Goods:  
weight 重量

morningAllowed 是否早上允许送货

afternoonAllowed 是否下午允许送货

nightAllowed 是否晚上允许送货

priority 货物的优先级，这个优先级是针对一个时间段下的一个节点而言的。

可以分为五个等级，分别是1，2，3，4，5 ，5表示此时不能送货。

更新时间:

每次选择一个节点继续送货后，需要更新这个节点下货物的优先级。

每次一个时间段开始后，都需要更新全体节点下货物的优先级。

compareTo() 实现Comparable接口，用于排序。  
 先比较优先级，高的在前。再比较重量，大的在前。

Node:

name 节点名称

goodsList 货物列表

morningDelivery 该节点可以在早上最多配送的货物量

afternoonDelivery 该节点可以在下午最多配送的货物量

nightDelivery 该节点可以在晚上最多配送的货物量  
 这些信息在每次添加货物和删除货物时都需要更新

*GoodsBatchSelectStrategy* 从该节点选出货物组合的策略，分层0-1背包推选出合适的货物

sortGoodsList() 按优先级和重量，排序货物列表

排序时间:更新优先级之后便需要重新排序

updateGoodsPriority() 更新货物列表下所有货物的优先级

更新时间:

每次选择一个节点继续送货后，需要更新这个节点下货物的优先级。

每次一个时间段开始后，都需要更新全体节点下货物的优先级。

addGoods() 增加单个货物  
deleteGoods() 删除单个货物  
isEmpty() 判断是否还剩余货物

deleteGoodsBatch() 便捷性方法，删除被取走的一组货物

getPriority() 对已经sort过的goodsList取第一个货物的优先级

Graph:

loader 用于为地图加载信息

nodes 单层map，key是节点名称，value是Node对象

graphMap 两层map组成的图结构，key1是起点名称，key2是终点名称，value是值对象

getDistance() 获得两节点之间的距离

isFinished() 判断该地图中是否还有需要送的货物

Initialize() 初始化地图

*MapLoader*:

loadMap() 加载地图节点及距离信息

loadGoods() 加载地图货物信息

DeliveryVehicle:

deliveryMap 送货地图

maxWeight 小货车允许的最大容量

currentWeight 当前配重

currentDistance 当前距离

currentNode 当前所在节点，初始化后是起点

*NodeJumpStrategy* 实现小车节点跳转和货物扣除的策略，分为中继节点跳转和初始节点跳转。每次跳转都需要时间控制系统，控制小车避免超时。

中继节点: 效率高选择策略。

初始节点：效率低选择策略。

*TimeControlSystem* 在小车跳转过程中，提供时间干预

operate() 小车运作

*NodeJumpStrategy:*

*getPrioritizedNodes() 获取前五个(如果有的话)，考虑到节点优先级和高送货效率的潜力节点*

*getLastPrioritizedNodes() 获取前五个(如果有的话)，考虑到节点优先级和低送货效率的潜力节点*

*jump() 考虑时间因素的情况下，小车进行策略最优的下一跳*

*TimeControlSystem:*days配送天数，实际等同于小车数

currentPeriod 当前时间段

currentBudget 当前时间段剩余可行驶路程

aheadDelivery 是否允许提前配送

delayedReturn 是否允许滞后返回

enterNextPeriod() 进入下一个时间段