Brief

The task is to develop a simplified simulation of a part of an automated warehouse, such as those used by Amazon .

The simulation is run in "ticks": each tick, all the actors will perform a bit of their behaviour. By running one tick after the other, we can see how the warehouse would operate given some parameters. The simulation starts with a predefined list of orders, which may be read from a file (more details in Section 4) or be randomly generated. The simulation continues until it fails or all orders are dispatched, reporting the number of ticks and the final result.

- In these warehouses, the floor is divided into a grid of cells. Each cell may contain entities of various types, all with unique identifiers (UIDs):

- **Idle packing stations** (identified as P1, P2, ...) take up the next order from the list, and then ask the robots to bring items from certain storage shelves. Once a packing station has all the items, it takes a number of ticks packing it (which depends on the order), and then dispatches it for delivery.
- **A storage shelf** (identified as S1, S2, ...) has the items which can be ordered by customers. Robots take items from these shelves. In this simulation, we do not care about the specific items themselves: orders simply list the UIDs of the shelves that we should take items from. Shelves are mostly passive markers for the robots to go to.
- **A charging pod** (identified as C1, C2, ...) charges the batteries of a robot c power-units per tick. Each robot has its own charging pod: robots can go through other pods, but they only charge at their own pod.
- Once per tick, a **robot** (identified as R1, R2, ...) can move up/down/left/right once on the grid, or it can take items from the shelf it is currently at. They can move under storage shelves, charging pods and packing stations. If two robots end up in the same cell, they have crashed: after notifying the user, the simulation must be stopped.
    - Robots are battery-powered. A robot starts with B power units in its battery: to move 1 space, the robot uses 1 power unit when not carrying anything, and 2 power units otherwise. Robots do not use power if they are not moving. If a robot runs out of battery power, this must be reported back to the user and the simulation must be stopped.
    - When a robot reaches its designated charging pod, it continues charging until the battery is half full. Robots should also return to the charging pod if they detect that they will not be able to reach their next destination and return to the charging pod with their current battery level.
    - When a robot reaches a packing station, as a simplification, we assume that it picks up all the desired items on the same tick. On the next tick, it will be able to move again.
    - A packing station will ask a robot if it can "bring items from shelf X". The robot will decide if it wants to accept the assignment or not: this will depend on the current battery level and how far the shelf and the packing station are. If all robots reject the assignment, the packing station will retry on the next tick in case one of the robots becomes available.

You are free to make the system smarter in how it assigns missions to robots, or how the robots deal with them. Extra marks may be given in those cases. Be sure you explain your strategy and how you have tested it!

You should provide some representation of the warehouse floor and the main metrics of the simulation. A text representation can work, but to achieve full marks you will need a graphical representation in JavaFX.

There should be a way to watch the simulation step by step. This will help with debugging and it is needed to mark your submission.

Noun/Verb Analysis

From this brief, we can extract all nouns and verbs so that we can narrow down what we need to do.

Nouns	Verbs
Actor	Perform
Amazon	-
Assignment	*
Available	*
Battery Level	*
Cell	*
Charging Pod	Charge
Customer	Order
Destination	*
File	*
Final Result	*
Floor	*
Grid of Cells	*
Item	*
Marker	*
Order	Random Generation, Read
Packing Station	Ask, Dispatch, Idle, Pack, Receive, Retry
Parameters	*
Power Units	*
Robot	Accept, Charge, Charge Pod, Crash, Decide, Detect Power, Move, Notify, Reject, Return, Start, Take
Simulation	Continue, Dispatch, Fail, Report, Run, Start
Storage Shelf	Has
Tick	Run
Unique Identifier (UID)	*
User	*
Warehouse	Operate

We want to ignore all references to the system itself. Here this relates to the ‘Simulation’ noun. This has been left in for reference of the overarching purpose of our program, but will not be seen further into this Noun/Adjective identification process.

We must remove any miscellaneous terms. These typically are adjectives, however there are some nouns that are unnecessary here.

- Removed Nouns
    - Amazon
    - Customer
    - Marker
    - User
- Removed Adjectives
    - Perform (Actor)
    - Decide (Robot)
    - Start (Robot)
    - Has (Storage Shelf)
    - Run (Tick) – This adjective is more applicable to the Simulation noun.

We will also remove

- Parameters, as this is a general term.
- Assignment, as this is another term for the purpose and Task of the Order, which will be contained in Order.

This brings our Noun/Adjective table to:

Nouns	Verbs
Actor	-
Available	*
Battery Level	*
Cell	*
Charging Pod	Charge
Destination	*
File	*
Final Result	*
Floor	*
Grid of Cells	*
Item	*
Order	Random Generation, Read
Packing Station	Ask, Dispatch, Idle, Pack, Receive, Retry
Power Units	*
Robot	Accept, Charge, Charge Pod, Crash, Detect Power, Move, Notify, Reject, Return, Take
Storage Shelf	-
Tick	*
Unique Identifier (UID)	*
Warehouse	Operate

Classes Vs Attributes

Classes	
Actor	
Cell	
Charging Pod	
Floor	
Grid of Cells	
Order	
Packing Station	

Robot	
Storage Shelf	
Warehouse	

Actor -> Generic Superclass for all permeant objects on the field (i.e. Packing Stations, Charging Pods, Robots and Storage Shelves). Probably an interface.
Cell -> A singular location within the grid.