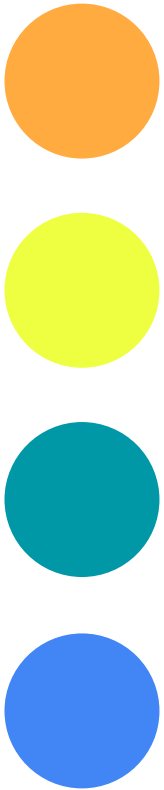
User Interface
Controller
Use Case
Entity

User Interface

ShellApplication

Responsibilities

- Allow the user to load a new warehouse and specify warehouse size (width and height)

Collaborators

Controller

WarehouseController

Responsibilities

- Create and store a warehouse
- Method `insertItem` to insert an Item into the warehouse into an available `Rack`
- Method `removeItem` to remove an Item from the warehouse via a `Depot`

Collaborators

- Warehouse
- Item

Use Case

Receivable

Interface

Responsibilities

- Interface specifying a contract for *receiving* items
- Abstract `receiveItem` method

Collaborators

- Item

Use Case

Distributable

Responsibilities

- Interface specifying a contract for *distributing* items
- Abstract `distributeItem` method

Collaborators

- Item

Use Case

Robot

Abstract Class

Responsibilities

Abstract class for implementing moving agents in the warehouse with arbitrary route-finders

- Method to move a given item from its source to a specified destination given an order using an arbitrary route-finder

Collaborators

- WarehouseController
- Pathfinder
- OrderQueue

Use Case

ObstacleAvoidanceRobot

Responsibilities

- A `Robot` with an obstacle avoidance strategy
- Method `clearPath` that uses existing `Pathfinder` algorithm while sidestepping obstacles as it encounters it by trying to wait for it, go around it, or recomputing the route

Collaborators

- WarehouseController
- Pathfinder
- Robot

Use Case

ItemScorer

Responsibilities

- `computeCost` method to compute the ‘cost’ of two items: that is, the distance between two items

Collaborators

- Item

Use Case

Pathfinder

Abstract Class

Responsibilities

- Create an instance of the Pathfinder class for pathfinders that discovers the “optimal” path for packages
- `findPath` method that returns the optimal path for packages using an ItemScorer

Collaborators

- Item
- Graph
- ItemScorer

Entity

Tile

Responsibilities

- Act as an empty tile in a warehouse grid
- Store it’s position

Collaborators

Entity

Graph

Responsibilities

- Create an instance of a graph class to use in the pathfinder algorithm
- Store graph nodes
- Store graph connections
- Method `getNode` to find the node corresponding to a given item ID
- Method to `getConnections` between nodes

Collaborators

- Item

Entity

Item

Responsibilities

- Create and store an `Item` in the inventory
- Store `Item` id, name, and description (create random ID if none given)
- Method to create a copy of this `Item`
- Getter for `Item` name
- Getter for `Item` description
- Getter for `Item` ID
- Setter for `Item` name
- Setter for `Item` description
- `toString` that returns string with labeled id, name and description
- `equals` returns whether this Item equals another
- Method `hashCode` returns has for id, name and description

Collaborators

Entity

InventoryCatalogue

Responsibilities

- Create and store an inventory of all Items in the warehouse
- Store a mapping of all Item id's to the Item objects
- `removeItem` method removes an item from the inventory (decreases quantity by 1 or, if quantity == 0, removes item ID key)
- `addItem` method adds an item to the inventory (increases quantity by 1 or, if the item ID is not already in the mapping, create a new key-value pair)
- The `getItemById` method returns the item with the given id

Collaborators

- Item

Entity

Warehouse

Responsibilities

- Create and store a 2D representation of a warehouse made up of `Tiles` with the given width and height
- Return the `Tile` at a given coordinate
- Return an empty `Tile` (Tile with no `StorageUnit`)
- Return an available `Rack` for the given `Item`
- Check whether a given tile coordinate is within the bounds of the warehouse layout

Collaborators

- Tile

Entity

OrderQueue

Responsibilities

- Create and store an `OrderQueue` that adds new requests for items to the back of the queue
- Method to remove an order from the front of the queue after being picked up by a `Robot`
- Method to add a new order to the queue

Collaborators

- Item
- Order

Entity

Order

Responsibilities

- Create and store an order that specifies an item to be moved, it's source, and it's destination

Collaborators

- Item

Entity

StorageUnit

Abstract Class

Responsibilities

- `StorageUnit` is an abstract class that inherits from `Distributable` and `Receivable` for a unit in the warehouse that can store items
- Abstract method `addItem` to add an item to the storage unit
- Abstract method `removeItem` to remove an item from the storage unit
- *Override* `receiveItem` from `Receivable` interface which calls on `addItem`
- *Override* `distributeItem` which calls on `removeItem`

Collaborators

- Item
- Distributable
- Receivable

Entity

Depot

Responsibilities

- Class that extends `StorageUnit`: creates and stores a storage unit with a given (or infinite) capacity in the warehouse that can store Item's of any type
- Store a mapping from Items to their quantity stored in this depot
- Store the depot's capacity
- Implement abstract `addItem` method from `StorageUnit` add an item if below capacity + return if this is successful
- Implement abstract `removeItem` method from StorageUnit remove an item from this `StorageUnit` if the item is in this storage unit and there is enough quantity
- Getter for `capacity`
- Method that returns if the depot has infinite capacity
- Method that returns the size of the depot (mapping size)

Collaborators

- StorageUnit
- Item

Entity

Rack

Responsibilities

- Create and store a storage unit (of a given or infinite capacity) in the warehouse that can store items of a single type
- Store the `Rack` capacity, the current item type stored in the track, and the size of the rack
- Getter for capacity
- Method that returns whether the `Rack` has infinite capacity
- *Override* `addItem` method from `StorageUnit` to add given item to the rack
- *Override* `removeItem` method from `StorageUnit` to remove given item from the rack

Collaborators

- StorageUnit
- Item