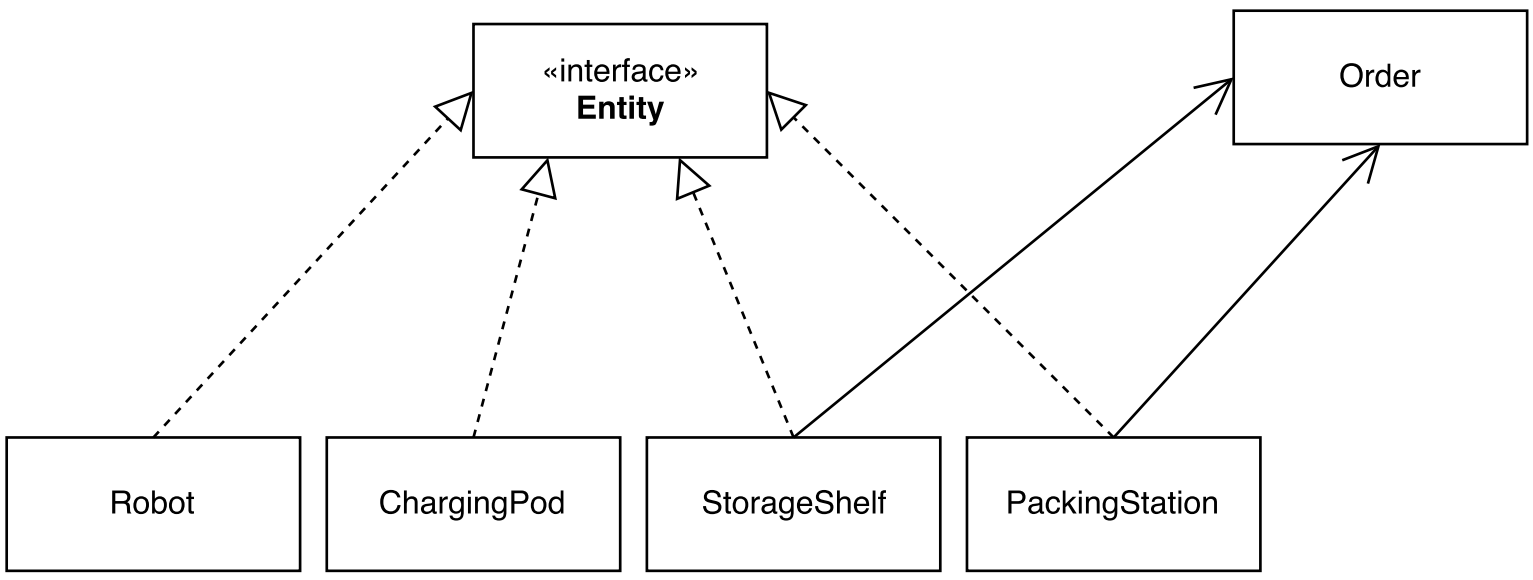
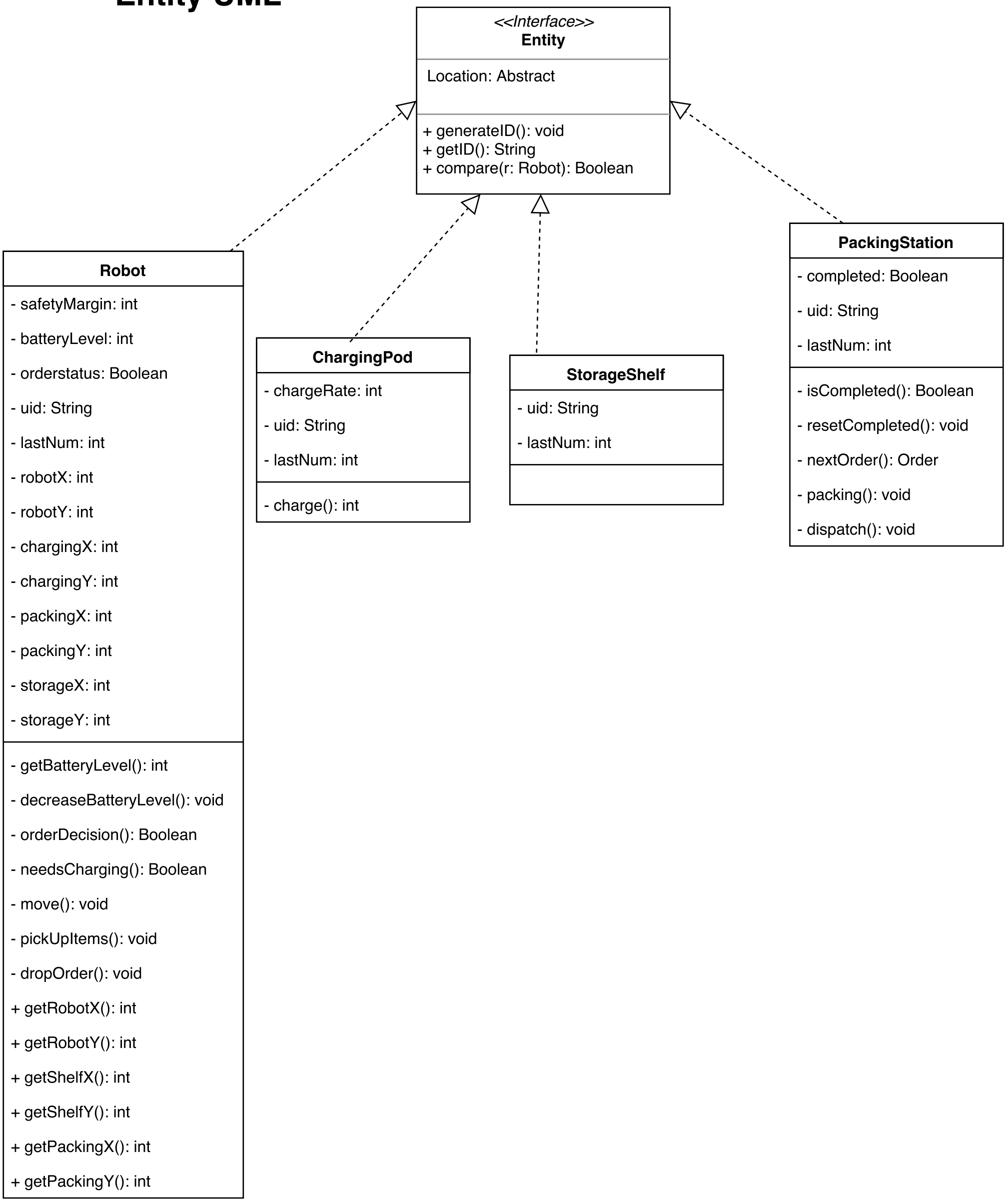
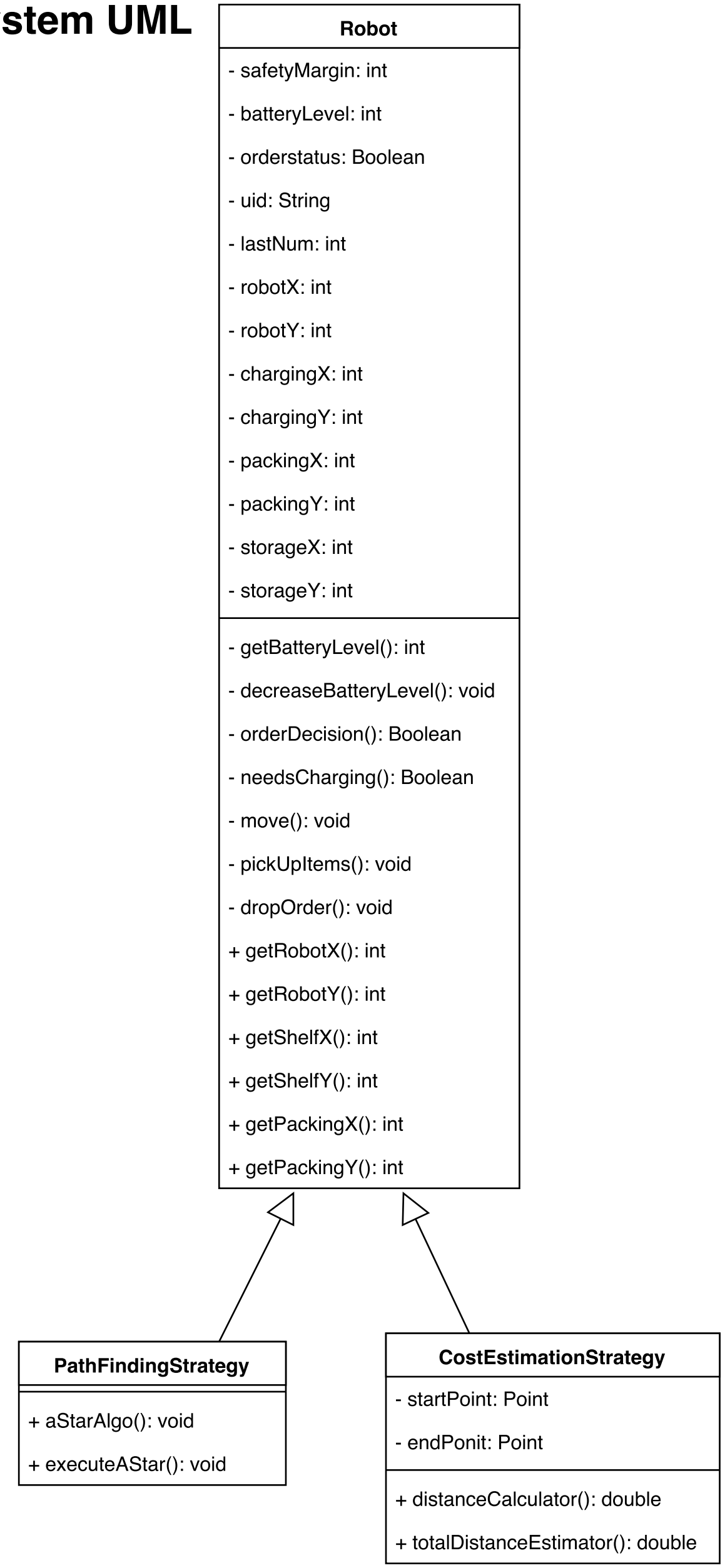
Overview

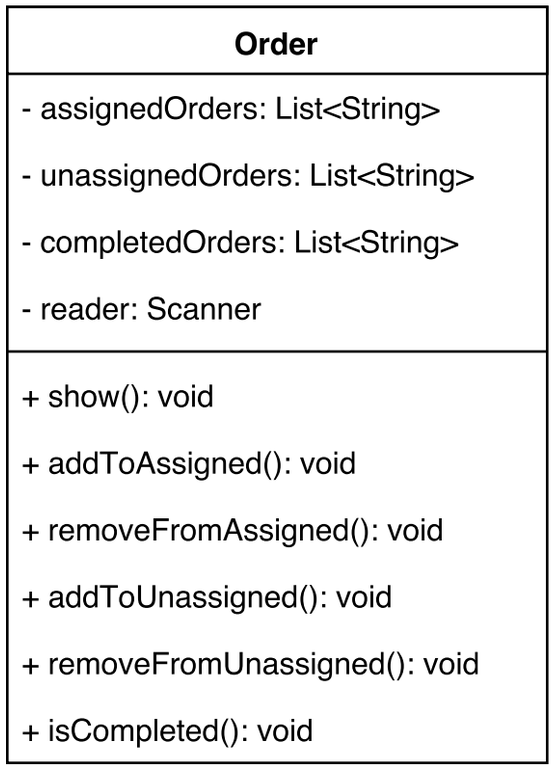

«interface»
Entity
Order
Robot
ChargingPod
StorageShelf
PackingStation

Entity UML


<<Interface>>
Entity
Location: Abstract
+ generateID(): void
+ getID(): String
+ compare(r: Robot): Boolean
Robot
- safetyMargin: int
- batteryLevel: int
- orderstatus: Boolean
- uid: String
- lastNum: int
- robotX: int
- robotY: int
- chargingX: int
- chargingY: int
- packingX: int
- packingY: int
- storageX: int
- storageY: int
- getBatteryLevel(): int
- decreaseBatteryLevel(): void
- orderDecision(): Boolean
- needsCharging(): Boolean
- move(): void
- pickUpItems(): void
- dropOrder(): void
+ getRobotX(): int
+ getRobotY(): int
+ getShelfX(): int
+ getShelfY(): int
+ getPackingX(): int
+ getPackingY(): int
ChargingPod
- chargeRate: int
- uid: String
- lastNum: int
- charge(): int
StorageShelf
- uid: String
- lastNum: int
PackingStation
- completed: Boolean
- uid: String
- lastNum: int
- isCompleted(): Boolean
- resetCompleted(): void
- nextOrder(): Order
- packing(): void
- dispatch(): void

Robot Subsystem UML




Order
- assignedOrders: List<String>
- unassignedOrders: List<String>
- completedOrders: List<String>
- reader: Scanner
+ show(): void
+ addToAssigned(): void
+ removeFromAssigned(): void
+ addToUnassigned(): void
+ removeFromUnassigned(): void
+ isCompleted(): void

UML Reasoning

Entity

This was made as an interface so that the methods are accessible in all classes required as well as their implementation according to their entities.

Robot

It was decided to have all functions of a robot into one class to increase cohesion and reduce coupling. The robot implements methods which are unique to its behaviour as well as getting the coordinates of each entity and enabling the path finding algorithm to be executed in its subclasses (which will be inherited). The Robot class will also contain fields, that will hold the coordinates of each entity. We deemed this appropriate as Robot is the only class that requires the coordinates of each entity, in order to carry out the A* algorithm. This will increase cohesion and also decrease coupling. This is subject to change, as once we fully understand the best way to implement the A* algorithm, the way in which the coordinates are stored may change.

ChargingPods/StorageShelf/PackingStation

These classes will contain the unique implementations of the methods from the Entity interface. They also contain their own unique methods, enabling it to carry out its own functionality. There have been no additional methods added to the StorageShelf class, as it will only need to implement the methods from the Entity class, to carry out its functionality.

PathFindingStrategy

PathFindingStrategy contains a method that will do the mathematical operations of the A* algorithm. This class will inherit the methods of the Robot class that get the X and Y coordinates of each entity. It will then use this information, to execute the A* Algorithm.

CostEstimationStrategy

Contains two fields enabling the distance to be calculated via the distanceCalculator method, and then estimate total distance using the totalDistanceEstimator.

Order

The order class will contain a scanner, that will be used to read the .sim file, that contains the information for each order. It has three List fields, which will divide up the orders into assigned, unassigned and completed. There will be a show method, that displays the order in the GUI. The class also contains methods, that adds orders to the respective lists, as well as remove orders from each list.