Advanced Modeling for Operations 23/24

ASSIGNMENT - PART 1

POLITECNICO
MILANO 1863

DIPARTIMENTO DI
INGEGNERIA GESTIONALE

Course organisation
Assessment

The **exam** includes**:**

- **Group assignment**  **70%** of the final mark
- **Individual written test**  **30%** of the final mark

- ***To pass the course**, students **must pass** (grade ≥ 18) **both the group assignment and the individual written test***
- *The **assignment grade** will be **valid for one academic year** (i.e., till September 2024)*
- *After the delivery of the assignment, **each student will be asked** to fill in a MS Forms to communicate **whether all the group members have actively participated and a penalty can be assigned***

Assignment Timeline

September

October

November

December

January

Assignment – Part 1

Lessons dedicated to the assignment:
October 23th and November 13th

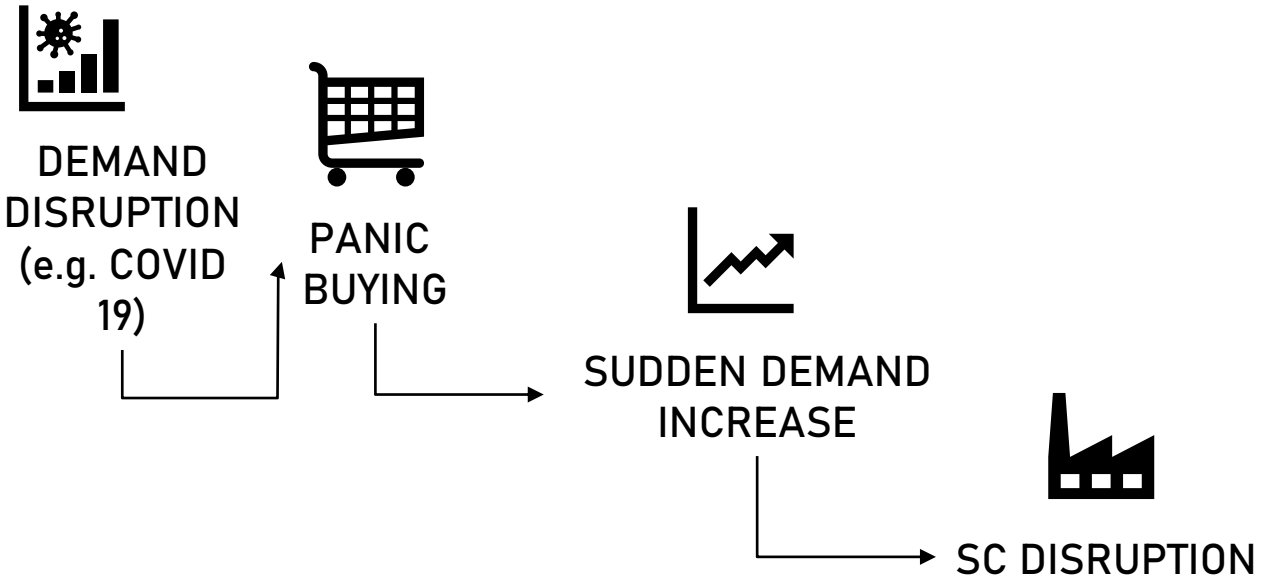
Assignment – Part 2

Lesson dedicated to the assignment:
December 4th and 18th

Same operations setting

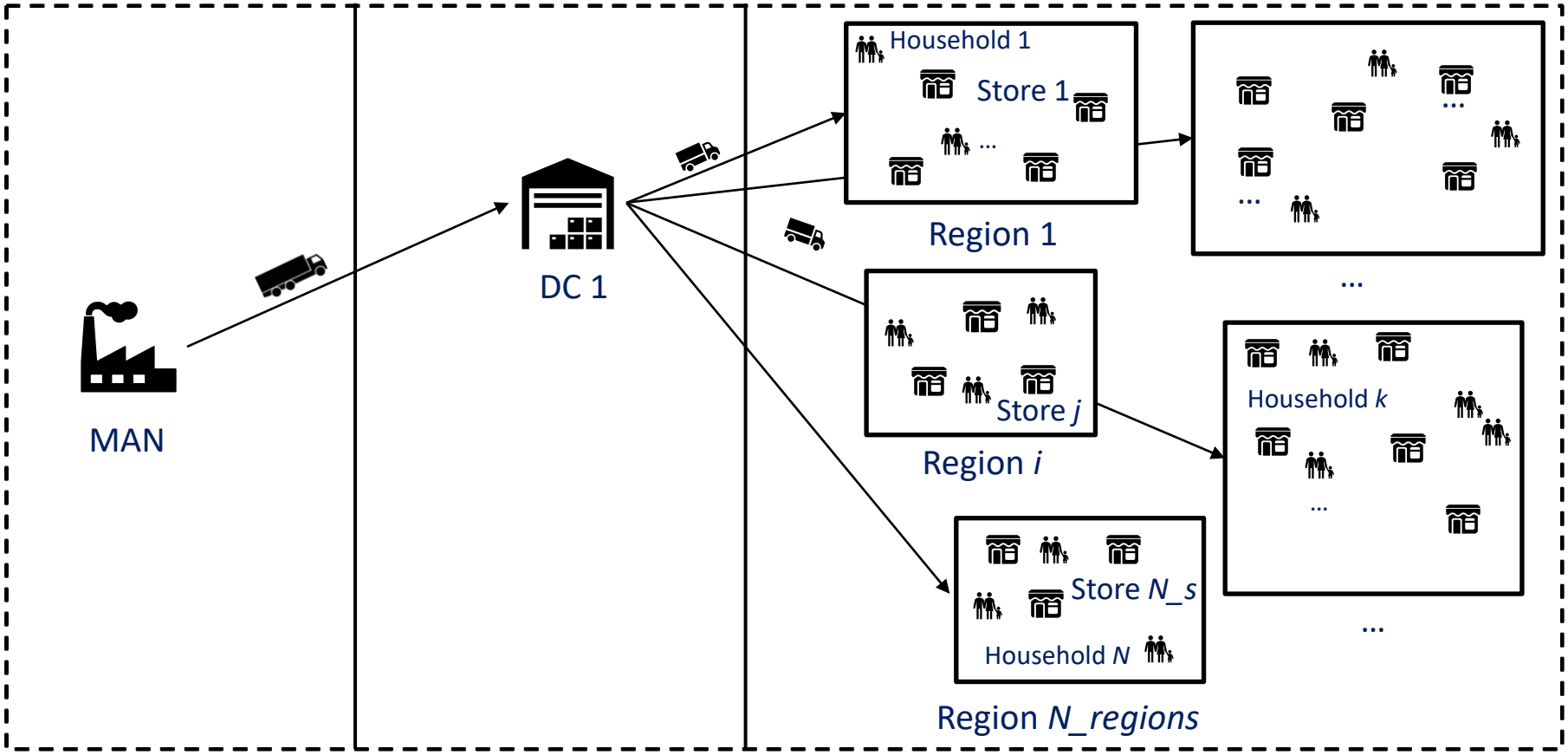
Deadline (for both Part 1 and Part 2):
January 8th at 23.59

Assignment
Problem description: objective



- **Objective:** investigate the **response** of **grocery supply chains to consumer panic buying** triggered by a large-scale disaster
- **Focus on a single product:** pasta

Assignment
Problem description: supply chain network

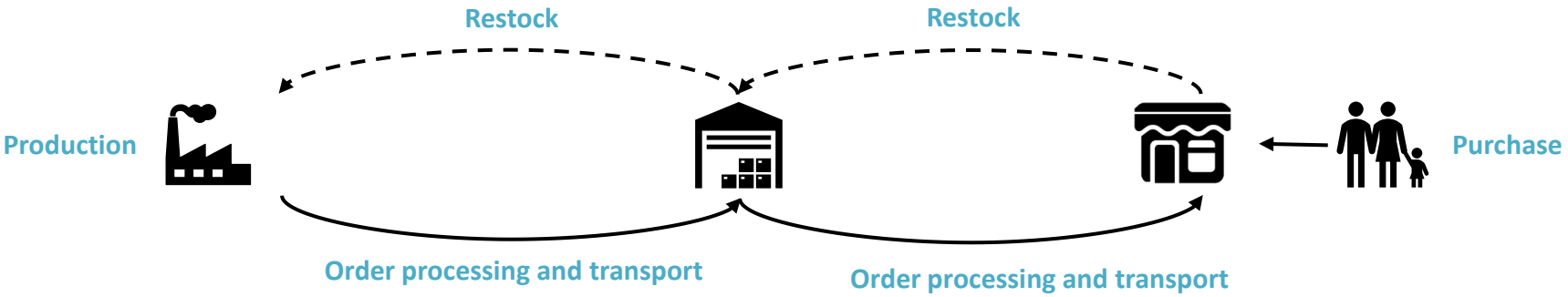


MANUFACTURER
RETAILER'S DISTRIBUTION CENTER
STORE
REGION
HOUSEHOLD

Assignment
Problem description: inventory management

- **MTS** production process **and PULL LOGIC**
- **Periodic review** inventory control **policy at the distribution center and stores**
- **No safety stocks** at the distribution center and stores
- The **plant's production capacity** is **always able to satisfy the distribution center demand**

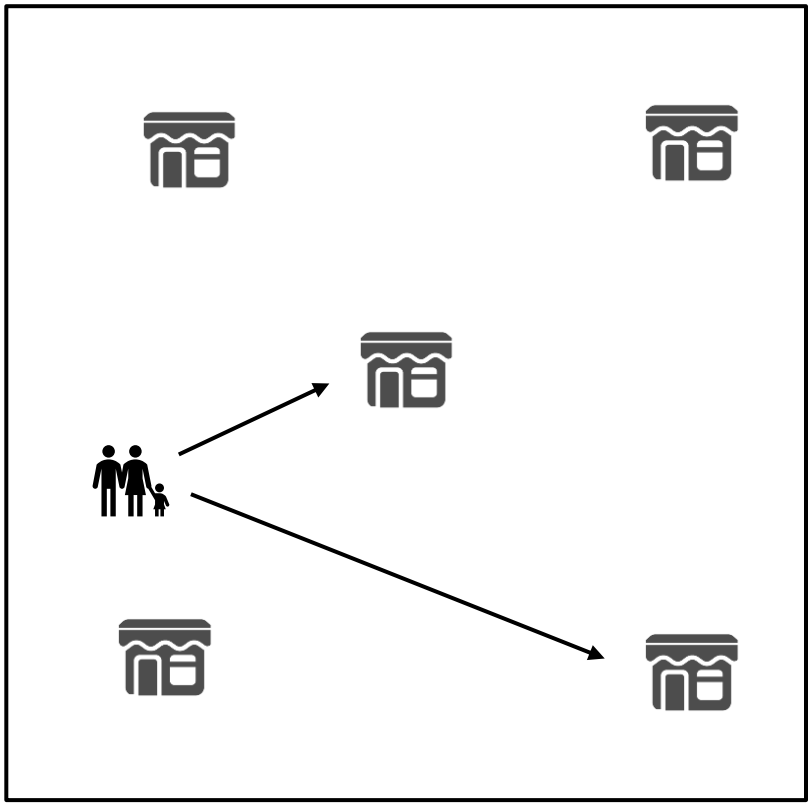
At the end of every fixed period the store submit a new order request



- **Request for restock every 3 days** (f_restock = 1/3)
- Manufacturer-to-DC **transport time: 2 days**
- **Order quantity: expected daily demand * 1/restock frequency**
- The **daily demand** is **estimated by looking** at the **restock period** (i.e. 3 days)

- **Request for restock every 3 days** (f_restock = 1/3)
- DC-to-store **transport time: 1 day**
- **Order quantity: expected daily demand * 1/restock frequency**
- The **daily demand** is **estimated by looking** at the **last 7 days**

Assignment
Problem description: purchasing process

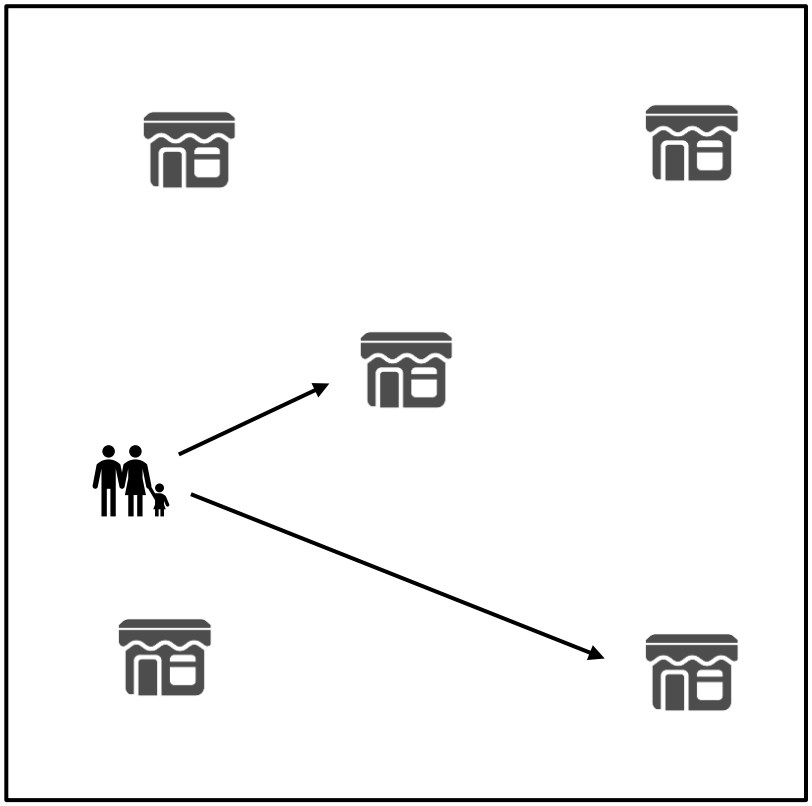


- A **household** (*customer*) purchase a **quantity of pasta *q*** with a given **frequency**

PURCHASING FREQUENCY (*f*)	PERCENTAGE OF HOUSEHOLDS	*q* (UNITS of 500 g)
Once every three days, i.e. $f = 0.333$	27%	1
Once per week, i.e. $f = 0.143$	61%	3
Once every 15 days, i.e. $f = 0.067$	10%	6
Once per month, i.e. $f = 0.033$	2%	11

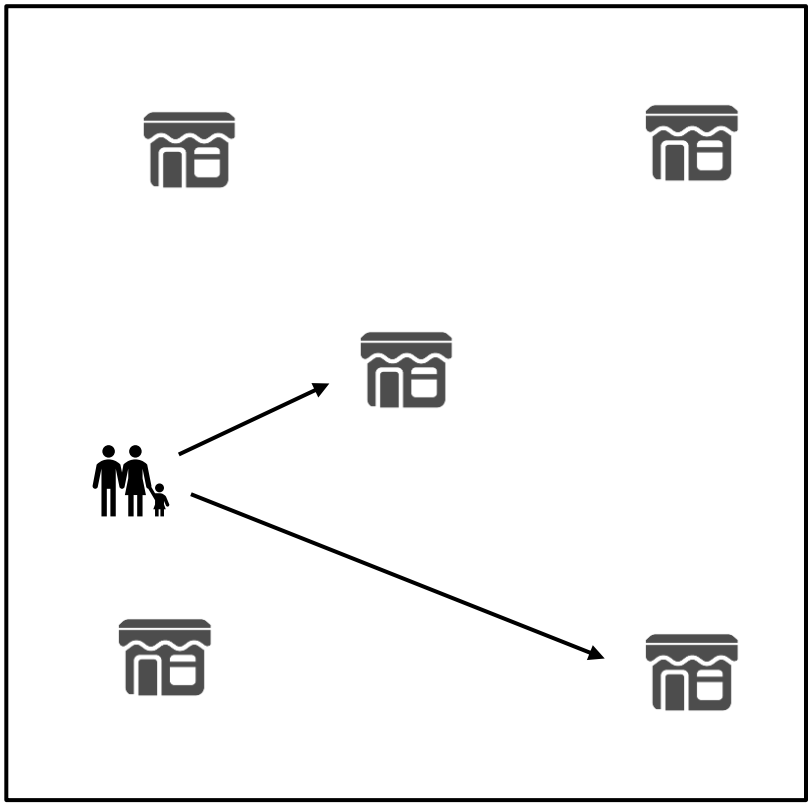
N.B.: the calculation of this distribution was performed starting from the average monthly expense of pasta equal to 11,78€ (Istat) per household, the average price per kg of pasta (2,14€/kg) (Food Report), and the frequency with which an Italian consumer goes to the supermarket (Censis-Coldiretti)

Assignment
Problem description: purchasing process

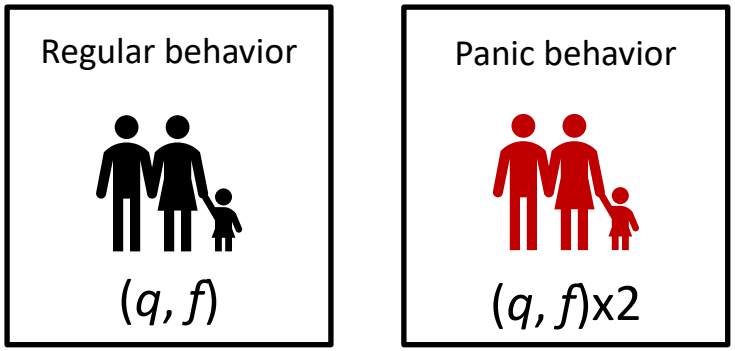


- A **household** (*customer*) purchase a **quantity of pasta *q*** with a given **frequency**
- The customer **randomly selects a store** when making a purchase
- **If** the selected **store does not have the entire** customer purchasing **quantity**, the customer purchases an amount equal to the **remaining inventory** at the store
- **In case** the store is in **stock out**, the **customer randomly selects another store** to visit (possibly the customer can visit all the stores)

Assignment
Problem description: panic buying

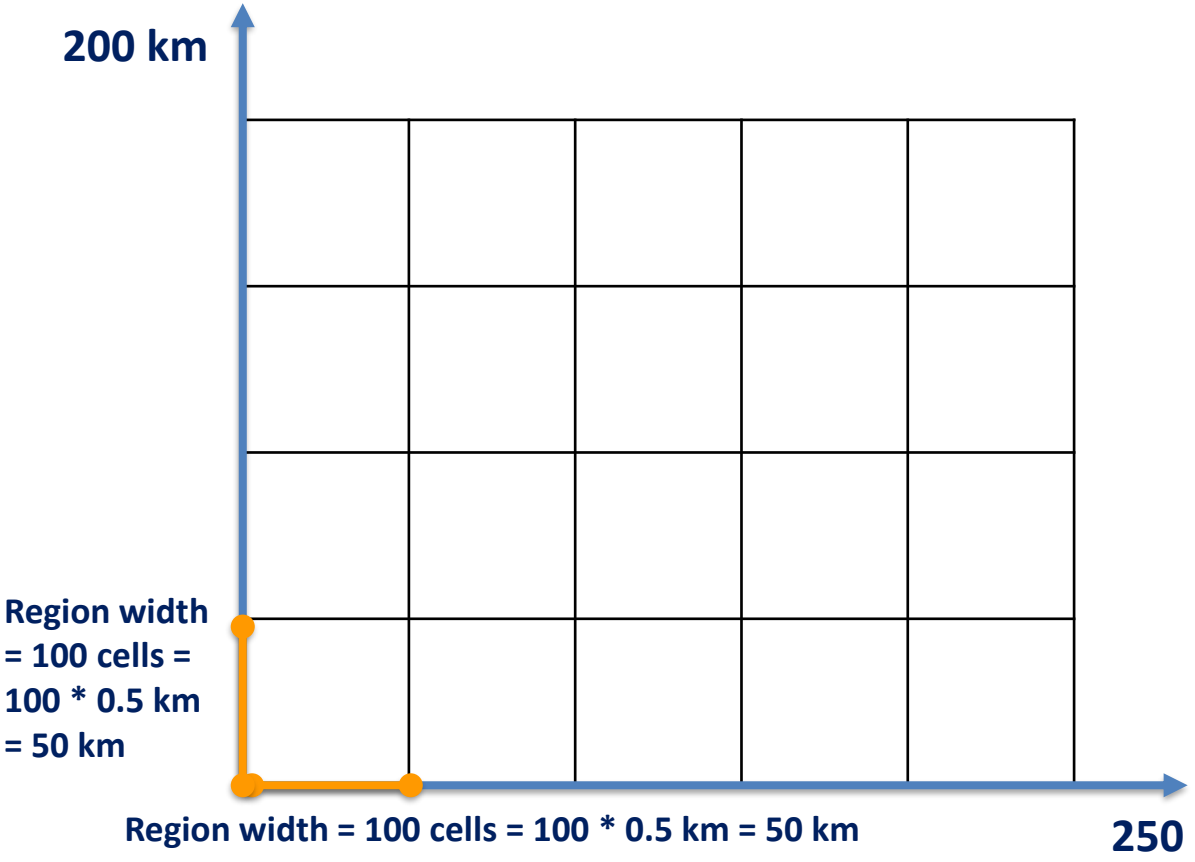


- At **day 15**, the **pandemic starts**
- At each step (day), the **percentage of customers in panic** is **randomly** selected **in** the range **[0.02; 0.04]**
- The customers with a **panic behavior**, double their **purchasing frequency and quantity** (*p_mult* = 2)



- The **pandemic** has a defined **duration** (i.e. **20 days**)

Assignment
Geographical representation: grid

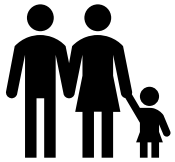


20 regions of 100 x 100 cells
(N_regions = 20)

100 stores randomly distributed on the grid
(N_s = 100)

30.000 households (customers) randomly distributed on the grid
(N = 30.000)

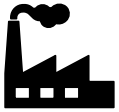
Assignment
Agents



Households → **class Customer**

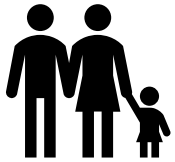
Stores → **class Store**

Distribution center → **class DistributionCenter**



Manufacturer → **class Manufacturer**

Assignment
Agents: attributes and methods

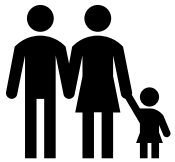


Class Customer

ATTRIBUTES

- Regular purchasing quantity [units] → ***q_base***
- Regular purchasing frequency [/day] → ***f_base***
- Store in which the customer makes a purchase → ***purchasing_store***
- Current purchasing quantity [units] → ***q***
- Current purchasing frequency [/day] → ***f***
- List of all the stores randomly ordered → ***stores***
- Boolean attribute indicating if the customer is in panic → ***pbs***

Assignment
Agents: attributes and methods



Class Customer

METHODS

At the first step, the **average daily demand for a store is estimated** → **store_visiting_list_generation()**

The **current purchasing quantity and frequency** are **set** according to the customer's status (panic or regular behavior) → **initial_panic()**

The **customer visits** the **store(s) and** makes the **purchase** (if one of the visited store has at least one item), **and** the **store's inventory and demand** are **updated** → **purchase()**

At the first step, the **average daily demand for a store is estimated**; then, at a generic step, the **panic or regular** purchasing **behavior** is regulated, **and, if the customer makes the purchasing process according to his purchasing frequency** → **step()**

Assignment
Agents: attributes and methods

Class Store

ATTRIBUTES

- Expected demand [units] → ***store_expected_demand***
- Current inventory [units] → ***store_inv***
- List of the demand at the store in the last 7 days → ***d_s***

Assignment
Agents: attributes and methods

METHODS

Class Store

The store **calculates** the **amount to be ordered and sends** a **restock request** to the DC → **store_request_restock()**

In the first step (day), the **starting inventory** of the store is determined; then, **at a generic step**, a **restock request** to the DC is placed accordingly to the restock period **and the demand of the past days is updated** → **step()**

Assignment
Agents: attributes and methods

Class DistributionCenter

ATTRIBUTES

- Expected demand [units] → ***dc_expected_demand***
- Current inventory [units] → ***dc_inv***
- List of pending orders → ***dc_orders_waitlist***
- List of orders currently being processed → ***dc_current_processed_orders***
- Restock amount for the next order to the manufacturer [units] → ***order***
- Amount of requested products that the DC is not able to satisfy in the step [units] → ***lost_restock***

Assignment
Agents: attributes and methods

Class DistributionCenter

METHODS

The **orders** are **processed** according to a FIFO policy, **and order waiting list,** the **dc's and stores' inventory, and lost demand** are **updated, as well as the restock amount to be ordered to the manufacturer**

→ **dc_order_processing()**

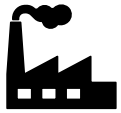
The DC **calculates** the **amount to be ordered and sends** a **restock request** to the manufacturer

→ **dc_request_restock()**

In the first step (day), the **starting inventory** of the DC is defined; then, **at a generic step**, **the lists of pending and processed orders are updated, and** a **restock request** to the manufacturer is placed accordingly to the restock period

→ **step()**

Assignment
Agents: attributes and methods

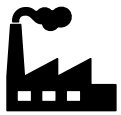


Class Manufacturer

ATTRIBUTES

- Manufacturer's production capacity [units] → ***m.capacity***
- Restock amount requested by the distribution centers [units] → ***restock_request***
- List of orders currently being processed → ***m_current_processed_orders***

Assignment
Agents: attributes and methods



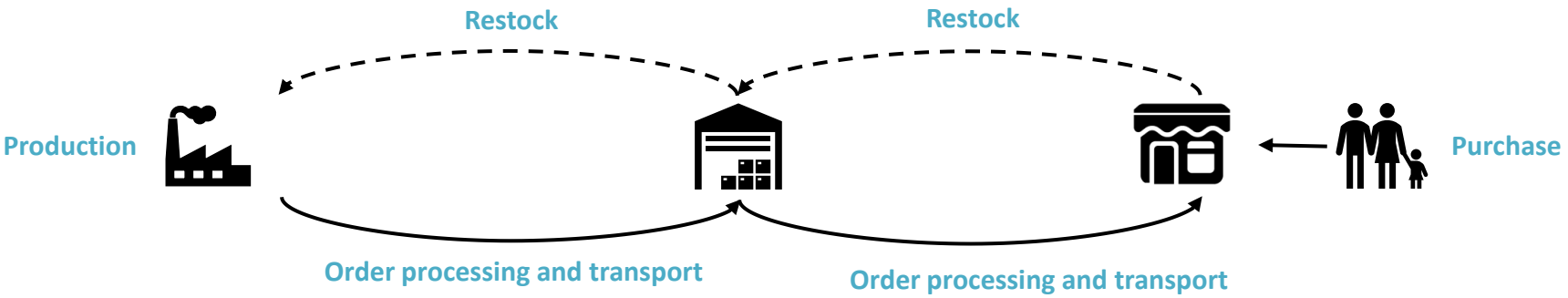
Class Manufacturer

METHODS

The **orders** are **processed** (according to a FIFO policy), **and** the **dc's inventory, lost demand, and order waiting list** are **updated** → **m_order_processing()**

In the first step (day), the **starting capacity** of the manufacturer **is defined**; then, at a generic step, the **lists of pending and processed orders** are **updated** → **step()**

Assignment
Performance measure



$$ITEM\ FILL\ RATE_{DC} = \frac{1 - LOST\ DEMAND}{TOTAL\ DEMAND}$$

Assignment
Simulation model (“BASE CASE”)

Panic Buying Model_base_case.py

```
"""System parameters"""
DC_starting_inventory = 10000
m_starting_capacity = 10000
wh_region = 100                         # Width & Height of a region
w_grid = wh_region*5                    # Full width of the grid
h_grid = wh_region*4                    # Full height of the grid
N_regions = int(w_grid/wh_region)*int(h_grid/wh_region) # Number of regions
f_restock = 1/3                         # Restock frequency [/day]
m_transport_time = 2                    # Manufacturer-to-DC transport time [days]
dc_transport_time = 1                   # DC-to-store transport time [days]

# Number of agents
N_m = 1                                 # Number of Manufacturers
N_dc = 1                                # Number of distribution centers
N_s = 5*N_regions                       # Number of stores
N = 300*N_s                             # Number of customers
customer_store_limit = N_s              # Limit of stores a customer will visit to satisfy the purchase
p_mult = 2                              # Multiplying factor for the increase in buying frequency and quantity of a customer

"""Consumption behavior of customers (equal for each region)"""
consumption_list_region = np.zeros([int(N/N_regions),2])
consumption_list_region[range(int(0.27*N/N_regions))] = [0.333, int(1)]
consumption_list_region[range(int(0.27*N/N_regions),int(0.88*N/N_regions))] = [0.143, 3]
consumption_list_region[range(int(0.88*N/N_regions),int(0.98*N/N_regions))] = [0.067, 6]
consumption_list_region[range(int(0.98*N/N_regions),int(N/N_regions))] = [0.033, 11]
consumption_list_region = consumption_list_region.tolist() #convert a given array to an ordinary list with the same items, e

"""Agent classes definition"""
class Customer(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.q_base = 0             # Customer's regular purchasing quantity
        self.f_base = 0             # Customer's regular purchasing frequency
        self.purchasing_store = 0   # Store in which the customer makes a purchase
        self.q = self.q_base        # Customer's current purchasing quantity
        self.f = self.f_base        # Customer's current purchasing frequency
        self.stores = []            # List of all stores. The list determines the sequence according to which the Customer v
        self.pbs = False            # Customer's panic buying sensitivity

    def store_visiting_list_generation(self):
        """The sequence of stores the Customer can visit is retrieved"""
        s_list = np.array(self.model.schedule.agents[N:N+N_s])      # the Store agents are ret
        self.stores = s_list
        self.purchasing_store=random.choice(self.stores) # the Customer chooses
        self.purchasing_store.store_expected_demand += self.q_base*self.f

    itial_panic(self):
```

```
day 80 regular behavior
day 81 regular behavior
day 82 regular behavior
day 83 regular behavior
day 84 regular behavior
day 85 regular behavior
day 86 regular behavior
day 87 regular behavior
day 88 regular behavior
day 89 regular behavior
day 90 regular behavior
day 91 regular behavior
day 92 regular behavior
day 93 regular behavior
day 94 regular behavior
day 95 regular behavior
day 96 regular behavior
day 97 regular behavior
day 98 regular behavior
day 99 regular behavior
minimal dc ifr =  0.12839113074068842
```

Assignment
Part 1 – Tasks

1. Identify the **"weaknesses" of the given model ("BASE CASE")**, **describing them and commenting on their impact on the model quality** (i.e., accuracy in representing the real-life problem and decision-making)

   Examples of "weaknesses":
   - Performance measures used to study the problem
   - Modeling of the agents' behavior
   - …

2. Create an **updated version of the model that allows achieving:**
   - an **average item fill rate** at the **distribution center higher than 95% before and after the pandemic;**
   - a **minimum item fill rate** at the **distribution center higher than 60% during the pandemic.**

   N.B.: the updated version is obtained by "solving" one or more of the previously identified weaknesses (you will choose which and how many weaknesses to "solve")

Assignment
Part 1 – Output

- **Power point** presentation of **15-20 slides**
- **Python script**
- **Excel file (if needed)**

- The files should be **uploaded on WeBeep (*Assignments* Section) in a zip folder named "Group_XX"**, where XX is the group number

Assignment
Evaluation criteria

- **Fulfilment of the requirements (tasks and deadline)**
- **Quality of:**
  - list of weaknesses and related motivations
  - obtained SC performance
  - updated version of the BASE CASE model
- **Originality**
- **Quality of the .ppt document** (i.e., sequence of the contents, clear description of the work, document readability)

Questions?