

D Product - Inv... X D Inventory Man... X D Production lin... X D Parkings simu... X | Code_lecture... X | LinkedLists - i... X | Opdrachtind... X | +

← → ⌂ dodona.be/nl/courses/4239/series/51166/activities/71750927/

Booking.com AliExpress Addons Store eBay Facebook YouTube ASUS Software Port... MyASUS Software ... McAfee LiveSafe

Tous les favoris

Dodona Algoritmen en Datastructuren > Examen 19 November > Inventory Management

Indienen

Inventory Management

In deze oefening moet je een voorraadbeheersysteem implementeren met behulp van stacks. Het doel is om realistische voorraadoperaties te simuleren, waarbij producten in batches worden beheerd en het Last-In-First-Out (LIFO)-principe wordt toegepast. Deze aanpak weerspiegelt scenario's waarin de meest recent toegevoegde voorraad als eerste wordt gebruikt, zoals bij duurzame goederen.

Implementeer de `Inventory_Manager` klasse (11 punten)

Maak de `Inventory_Manager` klasse (1 punt) die de voorraad beheert voor meerdere producten. Elk object bevat een woordenboek van producten waarbij `product_name` als sleutel wordt gebruikt.

6* Rechercher FRA BEFR 20:28 19-11-24

← → ⌂ dodona.be/nl/courses/4239/series/51166/activities/71750927/

Booking.com AliExpress Addons Store eBay Facebook YouTube ASUS Software Port... MyASUS Software ... McAfee LiveSafe

Tous les favoris

Dodona Algoritmen en Datastructuren > Examen 19 November > Inventory Management

Deze klasse moet de volgende methoden bevatten:

- `add_product(product_name, holding_cost, stockout_penalty)`: Voegt een nieuw product toe aan de voorraad (1 punt). Als het product al is toegevoegd, wordt het volgende bericht weergegeven:

Product [product_name] already exists.

- `restock_product(product_name, quantity, cost_per_unit)`: Vul een product aan door een nieuwe batch aan de stack toe te voegen (2 punten). Als het product niet wordt gevonden, wordt het volgende bericht weergegeven in de console:

Product [product_name] not found

- `simulate_demand(min_demand, max_demand)`: Genereert een willekeurige vraag voor elk product (1 punt). De methode retourneert een woordenboek waarbij de sleutel `product_name` is en de waarde de vraag is die overeenkomt met een willekeurige waarde tussen `min_demand` en `max_demand`. Standaard is `min_demand` gelijk aan 0 en `max_demand` gelijk aan 20.
- `simulate_day(demand)`: Simuleert een dag van operaties en retourneert de totale aanhoudingskosten en totale stockout-kosten (2 punten). De parameter van de methode komt overeen met de gesimuleerde vraag als een woordenboek (zie methode `simulate_demand`).
- `save_to_csv(filename)`: Slaat de voorraad op in een CSV-bestand (1 punt). Elke rij in het CSV-bestand komt overeen met:

D Product - Inv... D Inventory Ma... D Production lin... D Parkings simul... D Code_lecture... D LinkedLists - D Opdrachtind... +

dodona.be/nl/courses/4239/series/51166/activities/71750927/

Booking.com AliExpress Addons Store eBay Facebook YouTube ASUS Software Port... MyASUS Software ... McAfee LiveSafe Tous les favoris

D Dodona Algoritmen en Datastructuren > Examen 19 November > Inventory Management

[product_name],[batch_quantity],[batch_cost_per_unit]

- **load_from_csv(filename)**: Laadt de voorraad vanuit het CSV-bestand (2 punten). Als de producten nog niet in de voorraad aanwezig zijn, worden ze toegevoegd aan de voorraad.
- **print_inventory()**: De methode print de voorraad voor elk product door gebruik te maken van eerder gedefinieerde functies (1 punt). Bijvoorbeeld:

```
Current Inventory:
Product Widget:
Batch(quantity=100, cost_per_unit=2.5)
Batch(quantity=50, cost_per_unit=2.0)

Product Gadget:
Batch(quantity=70, cost_per_unit=3.0)
```

Test je code (2 punten)

Voeg een **main**-methode toe die twee producten toevoegt en voor elk product minimaal 2 batches maakt. Vervolgens moet de voorraad worden gesimuleerd en moet de voorraad opgeslagen in een CSV-bestand.



6° Rechercher FRA BEFR 20:28 19-11-24

D Product - Inv... D Inventory Ma... D Production lin... D Parkings simul... D Code_lecture... D LinkedLists - D Opdrachtind... +

dodona.be/nl/courses/4239/series/51166/activities/1780297260/

Booking.com AliExpress Addons Store eBay Facebook YouTube ASUS Software Port... MyASUS Software ... McAfee LiveSafe Tous les favoris

D Dodona Algoritmen en Datastructuren > Examen 19 November > Product - Inventory Management

Product - Inventory Management

In deze oefening moet je een voorraadbeheersysteem implementeren met behulp van stacks. Het doel is om realistische voorraadoperaties te simuleren, waarbij producten in batches worden beheerd en het Last-In-First-Out (LIFO)-principe wordt toegepast. Deze aanpak weerspiegelt scenario's waarin de meest recent toegevoegde voorraad als eerste wordt gebruikt, zoals bij duurzame goederen.

Implementeer de Batch-klasse (1 punt)

Maak de Batch-klasse die een batch van de voorraad van een product vertegenwoordigt. Elk Batch-object moet bevatten:

6° Rechercher FRA BEFR 20:27 19-11-24

D Product - Inv... X D Inventory Mar... X D Production lin... X D Parkings simul... X | D Code_lecture... X | D LinkedLists - F... X | D Opdrachtind... X | +

← → ⌂ dodona.be/nl/courses/4239/series/51166/activities/1780297260/

Booking.com AliExpress Addons Store eBay Facebook YouTube ASUS Software Port... MyASUS Software ... McAfee LiveSafe

Tous les favoris

Dodona Algoritmen en Datastructuren > Examen 19 November > Product - Inventory Management

Shaka nl

Examen 19 November
Product - Inventory Ma...
Inventory Management

- quantity: de hoeveelheid van het product in de batch
- cost_per_unit: de kosten voor het inkopen van het product in deze batch. Deze kosten worden niet op productniveau opgeslagen omdat ze per batch kunnen verschillen.

De klasse bevat ook een tostring-methode `str` die de batch als volgt weergeeft:

```
Batch(quantity=[quantity], cost_per_unit=[cost_per_unit])
```

Implementeer de Product-klasse (6 punten):

Maak de Product-klasse (1 punt) die een product vertegenwoordigt met een stack van voorraadbatches. Elk Product-object moet bevatten:

- product_name: naam van het product
- batches: stack van batches voor het product
- holding_cost: kosten voor het aanhouden van een eenheid van het product in de voorraad
- stockout_penalty: kosten voor het niet kunnen leveren van een eenheid van het product

Deze klasse moet de volgende methoden bevatten:

- add_batch(quantity, cost_per_unit): voegt een nieuwe batch toe aan de stack (1 punt).
- fulfill_demand(demand): vervult de vraag met behulp van de bovenste batch van de voorraad (2 punten). Als de bovenste batch niet voldoende is, wordt de volgende batch gebruikt. De functie retourneert 0 als de vraag door een of meer batches kan worden vervuld. Als de vraag niet kan worden vervuld, retourneert de functie

6* 20:27 19-11-24

D Product - Inv... X D Inventory Mar... X D Production lin... X D Parkings simul... X | D Code_lecture... X | D LinkedLists - F... X | D Opdrachtind... X | +

← → ⌂ dodona.be/nl/courses/4239/series/51166/activities/1780297260/

Booking.com AliExpress Addons Store eBay Facebook YouTube ASUS Software Port... MyASUS Software ... McAfee LiveSafe

Tous les favoris

Dodona Algoritmen en Datastructuren > Examen 19 November > Product - Inventory Management

Shaka nl

Examen 19 November
Product - Inventory Ma...
Inventory Management

- product_name: naam van het product
- batches: stack van batches voor het product
- holding_cost: kosten voor het aanhouden van een eenheid van het product in de voorraad
- stockout_penalty: kosten voor het niet kunnen leveren van een eenheid van het product

Deze klasse moet de volgende methoden bevatten:

- add_batch(quantity, cost_per_unit): voegt een nieuwe batch toe aan de stack (1 punt).
- fulfill_demand(demand): vervult de vraag met behulp van de bovenste batch van de voorraad (2 punten). Als de bovenste batch niet voldoende is, wordt de volgende batch gebruikt. De functie retourneert 0 als de vraag door een of meer batches kan worden vervuld. Als de vraag niet kan worden vervuld, retourneert de functie stockout_penalty voor het product vermenigvuldigd met het aantal producten dat niet kan worden geleverd.
- calculate_holding_cost(): deze functie retourneert de totale aanhoudingskosten voor dit product door de aanhoudingskosten voor elke batch op te tellen (1 punt).

De klasse bevat ook een tostring-methode `str` (1 punt) die het product als volgt weergeeft:

```
Product [product_name]:  
Batch(quantity=[quantity_batch1 from batch_1], cost_per_unit=[cost_per_unit from batch_1])  
Batch(quantity=[quantity_batch1 from batch_2], cost_per_unit=[cost_per_unit from batch_2])  
....  
Batch(quantity=[quantity_batch1 from batch_n], cost_per_unit=[cost_per_unit from batch_n])
```