Documentation

Executive Summary

The automated warehouse system features robots that move around the warehouse to stock and deliver products between the shelves and delivery trucks.

This particular showcase divides the four robots into two groups: one group of two dedicated to fulfilling warehouse orders and another group dedicated to re-stocking the warehouse with goods. All robots move in a clock-wise circular flow-path in order to negate the possibility of any deadlock between the robots.

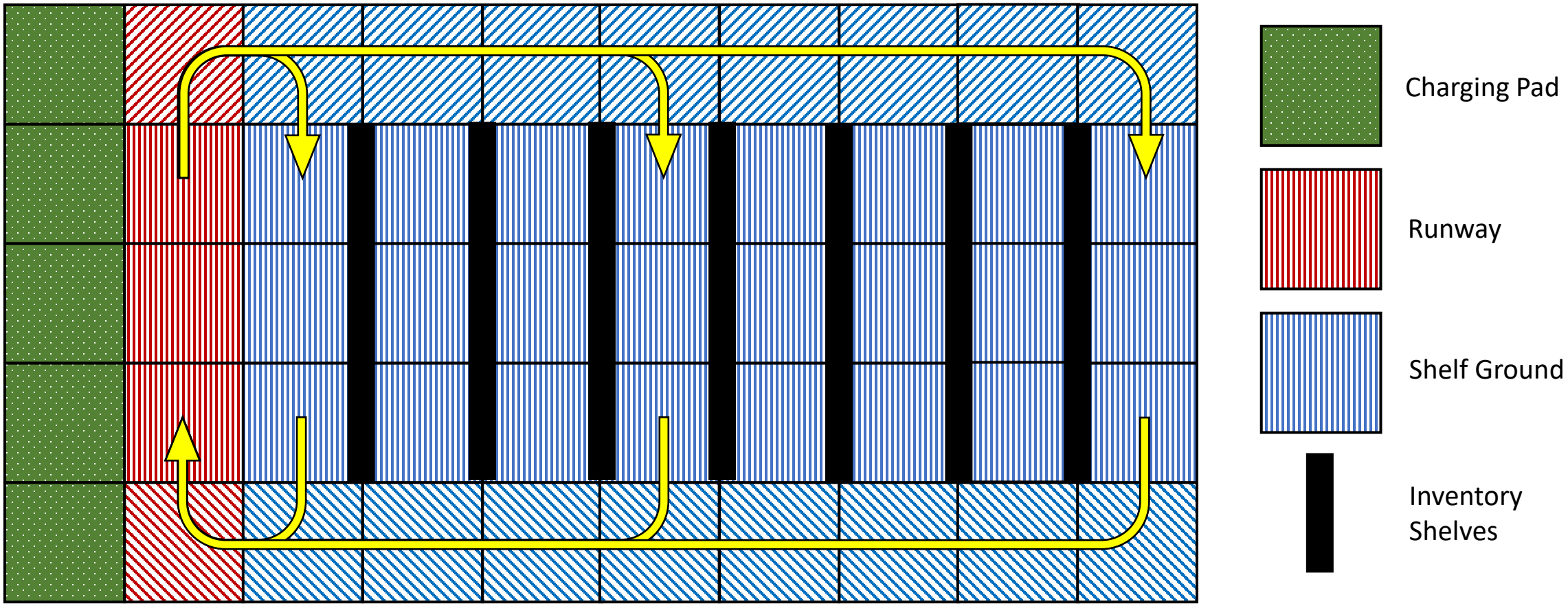
For this demonstration, there exist five types of items available for order, named items 1 to 5.

A central computer dedicates itself to deciding where the robots need to move and when in order to precisely fulfil each order correctly. The central computer, itself, is divided into two threads: one handling the partition of instructions for stocking the warehouse, another handling the steps necessary to fulfil each warehouse order.

Due to time constraints and technical limitations, this showcase will utilize the console's command window to demonstrate the features and capabilities of the program/system.

The program first initializes the data necessary to run the program. Because the warehouse is largely empty (or has any empty space), the program will automatically take stock of any empty space in the shelves as well as their positions and partition the robot instructions necessary to stock-up on inventory. Simultaneously, orders are completed one at a time and each order would be partitioned into smaller instructions for the robots to complete. When the truck is full, the warehouse experiences a delay which signifies one truck leaving and another truck coming in.

Warehouse Layout

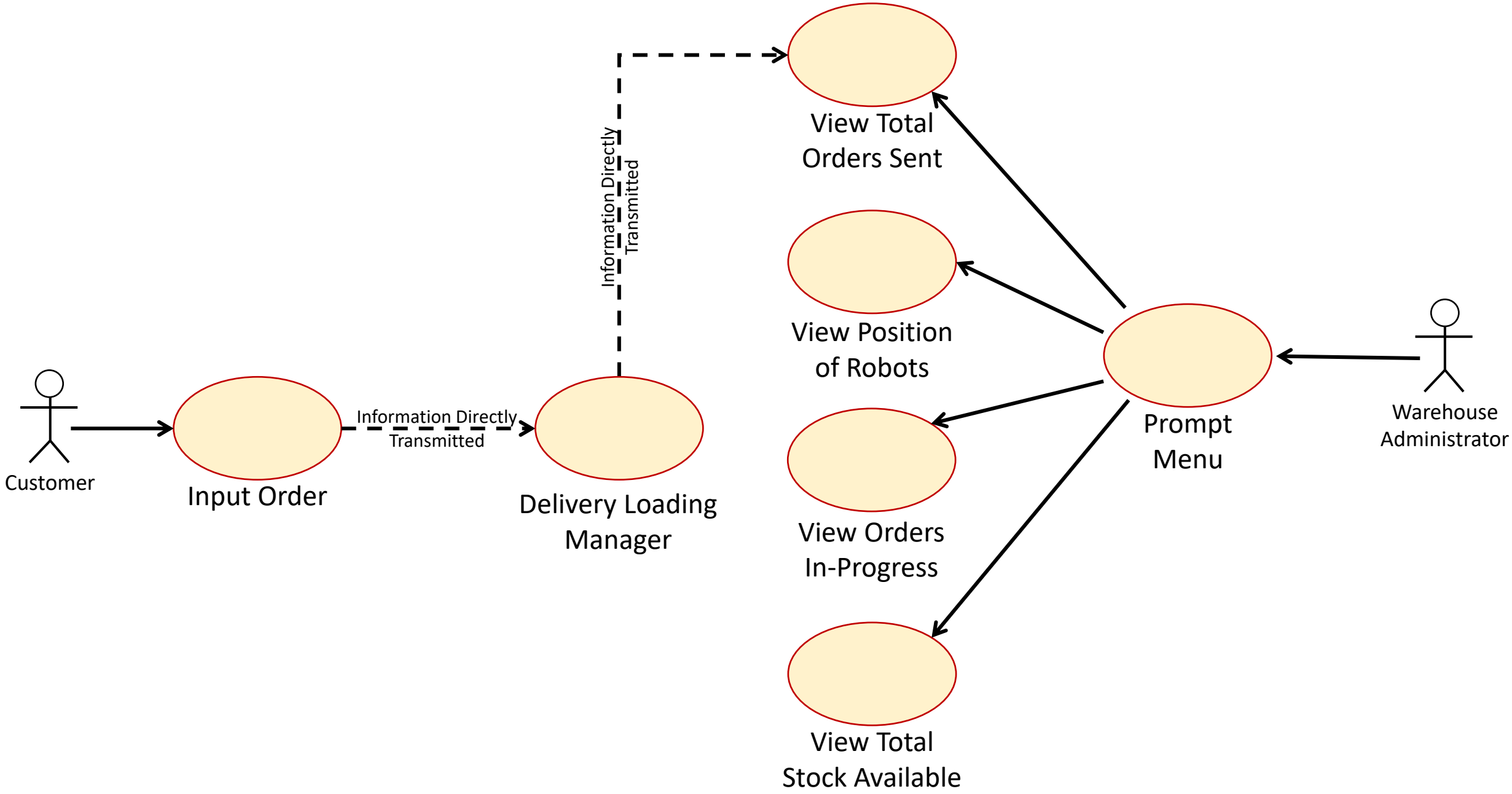


Charging Pad

Runway

Shelf Ground

Inventory Shelves


Customer
Input Order
Information Directly Transmitted
Delivery Loading Manager
Information Directly Transmitted
View Total Orders Sent
View Position of Robots
View Orders In-Progress
View Total Stock Available
Prompt Menu
Warehouse Administrator

Use-Case: Prompt Menu

Actor(s): Warehouse Administrator

Effects/Benefits:
Returns the program menu for various choices to be made.

1. Administrator opens program.
2. Administrator is prompted with menu of various functions.
3. If administrator wants to view the all the orders that have been fulfilled, then <<Scenario 1>>
4. If administrator wants to view all the orders that are in progress, then <<Scenario 2>>
5. If Administrator wants to view the layout of the warehouse and all the positions of the robots, then <<Scenario 3>>
6. If administrator wants to view the total stock of a specific item, then <<Scenario 4>>
7. Return to menu.
8. If administrator ends program: **end of Transaction**

Scenario 1: View Total Orders Sent
Scenario 2: View Orders In-Progress
Scenario 3: View Layout of Warehouse
Scenario 4: Total Stock of Warehouse

Use-Case: View Total Orders Sent

Actor(s): Warehouse Administrator

Effects/Benefits:
Returns the orders that have been sent out, the time-stamp of the completion, quantity of goods, time required to fulfil delivery.

1. Administrator is prompted with program menu.
2. Administrator enters command to view the orders that have been fulfilled at the warehouse.
3. Program returns information to administrator.
4. Administrator returns to main menu.

Use-Case: Input Order

Actor(s): Customer(s)

Effects/Benefits:
Customer inputs request for goods to be gathered and delivered from the warehouse.

1. Customer requests to input order.
2. Program returns query for customer to input quantity for each product.
3. Customer inputs quantity for each item.
4. Program returns a summary of the order, including the time required to deliver order and the total weight quantity of the order.
5. Order is relayed to the main queue or orders that need to be fulfilled.
6. Program returns to main menu.

Use-Case: View Orders In-Progress

Actor(s): Warehouse Administrator

Effects/Benefits:
Returns the orders that need to be fulfilled, the time needed to deliver goods, and their total quantity.

1. Administrator is prompted with program menu.
2. Administrator enters command to view the orders that need to be fulfilled at the warehouse.
3. Program returns information to administrator
4. Administrator returns to main menu.

Use-Case: View Layout of Warehouse

Actor(s): Warehouse Administrator

Effects/Benefits:
Generates an instantaneous view of the position of robots in warehouse

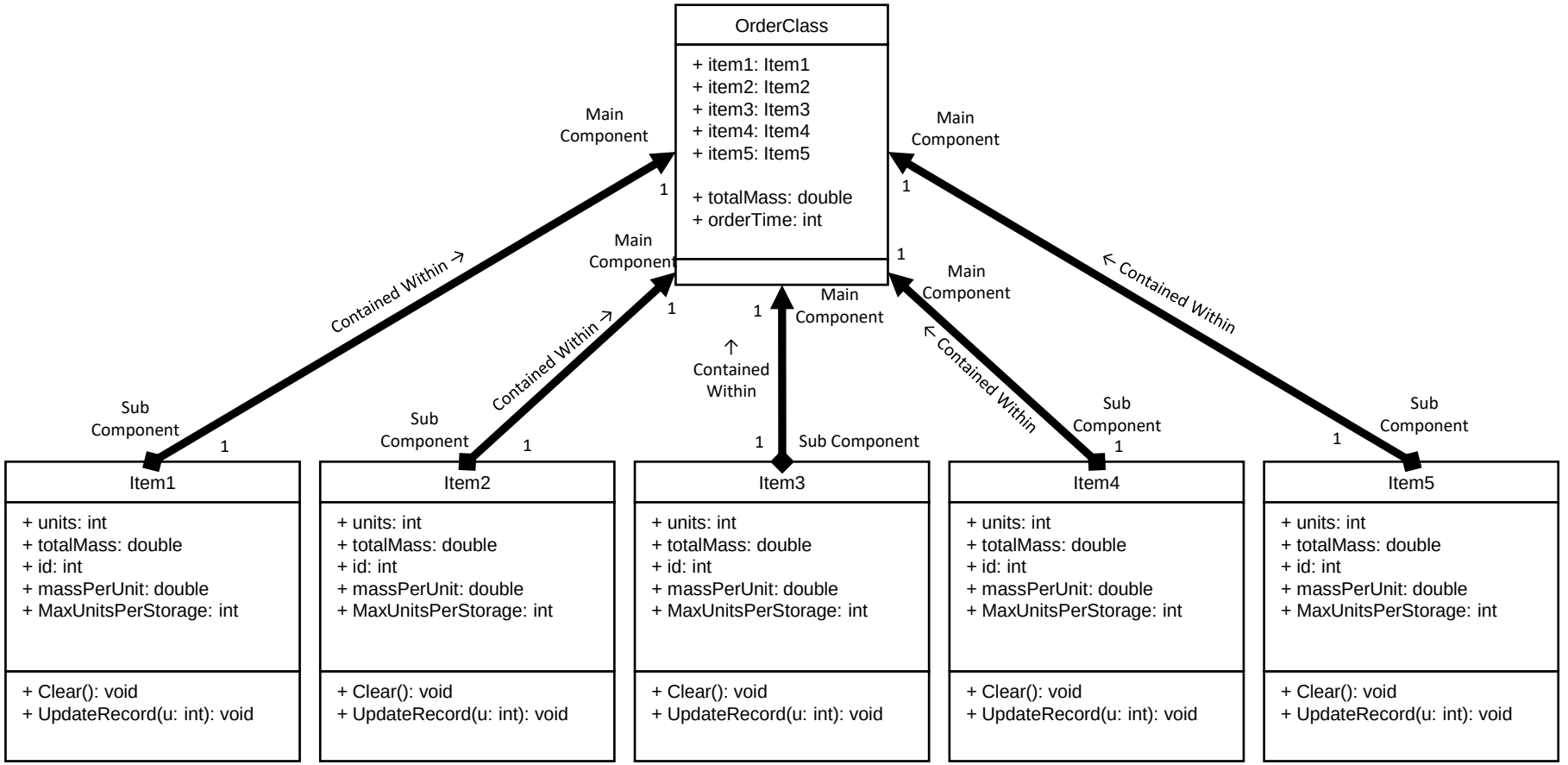
1. Administrator is prompted with program menu.
2. Administrator enters command to view the layout of the warehouse as well as the position of all the robots.
3. Program returns information to administrator
4. Administrator returns to main menu.

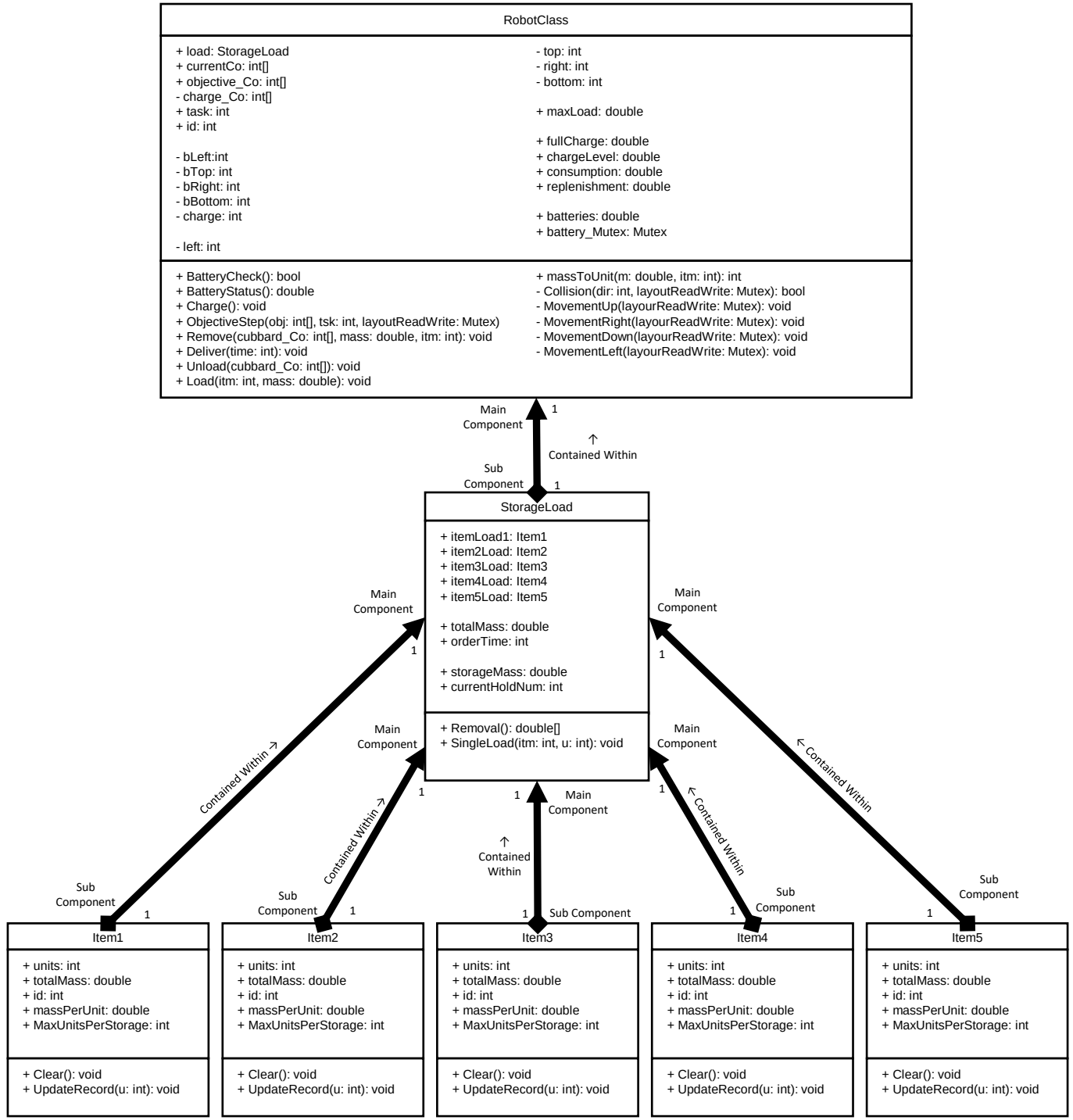
Use-Case: View Total Stock Available

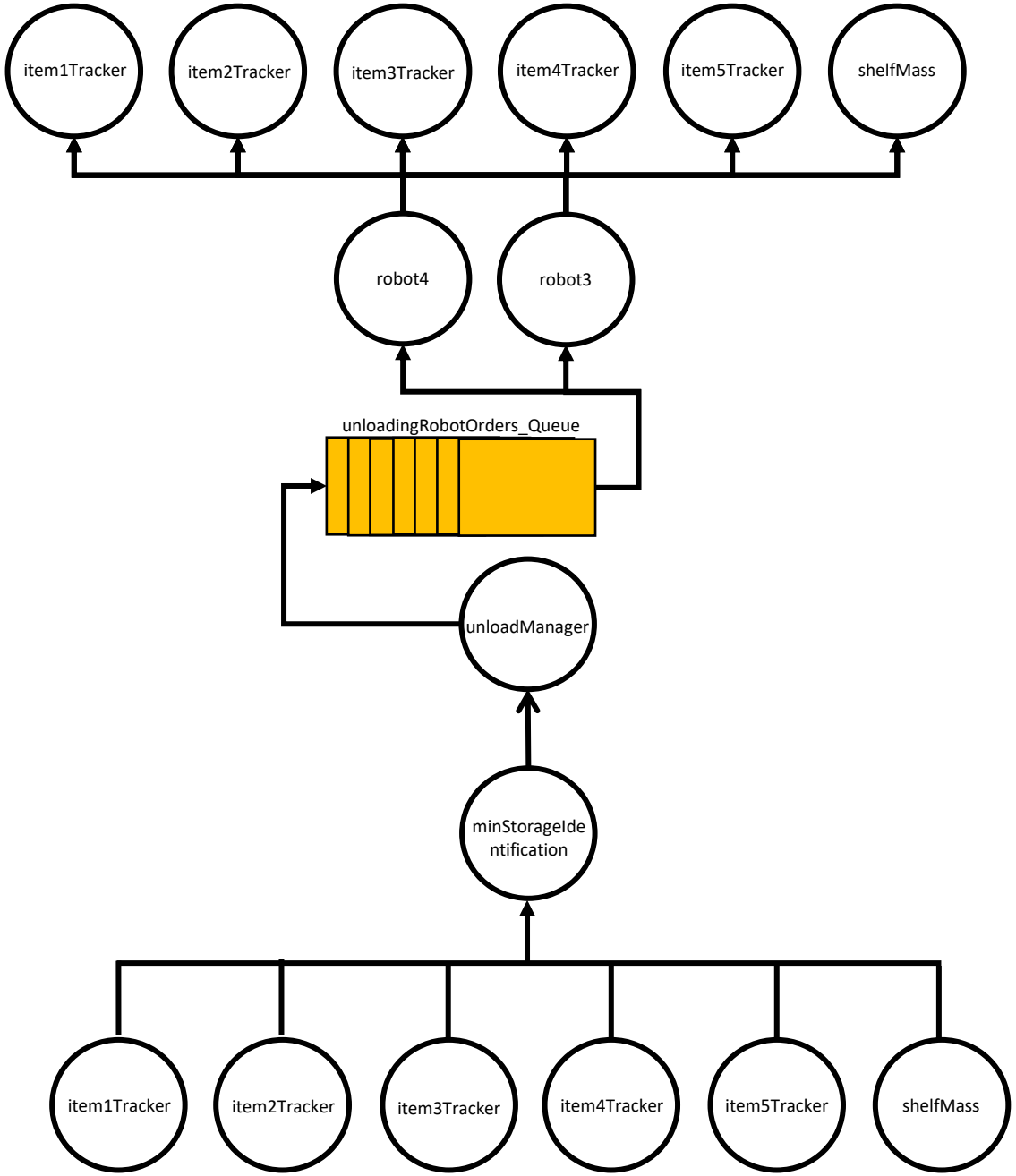
Actor(s): Warehouse Administrator

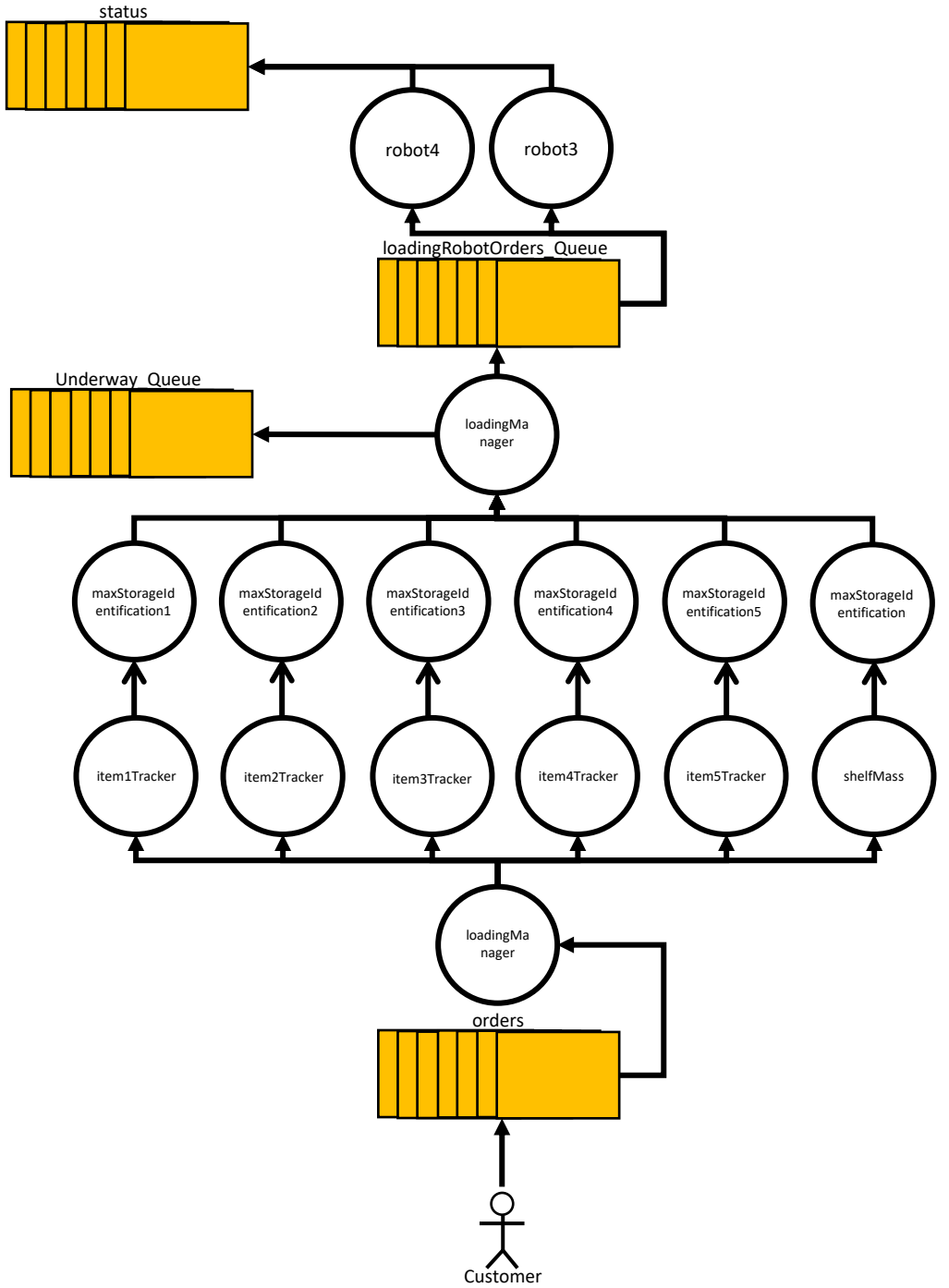
Effects/Benefits:
Returns the total quantity of a specific stock on any shelf that contains the item.

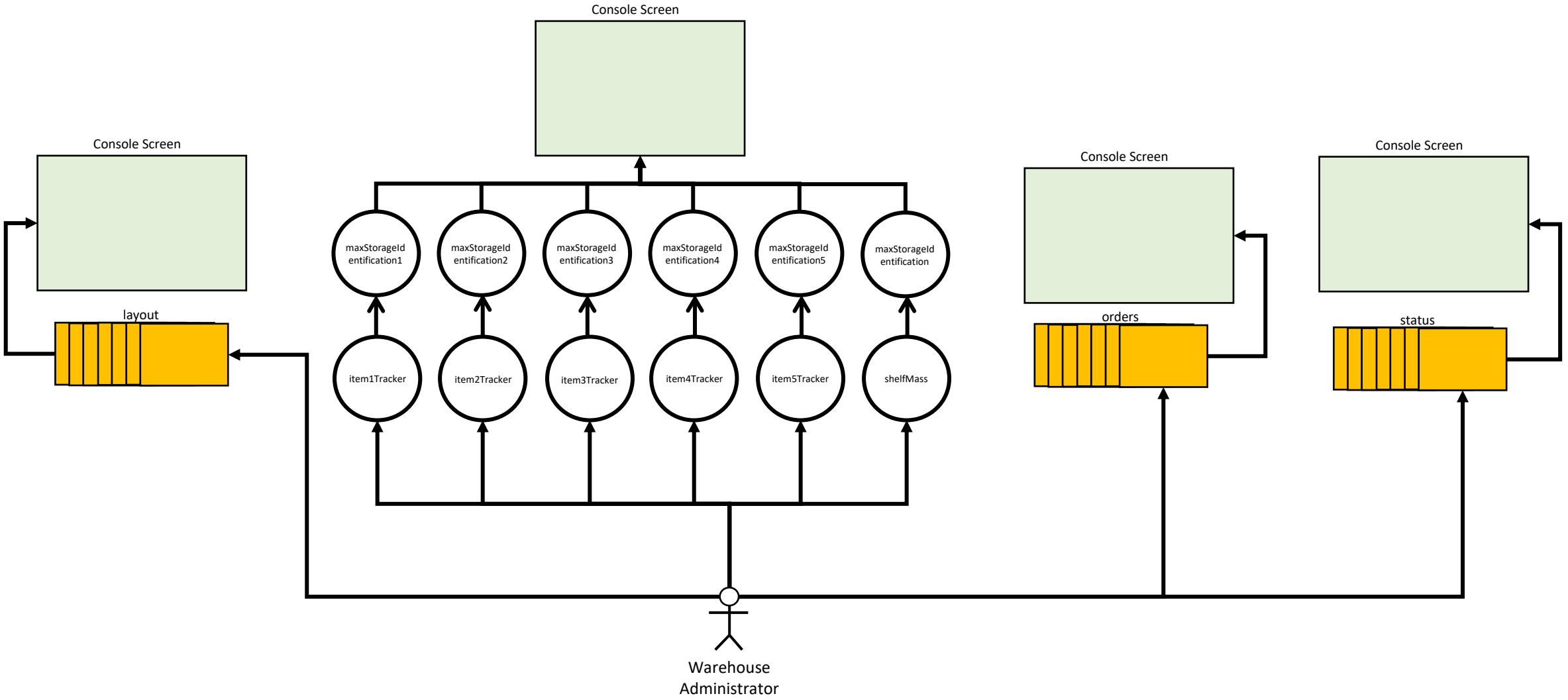
1. Administrator is prompted with program menu.
2. Administrator enters command specifying the quantity of each stock that they need to see.
3. Program returns information to administrator.
4. Administrator returns to main menu.

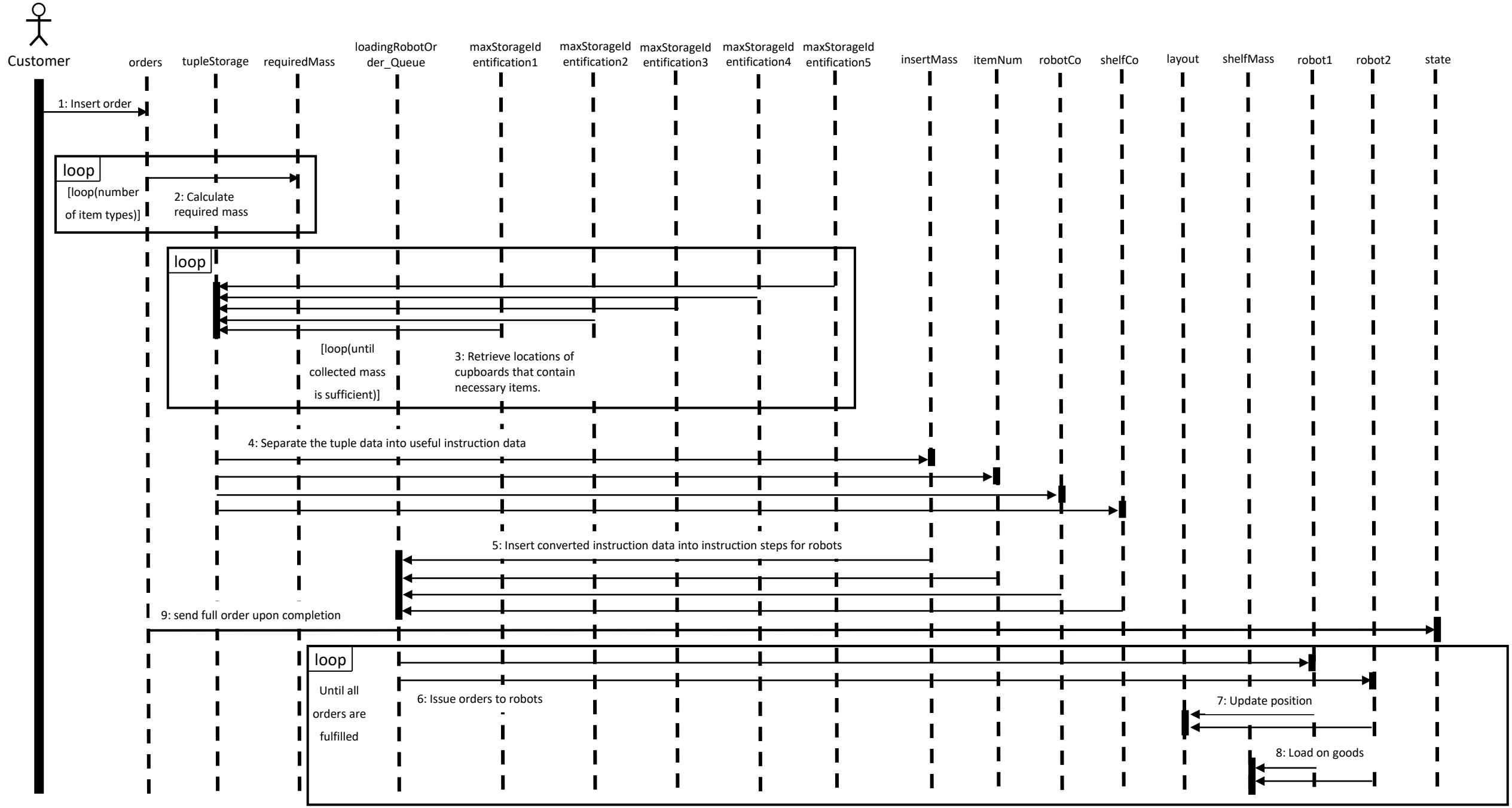

OrderClass
+ item1: Item1
+ item2: Item2
+ item3: Item3
+ item4: Item4
+ item5: Item5
+ totalMass: double
+ orderTime: int
Main Component
1
Contained Within
Sub Component
Item1
+ units: int
+ totalMass: double
+ id: int
+ massPerUnit: double
+ MaxUnitsPerStorage: int
+ Clear(): void
+ UpdateRecord(u: int): void
Item2
Item3
Item4
Item5


RobotClass
+ load: StorageLoad
+ currentCo: int[]
+ objective_Co: int[]
- charge_Co: int[]
+ task: int
+ id: int
- bLeft:int
- bTop: int
- bRight: int
- bBottom: int
- charge: int
- left: int
- top: int
- right: int
- bottom: int
+ maxLoad: double
+ fullCharge: double
+ chargeLevel: double
+ consumption: double
+ replenishment: double
+ batteries: double
+ battery_Mutex: Mutex
+ BatteryCheck(): bool
+ BatteryStatus(): double
+ Charge(): void
+ ObjectiveStep(obj: int[], tsk: int, layoutReadWrite: Mutex)
+ Remove(cubbard_Co: int[], mass: double, itm: int): void
+ Deliver(time: int): void
+ Unload(cubbard_Co: int[]): void
+ Load(itm: int, mass: double): void
+ massToUnit(m: double, itm: int): int
- Collision(dir: int, layoutReadWrite: Mutex): bool
- MovementUp(layourReadWrite: Mutex): void
- MovementRight(layourReadWrite: Mutex): void
- MovementDown(layourReadWrite: Mutex): void
- MovementLeft(layourReadWrite: Mutex): void
Main Component
Sub Component
Contained Within
StorageLoad
+ itemLoad1: Item1
+ item2Load: Item2
+ item3Load: Item3
+ item4Load: Item4
+ item5Load: Item5
+ totalMass: double
+ orderTime: int
+ storageMass: double
+ currentHoldNum: int
+ Removal(): double[]
+ SingleLoad(itm: int, u: int): void
Item1
Item2
Item3
Item4
Item5
+ units: int
+ totalMass: double
+ id: int
+ massPerUnit: double
+ MaxUnitsPerStorage: int
+ Clear(): void
+ UpdateRecord(u: int): void


item1Tracker
item2Tracker
item3Tracker
item4Tracker
item5Tracker
shelfMass
robot4
robot3
unloadingRobotOrders_Queue
unloadManager
minStorageIde
ntification
item1Tracker
item2Tracker
item3Tracker
item4Tracker
item5Tracker
shelfMass


status
robot4
robot3
loadingRobotOrders_Queue
Underway_Queue
loadingMa
nager
maxStorageId
entification1
maxStorageId
entification2
maxStorageId
entification3
maxStorageId
entification4
maxStorageId
entification5
maxStorageId
entification
item1Tracker
item2Tracker
item3Tracker
item4Tracker
item5Tracker
shelfMass
loadingMa
nager
orders
Customer


Console Screen
Console Screen
Console Screen
Console Screen
Console Screen
layout
orders
status
maxStorageId entification1
maxStorageId entification2
maxStorageId entification3
maxStorageId entification4
maxStorageId entification5
maxStorageId entification
item1Tracker
item2Tracker
item3Tracker
item4Tracker
item5Tracker
shelfMass
Warehouse Administrator


Customer
orders
tupleStorage
requiredMass
loadingRobotOrder_Queue
maxStorageIdentification1
maxStorageIdentification2
maxStorageIdentification3
maxStorageIdentification4
maxStorageIdentification5
insertMass
itemNum
robotCo
shelfCo
layout
shelfMass
robot1
robot2
state
1: Insert order
loop
[loop(number of item types)]
2: Calculate required mass
loop
[loop(until collected mass is sufficient)]
3: Retrieve locations of cupboards that contain necessary items.
4: Separate the tuple data into useful instruction data
5: Insert converted instruction data into instruction steps for robots
9: send full order upon completion
loop
Until all orders are fulfilled
6: Issue orders to robots
7: Update position
8: Load on goods