStorageComponent, in boxIdx :int) :boolean

LostStation
- requiredCrew :int
- rewardBoxes :ArrayList<Box>
- penaltyDays :int
- + getRequiredCrew () :int
- + getRewardBoxes () :ArrayList<Box>
- + getPenaltyDays () :int
- + chooseRewardBox (in component :BoxStorageComponent, in boxIdx :int, in box :Box) :boolean
- + chooseDiscardedBox (in player :Player, in component :BoxStorageComponent, in boxIdx :int) :boolean
- + chooseDiscardedBattery (in component :StorageComponent) :boolean
- + penalty (in board :Board, in player :Player) :void

Sabotage
- lessPopulatedSpaceship (in players :Player[]) :Player
- + penalty (in row :int, in column :int, in player :Player) :boolean

Epidemic
- + findInfectedComponents (in i :int, in j :int, in firstIteration :boolean, in prevComponent :StorageComponent, in buildingBoard :BuildingBoard, in visitedCells :boolean[][], in infectedComponents :Set<StorageComponent>) :void
- + epidemicResult (in player :Player) :void

Projectile
- size :ProjectileSize
- from :int
- + getSize () :ProjectileSize
- + getFrom () :int

ProjectileSize «enumeration»
- Big
- Small

BatteryStorage
- capacity :int
- itemsCount :int
- + getCapacity () :int
- + getItemsCount () :int
- + incrementItemsCount (in num :int) :boolean
- + decrementItemsCount (in num :int) :boolean

HousingUnit
- capacity :int
- crewmateCount :int
- hasOrangeAlien :boolean
- hasPurpleAlien :boolean
- + getCapacity () :int
- + getItemsCount () :int
- + incrementCrewmateCount (in num :int) :boolean
- + decrementCrewmateCount (in num :int) :boolean
- + getOrangeAlien () :boolean
- + getPurpleAlien () :boolean
- + setOrangeAlien (in orangeAlien :boolean) :void
- + setPurpleAlien (in purpleAlien :boolean) :void

BoxStorage
- boxStorage :Box[]
- isRed :boolean
- + addBox (in element :box, in idx :int) :boolean
- + removeBox (in num :int) :boolean
- + getBoxStorage () :Box[]
- + getIsRed () :boolean
- + getBoxCount () :int

Box
- type :BoxType
- value :int
- + getType () :BoxType
- + setType (in BoxType :enum)
- + getValue () :int

BoxType «enumeration»
- Red
- Yellow
- Green
- Blue